



UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

ERROR RESILIENCE FOR OBJECT-BASED VIDEO CODING

Luís Eduardo de Pinho Ducla Soares

(Licenciado)

**DISSERTAÇÃO PARA OBTENÇÃO DO GRAU DE DOUTOR
EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES**

Orientador: Doutor Fernando Manuel Bernardo Pereira

Júri:

Presidente: Reitor da Universidade Técnica de Lisboa

Vogais: Doutor Aggelos K. Katsaggelos
Doutor Carlos Eduardo do Rego da Costa Salema
Doutor Moisés Simões Piedade
Doutor Fernando Manuel Bernardo Pereira
Doutor Pedro António Amado de Assunção
Doutor Mário Alexandre Teles de Figueiredo

ABRIL DE 2004

To my wife Carla.

Acknowledgements

I am indebted to the many people who have made this Thesis possible and, although I would like to express my gratitude to all of them here, it is unfortunately not possible in such a limited space. However, some of these people have played such an essential part in my finishing this Thesis that they at least deserve to have their names mentioned.

The first person I would like to thank is clearly Prof. Fernando Pereira, supervisor of this Thesis. Without his permanent support and enthusiasm, this work would never have been possible. I believe that his work as a supervisor was truly outstanding, exceeding by far the typical obligations of a supervisor. In this aspect, I would say without any hesitation that he is second to none!

I would also like to thank all my colleagues of the Image Group of Instituto Superior Técnico for creating such a friendly and cooperative working environment. In particular, Paulo Nunes and Paulo Lobato Correia deserve a special mention, for their friendship and for their help in solving both technical and logistic problems. Manuel Menezes de Sequeira, João Valentim and João Ascenso were also always very helpful.

Fortunately for me, I was also able to count on the help of members of the international research community from all over the world. In particular, the participants of the MoMuSys ACTS project and the members of the Error Resilience Adhoc Group of MPEG have been of great help, through the fruitful discussions and the effective cooperation. In addition, I would also like to thank all the people at NTT DoCoMo (Japan), where I had the pleasure of working for 4 months, especially Mr. Satoru Adachi with whom I worked closely and Dr. Toshio Miki for inviting me.

Of course, all of the above help would have been wasted if it were not for the PRAXIS XXI scholarship granted by the Fundação para a Ciência e a Tecnologia.

In my personal life, many people have also contributed to make this Thesis possible; even though the help provided was not of a technical nature, it was no less important. In particular, without my wife Carla, this work would have never reached an end.

My parents and my brother have also played an important part in helping me finish all the work, through their permanent support and their help in the most difficult moments.

And last, but certainly not least, I also have to thank all my friends, simply for being there.

Abstract

To transmit object-based video over error-prone networks, object-based error resilience techniques are needed. Recognizing that only very few such techniques are currently available, this Thesis considers the definition of new object-based error resilience techniques. In particular, contributions are made for both the encoder and the decoder side of the video communication chain.

At the encoder side, this Thesis starts by recognizing that the various objects in a given video scene may have very different error resilience needs and, therefore, may need different treatment. With this in mind, a set of metrics that can be used by the encoder to decide how much error resilience should be assigned to each video object is proposed. Additionally, based on these metrics, an adaptive shape and texture intra coding refreshment scheme for object-based video encoders is proposed.

At the decoder side, the importance of the shape data is recognized, because of the subjective impact it has on the user, but also because most object-based functionalities depend on it. With this in mind, two different shape error concealment techniques are proposed, one spatial and one temporal. Additionally, a spatio-temporal technique is also proposed, which basically combines in a quality efficient way the two other proposed techniques.

KEYWORDS

Object-based video coding; Object-based error resilience; Refreshment need metrics; Shape and texture intra coding refreshment; Shape data concealment; MPEG-4.

Resumo

A transmissão de vídeo baseado em objectos através de redes sujeitas a erros exige o uso de técnicas de resiliência a erros adequadas a representações baseadas em objectos. Reconhecendo que muito poucas dessas técnicas existem actualmente, considerou-se nesta Tese o desenvolvimento de novas técnicas deste tipo. Em particular, nesta Tese são feitas contribuições para aumentar a resiliência em ambos os lados da cadeia de comunicação, ou seja no codificador e no decodificador.

No lado do codificador, esta Tese começa por reconhecer que os vários objectos numa dada cena de vídeo têm, tipicamente, características muito diferentes em termos de necessidades de resiliência a erros e, portanto, podem precisar de um tratamento diferenciado. Neste contexto, propõe-se um conjunto de métricas que podem ser usadas pelo codificador para decidir a quantidade de resiliência a erros que deve ser atribuída a cada objecto de vídeo ao longo do tempo. Com base nessas métricas, propõe-se também um esquema adaptativo de refrescamento com base em codificação intra, quer para a forma, quer para a textura, sempre no âmbito dos codificadores de vídeo baseados em objectos.

No lado do decodificador, reconhece-se a importância da informação de forma, devido ao seu enorme impacte subjectivo no utilizador, mas também porque a maioria das funcionalidades baseadas em objectos dependem da sua qualidade e fidelidade. Neste contexto, propõem-se duas técnicas de ocultação de erros para a forma de objectos de vídeo, uma espacial e uma temporal. Além disso, propõe-se também uma técnica espacio-temporal, que basicamente combina de uma forma eficiente, em termos de qualidade, as duas técnicas anteriores.

PALAVRAS-CHAVE

Codificação de video baseada em objectos; Resiliência a erros baseada em objectos; Métricas de necessidade de refrescamento; Refrescamento com codificação intra para forma e textura; Ocultação de erros na informação de forma; MPEG-4.

Contents

Chapter 1	Introduction	1
1.1	Context and Motivation	1
1.1.1	The Object-based Representation Approach	5
1.1.2	Error Resilience for Object-based Video Applications	9
1.2	Objectives and Original Contributions of this Thesis	15
1.3	Outline of this Thesis.....	19
Chapter 2	Error Resilient Video Coding: Organizing and Classifying.....	21
2.1	Introduction	21
2.2	Maximizing the Video Subjective Impact	22
2.2.1	Quality Assessment.....	22
2.2.2	Improving Error Resilience in Video Communication Systems	24
2.3	Error Resilient Encoding	27
2.3.1	Forward Error Resilient Systems	27
2.3.2	Back-Channeling Systems	32
2.4	Error Resilient Decoding and Concealment	34
2.4.1	Error Detection	35
2.4.2	Error Localization	35
2.4.3	Error Concealment	37
2.4.4	Post-Processing Techniques	38
2.5	Error Resilience in Object-based Video Coding Systems	40
2.5.1	Error Resilient Object-based Video Coding Architecture	41
2.5.2	Object-based Video Quality Metrics	43
2.5.3	Error Resilient Object-based Encoding	45

2.5.4	Error Resilient Object-based Decoding and Concealment	47
2.6	Final Remarks.....	52
Chapter 3	Error Resilient Video Coding: A Review	55
3.1	Introduction	55
3.2	Error Resilient Encoding	56
3.2.1	Forward Systems.....	56
3.2.2	Back-Channeling Systems	64
3.3	Error Resilient Decoding and Concealment	66
3.3.1	Spatial Error Concealment.....	66
3.3.2	Temporal Error Concealment	76
3.3.3	Spatio-Temporal Error Concealment.....	80
3.4	Final Remarks.....	82
Chapter 4	Object-based Refreshment Need Metrics.....	83
4.1	Introduction	83
4.2	Critical Refreshment Factors	84
4.3	MPEG-4 Refreshment Constraints	85
4.3.1	Inter Coding of Shape Modes	86
4.3.2	Decodability Dependency on Correctness of Past Information.....	86
4.4	Proposal of Refreshment Need Metrics.....	87
4.4.1	Shape Refreshment Need.....	88
4.4.2	Texture Refreshment Need	112
4.5	Final Remarks.....	123
Chapter 5	Adaptive Object-based Video Coding Refreshment Scheme	125
5.1	Introduction	125
5.2	Combined Shape-Texture Intra Refreshment Procedure.....	126
5.2.1	High Level Description of the Combined Intra Refreshment Procedure.....	126
5.2.2	Encoding Module.....	128
5.2.3	Refreshment Decision Module	129
5.2.4	Step-by-Step Algorithm Specification.....	137
5.3	Performance Evaluation	138
5.3.1	Reference Refreshment Technique	138
5.3.2	Evaluation Methodology.....	139
5.3.3	Test Conditions	144

5.3.4	Simulation Results	147
5.4	Final Remarks.....	164
Chapter 6	Shape Error Concealment for Object-based Video Coding.....	165
6.1	Introduction	165
6.2	Spatial Shape Error Concealment.....	166
6.2.1	Proposal for a Spatial Shape Error Concealment Technique.....	166
6.2.2	Performance Evaluation.....	183
6.3	Temporal Shape Error Concealment	190
6.3.1	Proposal for a Temporal Shape Error Concealment Technique	191
6.3.2	Performance Evaluation.....	210
6.4	Combination of the Spatial and Temporal Concealment Techniques	223
6.4.1	Proposal for an Adaptive Spatio-Temporal Shape Concealment Method.....	224
6.4.2	Performance Evaluation.....	228
6.5	Final Remarks.....	233
Chapter 7	Achievements and Future Directions.....	235
7.1	Discussion of Achievements	235
7.2	Future Directions	239
Annex A	Test Sequences	243
A.1	Akiyo Sequence.....	243
A.2	Children Sequence.....	245
A.3	Coastguard Sequence.....	247
A.4	Cyclamen Sequence.....	250
A.5	Hall Monitor Sequence.....	251
A.6	News Sequence.....	254
A.7	Stefan Sequence.....	257
A.8	Weather Sequence	259
References	261

List of Figures

<i>Figure 1.1 – News sequence and its video objects: (a) Composed scene; (b) Background; (c) Dancers object; (d) Speakers object; (e) Logo object.....</i>	<i>5</i>
<i>Figure 1.2 – Possible changes to the News sequence: (a) Logo is removed; (b) Logo changes position; (c) Logo is removed and ticker bar is added; (d) Speakers are placed on a different setting.....</i>	<i>7</i>
<i>Figure 1.3 – Object-based system architecture [Pereira02]</i>	<i>8</i>
<i>Figure 1.4 – Example of a mobile video telephone [Nokia]</i>	<i>10</i>
<i>Figure 1.5 – Examples of terminals for receiving mobile multimedia broadcasts: (a) Mobile phone [News14]; (b) PDA [Casio]; (c) Car terminal [Clarion]</i>	<i>11</i>
<i>Figure 1.6 – Example of Internet-based personal communication system: (a) Typical PC setup [CSE]; (b) Computer window used for the video communication [Connectix]</i>	<i>12</i>
<i>Figure 1.7 – Example of video broadcast over the Internet: (a) News program [BBC]; (b) Weather forecast [Weather]</i>	<i>13</i>
<i>Figure 1.8 – Examples of a shopping channel on interactive television [TCDesigns]</i>	<i>14</i>
<i>Figure 1.9 – Example of interactive digital television application [Philips]: (a) Position in the track; (b) Speed, gear and rotations per minute.....</i>	<i>14</i>
<i>Figure 1.10 – Examples of surveillance cameras - (a) Surveillance camera in a bus [TransBus]; (b) Babycam surveillance camera [iWL]</i>	<i>15</i>
<i>Figure 2.1 – Simplified architecture of a video communication system</i>	<i>25</i>
<i>Figure 2.2 – Error localization capability of RVLC codes</i>	<i>36</i>
<i>Figure 2.3 – Images before (a) and after a deblocking filter has been applied (b) [Schafer03]</i>	<i>39</i>
<i>Figure 2.4 – Images before (a) and after pixel value post-processing (b)</i>	<i>40</i>
<i>Figure 2.5 – Images before (a) and after continuity post-processing (b)</i>	<i>40</i>
<i>Figure 2.6 – Example of an error resilient object-based video coding architecture</i>	<i>42</i>

<i>Figure 2.7 - Examples of shape artifacts due to errors – (a) Original of an object with two regions; (b) Severe shape distortion in one of the object regions; (c) Appearance of a hole in one of the object regions; (d) Appearance of a new non-connected region; (e) Splitting of a region; (f) Disappearance of one region; (g) Merging of two regions.</i>	<i>49</i>
<i>Figure 2.8 – Two different scene types: (a) Segmented scene; (b) Composed scene.....</i>	<i>51</i>
<i>Figure 3.1 – Layered coding system (two layers) where channels with different channel error characteristics are used</i>	<i>62</i>
<i>Figure 3.2 – Multiple description coding and decoding (adapted from [Wang98]).....</i>	<i>63</i>
<i>Figure 3.3 – Sketch recovery technique [Atzori98]</i>	<i>68</i>
<i>Figure 3.4 – Typical “head and shoulders” natural images: (a) Lena; (b) Salesman</i>	<i>68</i>
<i>Figure 3.5 – Sketch recovery procedure [Atzori98]: (a) Original Maskerade image (detail); (b) Simulated losses of 16×16 blocks; (c) Recovered sketch; (d) Concealed image</i>	<i>69</i>
<i>Figure 3.6 – Illustration of smoothing constraints, where an arrow between two samples means that the difference between these two samples occurs in the smoothness measure: (a) Smoothing constraint imposed only on the block boundary; (b) Smoothing constraint imposed on each sample in the direction towards the nearest block boundary [Wang93]</i>	<i>70</i>
<i>Figure 3.7 – Illustration of the adaptive POCS iterative restoration process [Sun95]</i>	<i>72</i>
<i>Figure 3.8 – Spatial interpolation for error concealment – (a) Block-based; (b) Macroblock-based (adapted from [Aign95]).....</i>	<i>73</i>
<i>Figure 3.9 – Division of a block with 64 DCT coefficients into five different subspectra ..</i>	<i>74</i>
<i>Figure 3.10 – (a) The 16 neighboring shapels used to estimate the value of the lost shapel (in gray) and (b) The 8 possible edge directions used to determine the weight associated with each of the 16 neighboring shapels.....</i>	<i>75</i>
<i>Figure 3.11 – Different modules of the global motion compensation shape concealment method.....</i>	<i>78</i>
<i>Figure 3.12 – Illustration of the global motion compensation method to recover the object contour – (a) Broken contour extracted from a corrupted shape of the Bream object; (b) Reference contour with overlapped end points from the corrupted contour</i>	<i>79</i>
<i>Figure 3.13 – Shape error concealment by global motion compensation as in [Salama02] - (a) Original shape; (b) Corrupted shape; (c) Reference shape; (d) Recovered shape; (e) Contour of the corrupted shape; (f) Differences between original and recovered shapes</i>	<i>80</i>
<i>Figure 4.1 – Shapels used for context computation in the CAE algorithm (the question mark represents the shapel for which the context is being computed): (a) Intra mode; (b) Inter mode (the question mark is co-located with c6).....</i>	<i>86</i>

Figure 4.2 – Shapes with different intrinsic shape complexity: (a) low complexity; (b) high complexity	95
Figure 4.3 – Zero-crossings of the curvature	97
Figure 4.4 – CSS image formation [MPEG7-V]	98
Figure 4.5 – Fish object: (a) original image; (b), (d), (f), (h) and (j) contours with progressive amounts of low-pass filtering (after 3, 13, 30, 45 and 60 passes of the low-pass filter, respectively); (c), (e), (g), (i) and (k) corresponding progressive formation of the CSS representation [SQUID]	100
Figure 4.6 – (a) Video object with (6) holes and (b) its multiple contours (7)	101
Figure 4.7 – Types of shape blocks in a VOP	102
Figure 4.8 – Shapes with different spatial concealment difficulty in their bounding boxes (QCIF) with the coding grid overlaid (16x16 shapels): (a) Cyclamen; (b) Tennis Player; (c) Weather Girl; (d) Logo	103
Figure 4.9 – Shapes with different temporal concealment difficulties in their bounding boxes (QCIF) with the coding grid overlaid (16x16 shapels): (a) frames 0 and 3 of the Cyclamen object; (b) frames 180 and 182 of the Tennis Player object; (c) frames 0 and 3 of the Weather Girl object; (d) frames 0 and 3 of the Logo object; (e) frames 66 and 69 of the Large Boat object (Coastguard sequence); (f) frames 69 and 72 of the Small Boat object (Coastguard sequence)	108
Figure 4.10 – TSCD for the Tennis Player object: (a) QCIF at 15 fps; (b) CIF at 15 fps	109
Figure 4.11 – TSCD for the Small Boat object: (a) QCIF at 10 fps; (b) CIF at 10 fps	110
Figure 4.12 – Determination of $MCATR_N_i$ by motion-compensating $ATR_{N_{i-1}}$ with the texture motion vectors determined for VOP i	112
Figure 4.13 – Macroblocks with different complexities: (a) and (b) macroblocks 71 and 15 of Weather Girl (frame 0, CIF); (c) macroblock 14 of Background (frame 0, CIF); (d) macroblock 14 of Tennis Player (frame 0, CIF)	117
Figure 4.14 – STCD for the Background object of the Stefan sequence: (a) QCIF at 30 fps; (b) CIF at 30 fps	118
Figure 4.15 – STCD for the Kids object: (a) QCIF at 30 fps; (b) CIF at 30 fps.....	118
Figure 4.16 – Samples of texture macroblock pairs from various object sequences: (a) Cyclamen – macroblock 104 of VOP 3 (left) and co-located one in VOP 0 (right); (b) Weather Girl – macroblock 15 of VOP 3 (left) and co-located one in VOP 0 (right); (c) Tennis Player – macroblock 19 of VOP 99 (left) and co-located one in VOP 96 (right); (d) Tennis Player – macroblock 20 of VOP 99(left) and co-located one in VOP 96 (right).....	120
Figure 4.17 – TTCD for the Tennis Player object: (a) QCIF at 15 fps; (b) CIF at 15 fps	121
Figure 4.18 – TTCD for the Background object of the Stefan sequence: (a) QCIF at 15 fps; (b) CIF at 15 fps	122

Figure 5.1 – Multi-object intra refreshment encoder architecture	128
Figure 5.2 – Linear transformation of $ASRN_{i-1}$ values into SRP_i values: (a) Usage of lower SRP range; (b) Usage of full SRP range; (c) Usage of upper SRP range	133
Figure 5.3 – Test sequence samples: (a) Weather; (b) Coastguard; (c) Stefan.....	145
Figure 5.4 - Instantaneous SRP for the Weather sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 10 fps, EC5): (a) Weather Map; (b) Weather Girl	149
Figure 5.5 – Instantaneous SRP for the Coastguard sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 10 fps, EC5): (a) Water; (b) Large Boat; (c) Small Boat; (d) Shore	149
Figure 5.6 - Instantaneous SRP for the Stefan sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 15 fps, EC5): (a) Background; (b) Tennis Player.....	150
Figure 6.1 – Cyclamen object (only luminance) – (a) Original; (b) Extracted edges	167
Figure 6.2 – Cyclamen alpha plane – (a) Original; (b) Extracted contour	167
Figure 6.3 – Proposed spatial shape concealment process	169
Figure 6.4 – Exemplifying the steps of the spatial shape concealment process: (a) Lost blocks surrounded by the available alpha plane blocks; (b) Lost blocks surrounded by the available contours; (c) Interpolated contours inside the lost blocks; (d) Recovered alpha plane	169
Figure 6.5 – (a) Alpha plane with several bursts of lost blocks and (b) Corresponding broken contours	170
Figure 6.6 - Simplified binary alpha plane (white for shapels belonging to the object) ..	171
Figure 6.7 – Different representations of the object contour in Figure 6.6 – (a) Edge representation; (b) Vertex representation; (c) Shapel representation	171
Figure 6.8 - Illustration of contour coupling	173
Figure 6.9 – Relevant cases for the determination of the initial number of points to be used for curve fitting	177
Figure 6.10 - Illustration of contour recovery	178
Figure 6.11 - Steps to obtain the discrete edge representation of the continuous Bézier interpolating contour: (a) Continuous interpolating contour; (b) First three edge site candidates; (c) First chosen edge site and next three candidate edge sites; (d) Final edge representation of the interpolating contour.....	181
Figure 6.12 - Example of shape filling with overlapping contours: (a) Shape filling with artifact on the third line (shown in gray); (b) Correct shape filling	182
Figure 6.13 - Correct shape filling with overlapping contours: (a) Contour overlapping in one edge site; (b) Correct shape filling	182

<i>Figure 6.14 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Akiyo video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3.....</i>	<i>186</i>
<i>Figure 6.15 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Bream video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3.....</i>	<i>187</i>
<i>Figure 6.16 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Stefan video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4</i>	<i>188</i>
<i>Figure 6.17 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Stefan video object with a video packet loss rate of 10% – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4.....</i>	<i>189</i>
<i>Figure 6.18 – Shape of the first VOP of the Bream video object – (a) Original; (b) Corrupted; (c) Concealed.....</i>	<i>190</i>
<i>Figure 6.19 – Shape of the first VOP of the Bream video object concealed with the scheme proposed in [Shirani00].....</i>	<i>190</i>
<i>Figure 6.20 – Proposed temporal shape concealment process.....</i>	<i>192</i>
<i>Figure 6.21 – Exemplifying the steps of the temporal shape concealment process for the Bream video object (CIF, 10 fps): (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Previous alpha plane; (d) Motion compensated previous alpha plane; (e) Concealed alpha plane without local motion refinement</i>	<i>193</i>
<i>Figure 6.22 – Exemplifying the steps of the temporal shape concealment process for the Stefan video object (CIF, 15 fps): (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Previous alpha plane; (d) Motion compensated previous alpha plane; (e) Concealed alpha plane without local motion refinement; (f) Concealed alpha plane with local motion refinement</i>	<i>194</i>
<i>Figure 6.23 – Definition of camera motion [Correia02]</i>	<i>195</i>
<i>Figure 6.24 – Bream video object – (a) Alpha plane; (b) Shapel representation of the contour.....</i>	<i>196</i>
<i>Figure 6.25 – Determination of contour point motion vectors – (a) Current alpha plane with the shape context of the contour point under consideration; (b) Previous alpha plane with two shape blocks perfectly matching the considered shape context; (c) Current texture with the texture context of the contour point under consideration; (d) Previous texture with the two texture blocks whose corresponding shape perfectly matches the considered shape context; (e) Detail of (c); (f) Detail of (d)</i>	<i>198</i>

Figure 6.26 – Creation of the global motion compensated previous alpha plane - (a) Corrupted alpha plane at current time instant; (b) Previous alpha plane to be global motion compensated; (c) Actual global motion compensated previous alpha plane (solid) and the new alpha plane created with only the global motion compensated opaque shapels (dashed)	201
Figure 6.27 – Examples of global motion compensated Bream alpha planes with different parameter values: (a) Original alpha plane without motion compensation; (b) Pan and tilt: $c_1=1.0$, $c_2=0.0$, $c_3=15.0$ and $c_4=30.0$; (c) Zoom: $c_1=0.5$, $c_2=0.0$, $c_3=0.0$ and $c_4=0.0$; (d) Rotation: $c_1=1.0$, $c_2=0.5$, $c_3=0.0$ and $c_4=0.0$; (e) Combination of zoom, rotation, pan and tilt: $c_1=0.25$, $c_2=0.5$, $c_3=50.0$ and $c_4=150.0$	202
Figure 6.28 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane	202
Figure 6.29 – The eight closest neighbors surrounding the concealed shape block shown in dark gray	203
Figure 6.30 – Example of continuity metric computation; the borders across which there are different shapels are shown with a dashed line (this happens for two shapels in the left border and for three pixels in the right border)	205
Figure 6.31 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane with global motion compensation only; (d) Concealed alpha plane after shape blocks with correctly decoded neighbors have been local motion refined	206
Figure 6.32 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane with global motion compensation only; (d) Concealed alpha plane after the shape blocks with correctly decoded neighbors have been refined; (e) Concealed alpha plane after all the shape blocks have been refined	208
Figure 6.33 – Elimination of small opaque region inside the concealed area: (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Concealed alpha plane before small regions elimination; (d) Concealed alpha plane after small regions elimination	209
Figure 6.34 – Elimination of small transparent region outside but in contact with the concealed area: (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Concealed alpha plane before small regions elimination; (d) Concealed alpha plane after small regions elimination	209
Figure 6.35 – Examples of corrupted and corresponding concealed alpha planes and textures for the Akiyo video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3	215

<i>Figure 6.36 – Concealed alpha planes and textures for the Akiyo video object without the small region elimination filter (video packet loss rate of 20%) – (a) Error pattern 1; (b) Error pattern 2</i>	<i>216</i>
<i>Figure 6.37 – Examples of corrupted and corresponding concealed alpha planes and textures for the Bream video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3</i>	<i>217</i>
<i>Figure 6.38 – Examples of corrupted and corresponding concealed alpha planes and textures for the Stefan video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3; (f) Error pattern 4</i>	<i>219</i>
<i>Figure 6.39 – Examples of concealed alpha planes and textures for the Stefan video object without local motion refinement (video packet loss rate of 20%) – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4</i>	<i>220</i>
<i>Figure 6.40 – Examples of corrupted and corresponding concealed alpha planes and textures for the Stefan video object with a video packet loss rate of 10% – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4</i>	<i>221</i>
<i>Figure 6.41 – Proposed adaptive spatio-temporal shape concealment process</i>	<i>226</i>
<i>Figure 6.42 – Examples of corrupted alpha planes where the corrupted areas have the same size, but the length of the corrupted contour segment is very different – (a) Short corrupted contour segment; (b) Long corrupted contour segment</i>	<i>227</i>
<i>Figure 6.43 – Examples of corrupted alpha planes where the maximum distance separating the coupled contour endings are similar, but the size of the corrupted areas is very different – (a) Small corrupted area; (b) Large corrupted area; (c) Larger corrupted area</i>	<i>227</i>
<i>Figure 6.44 – Examples of corrupted and (three) corresponding concealed alpha planes for the Stefan video object, obtained with the proposed spatial concealment technique, the proposed temporal concealment technique and the adaptive spatio-temporal concealment technique – (a) Uncorrupted original alpha plane; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4</i>	<i>230</i>
<i>Figure 6.45 – Examples of corrupted and (three) corresponding concealed alpha planes for the First Man video object from the Hall Monitor sequence, obtained with the proposed spatial concealment technique, the proposed temporal concealment technique and the adaptive spatio-temporal concealment technique – (a) Uncorrupted original alpha plane; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4</i>	<i>232</i>
<i>Figure A.1 – Akiyo sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>244</i>

<i>Figure A.2 – Akiyo sequence: Akiyo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>244</i>
<i>Figure A.3 – Children sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>245</i>
<i>Figure A.4 – Children sequence: Kids alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>246</i>
<i>Figure A.5 – Children sequence: Logo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>246</i>
<i>Figure A.6 – Coastguard sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>247</i>
<i>Figure A.7 – Coastguard sequence: Water alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>248</i>
<i>Figure A.8 – Coastguard sequence: Large Boat alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>248</i>
<i>Figure A.9 – Coastguard sequence: Small Boat alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>249</i>
<i>Figure A.10 – Coastguard sequence: Shore alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>249</i>
<i>Figure A.11 – Cyclamen sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>250</i>
<i>Figure A.12 – Cyclamen sequence: Cyclamen alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>251</i>
<i>Figure A.13 – Hall Monitor sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>252</i>
<i>Figure A.14 – Hall Monitor sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>252</i>
<i>Figure A.15 – Hall Monitor sequence: First Man alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>253</i>
<i>Figure A.16 – Hall Monitor sequence: Second Man alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>253</i>
<i>Figure A.17 – News sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>254</i>
<i>Figure A.18 – News sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>255</i>
<i>Figure A.19 – News sequence: Dancers alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>255</i>
<i>Figure A.20 – News sequence: Newscasters alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>256</i>

<i>Figure A.21 – News sequence: Logo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>256</i>
<i>Figure A.22 – Stefan sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>257</i>
<i>Figure A.23 – Stefan sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249.....</i>	<i>258</i>
<i>Figure A.24 – Stefan sequence: Tennis Player alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249.....</i>	<i>258</i>
<i>Figure A.25 – Weather sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>259</i>
<i>Figure A.26 – Weather sequence: Weather Map alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>260</i>
<i>Figure A.27 – Weather sequence: Weather Girl alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249</i>	<i>260</i>

List of Tables

<i>Table 4.1 – SSCD measures for various shapes at QCIF (176×144) and CIF (352×288) resolutions.....</i>	<i>103</i>
<i>Table 4.2 –TSCD measures for the various shapes shown in Figure 4.9</i>	<i>108</i>
<i>Table 4.3 – Shape refreshment need (SRN) values for the Stefan sequence objects, in addition to their shape error vulnerability (SEV) and scaled temporal shape concealment difficulty (\overline{TSCD}).....</i>	<i>111</i>
<i>Table 4.4 – Shape refreshment need (SRN) values for the Weather sequence objects, in addition to their shape error vulnerability (SEV) and scaled temporal shape concealment difficulty (\overline{TSCD}).....</i>	<i>111</i>
<i>Table 4.5 - STCD measures for texture macroblocks with different complexities</i>	<i>117</i>
<i>Table 4.6 – TTCD measures for different texture macroblocks</i>	<i>121</i>
<i>Table 4.7 – Texture refreshment need (TRN) values for two different macroblocks in the Stefan sequence, in addition to their texture error vulnerability (TEV) and scaled temporal texture concealment difficulty (\overline{TTCD}).....</i>	<i>123</i>
<i>Table 4.8 – Texture refreshment need (TRN) values for two different macroblocks in the Weather sequence, in addition to their texture error vulnerability (TEV) and scaled temporal texture concealment difficulty (\overline{TTCD}).....</i>	<i>123</i>
<i>Table 5.1 – Selected video test sequences</i>	<i>145</i>
<i>Table 5.2 – Bit rate used for each video test sequence</i>	<i>145</i>
<i>Table 5.3 – Average relative contextual relevance for the video objects in the test sequences (QCIF format)</i>	<i>146</i>
<i>Table 5.4 – Test error conditions</i>	<i>146</i>
<i>Table 5.5 – Coding conditions used to determine $ASRN_{min}$ and $ASRN_{max}$.....</i>	<i>147</i>
<i>Table 5.6 – Maximum shape refreshment need for the various error conditions</i>	<i>147</i>
<i>Table 5.7 - Dn values for the Weather sequence video objects</i>	<i>152</i>
<i>Table 5.8 - PSNR values for the Weather sequence video objects</i>	<i>153</i>

<i>Table 5.9 - Dn values for the Coastguard sequence video objects</i>	<i>153</i>
<i>Table 5.10 - PSNR values for the Coastguard sequence video objects</i>	<i>153</i>
<i>Table 5.11 - Dn values for the Stefan sequence video objects</i>	<i>154</i>
<i>Table 5.12 - PSNR values for the Stefan sequence video objects</i>	<i>154</i>
<i>Table 5.13 - Dn values for the Weather sequence video objects</i>	<i>157</i>
<i>Table 5.14 - PSNR values for the Weather sequence video objects</i>	<i>157</i>
<i>Table 5.15 - Dn values for the Coastguard sequence video objects</i>	<i>158</i>
<i>Table 5.16 - PSNR values for the Coastguard sequence video objects</i>	<i>158</i>
<i>Table 5.17 - Dn values for the Stefan sequence video objects</i>	<i>158</i>
<i>Table 5.18 - PSNR values for the Stefan sequence video objects</i>	<i>159</i>
<i>Table 5.19 - Dn values for the Weather sequence video objects</i>	<i>161</i>
<i>Table 5.20 - PSNR values for the Weather sequence video objects</i>	<i>162</i>
<i>Table 5.21 - Dn values for the Coastguard sequence video objects</i>	<i>162</i>
<i>Table 5.22 - PSNR values for the Coastguard sequence video objects</i>	<i>162</i>
<i>Table 5.23 - Dn values for the Stefan sequence video objects</i>	<i>163</i>
<i>Table 5.24 - PSNR values for the Stefan sequence video objects</i>	<i>163</i>
<i>Table 6.1 – Dn values for the tested video object sequences</i>	<i>185</i>
<i>Table 6.2 – Summary of encoding parameters</i>	<i>212</i>
<i>Table 6.3 – Error-free PSNR obtained for the chosen encoding parameters</i>	<i>212</i>
<i>Table 6.4 – Dn values for the tested video object sequences</i>	<i>213</i>
<i>Table 6.5 – PSNR values for the tested video object sequences</i>	<i>213</i>
<i>Table 6.6 – Error-free PSNR obtained for the chosen encoding parameters</i>	<i>222</i>
<i>Table 6.7 – Dn values for the tested video object sequences</i>	<i>222</i>
<i>Table 6.8 – PSNR values for the tested video object sequences</i>	<i>223</i>
<i>Table 6.9 – Dn values for the various concealed alpha planes shown in Figure 6.44, obtained when the various shape concealment techniques proposed in this chapter are used</i>	<i>231</i>
<i>Table 6.10 – Dn values for the various concealed alpha planes shown in Figure 6.45, obtained when the various shape concealment techniques proposed in this chapter are used</i>	<i>233</i>
<i>Table A.1 – Main characteristics of the Akiyo sequence</i>	<i>243</i>
<i>Table A.2 – Main characteristics of the Children sequence</i>	<i>245</i>
<i>Table A.3 – Main characteristics of the Coastguard sequence</i>	<i>247</i>
<i>Table A.4 – Main characteristics of the Cyclamen sequence</i>	<i>250</i>

<i>Table A.5 – Main characteristics of the Hall Monitor sequence</i>	<i>251</i>
<i>Table A.6 – Main characteristics of the News sequence</i>	<i>254</i>
<i>Table A.7 – Main characteristics of the Stefan sequence.....</i>	<i>257</i>
<i>Table A.8 – Main characteristics of the Weather sequence</i>	<i>259</i>

List of Acronyms

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
3G	Third Generation
3GPP	Third Generation Partnership Project
AATRn	Average Accumulated Texture Refreshment Need
AC	Alternating Current
ACR	Absolute Contextual Relevance
AIR	Adaptive Intra Refreshment
ARCR	Average Relative Contextual Relevance
ARQ	Automatic Repeat reQuest
ART	Angular Radial Transform
ASRN	Average Shape Refreshment Need
ASRP	Average Shape Refreshment Period
ATM	Asynchronous Transfer Mode
ATRn	Accumulated Texture Refreshment Need
AV	Audiovisual
AVC	Advanced Video Coding
BAB	Binary Alpha Block
BCH	Bose-Chaudhuri-Hocquenghem block codes
BER	Bit Error Rate
CAE	Content-based Arithmetic Encoding
CD	Compact Disc
CDMA	Code Division Multiple Access
CIF	Common Intermediate Format for images (288 lines by 352 columns)
CIR	Cyclic Intra Refreshment
CODEC	Coder/Decoder
CSS	Curvature Scale Space
DC	Direct Current
DCT	Discrete Cosine Transform
Dn	Shape Distortion Metric
DPCM	Differential-PCM
DSCQS	Double Stimulus Continuous Quality Scale
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
EC	Error Condition

EREC	Error Resilient Entropy Coding
FEC	Forward Error Correction
GSM	<i>Groupe Spécial Mobile</i>
JPEG	Joint Photographic Experts Group
JVT	(MPEG/VCEG) Joint Video Team
JTC	(ISO/IEC) Joint Technical Committee
HDTV	High-Definition Television
HVS	Human Visual System
IEC	International Electrotechnical Commission
ISC	Intrinsic Shape Complexity
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
IST	<i>Instituto Superior Técnico</i>
ITU	International Telecommunications Union
ITU-R	Radio standardization sector of ITU
ITU-T	Telecommunications standardization sector of ITU
LRS	Last Refreshed Shape
MAP	Maximum <i>A Posteriori</i>
MCATRN	Motion-Compensated Accumulated Texture Refreshment Need
MDC	Multiple Description Coding
MPEG	Motion Pictures Experts Group
MRF	Markov Random Field
MSE	Mean Square Error
NRM	Number of Refreshment Macroblocks
NTMR	Number of Texture Macroblocks to be Refreshed
NTSC	National Television Systems Committee
OBMC	Overlapped Block Motion Compensation
PAL	Phase Alternating Line
PAM	Pulse Amplitude Modulation
PC	Personal Computer
PCM	Pulse Coded Modulation
PDA	Personal Digital Assistant
PLR	Packet Loss Rate
POCS	Projection Onto Convex Sets
PSK	Phase Shift Keying
PSNR	Peak Signal-to-Noise Ratio
PSTN	Public Switched Telephone Network
QAM	Quadrature-carrier Amplitude Modulation
QCIF	Quarter Common Intermediate Format for images (144 lines by 176 columns)
QoS	Quality of Service
RCPC	Rate Compatible Punctured Convolutional codes
RCR	Relative Contextual Relevance
RVLC	Reversible Variable Length Coding
SAD	Sum of Absolute Differences
SCD	Shape Concealment Difficulty
SCM	Shape Continuity Metric
SECAM	<i>Séquentiel Couleur avec Memoire</i>
SEV	Shape Error Vulnerability
SDC	Single Description Coder

SGF	Spatial Grid Factor
SI	Spatial perceptual Information
SIF	Source Intermediate Format for images (240 lines by 352 columns)
SRD	Shape Refreshment Decision
SRN	Shape Refreshment Need
SRP	Shape Refreshment Period
SS	Single Stimulus
SSCD	Spatial Shape Concealment Difficulty
STCD	Spatial Texture Concealment Difficulty
SVD	Singular Value Decomposition
TCD	Texture Concealment Difficulty
TEV	Texture Error Vulnerability
TGF	Temporal Grid Factor
TRD	Texture Refreshment Decision
TRN	Texture Refreshment Need
TSCD	Temporal Shape Concealment Difficulty
TSS	Temporal Shape Stability
TTCD	Temporal Texture Concealment Difficulty
UMTS	Universal Mobile Telecommunication System
VCEG	Video Coding Experts Group
VQEG	Video Quality Experts Group
VLC	Variable Length Coding
VOP	Video Object Plane
VP	Video Packet
VSF	Vestigial Side Band modulation
W-CDMA	Wideband CDMA

Chapter 1

Introduction

1.1 Context and Motivation

Some twenty years ago, digital video started to emerge. Back then, digital video was just a digital representation of the corresponding analog video, in the sense that the data model was the same. In fact, both analog and digital video consisted of a periodic sequence of (rectangular) frames or fields and the sole conceptual difference between the two models was the fact that in the analog representation each frame or field was made of a number of (analog) lines, whereas in the digital representation the frames or fields corresponded to matrices of picture elements, also known as *pixels* [BT.601]. Nowadays, this type of digital video is commonly referred to as “frame-based video”.

Although the data model was basically the same, the (frame-based) digital representation of video had several important advantages over the analog one. For instance, since all information could be represented in digital format, similar techniques and systems could be used to process, store, and transmit a large range of digital data types, namely audio and video, which is the basic principle behind *multimedia* (multiple media) [Riley96]. In addition, since digital information was so much easier to process, with the digital representation of video, a multitude of signal processing algorithms could now be used to efficiently perform tasks such as error detection/correction or encryption. Moreover, the degradation typically associated with the transmission or storage of analog information was also easy to avoid because digital information could be simply regenerated (e.g., copies of digital video recordings could be made without any quality loss in video storage archives). However, the digital representation had one clear major drawback. When the analog information was sampled, quantized and then coded to a binary format using Pulse Coded Modulation (PCM) in order to generate the digital representation of the video, a rather high bit rate was obtained, making it very difficult to fit digital video in the frequency band that was used for the analog equivalent, even after adequate modulation.

To illustrate this, the analog Phase Alternating Line (PAL) television standard [BT.470], which has a total of 625 lines and 50 Hz interlaced scanning, can be considered. The baseband bandwidth of the video signal in the PAL television standard is 5.0 MHz for most of continental Europe (i.e., PAL B/G). When this signal is modulated with the recommended Vestigial Side Band (VSB) modulation, the necessary bandwidth increases to 5.75 MHz¹. The digital equivalent of the PAL analog standard is a video signal with a resolution of 720×576 pixels for the luminance signal and 360×576 pixels for each of the two chrominance signals (i.e., 4:2:2 format) with a 50 Hz interlaced scan², as defined in ITU-R Recommendation 601 [BT.601]. If 8-bit PCM is adopted, the obtained bit rate is approximately 166 Mbps. To fit this digital video signal in the 5.0 MHz PAL baseband bandwidth, a Pulse Amplitude Modulation (PAM) signal with at least 131072 levels (corresponding to 17 bits per level) would be needed [Carlson02], which is quite unrealistic. If, on the other hand, modulated transmission is targeted, the digital equivalent of the analog PAL signal has to fit in the 5.75 MHz bandpass bandwidth, which is even harder. In this case, an M -ary modulation, such as M -PSK (Phase Shift Keying) or M -QAM (Quadrature-carrier Amplitude Modulation), with M at least equal to 536870912 (corresponding to 29 bits per modulated symbol) [Carlson02], has to be used, which is even more unrealistic than for baseband transmission. The above results imply that to transmit the same video information, the PCM digital representation would actually require much more bandwidth than the analog equivalent. If a different analog color television standard had been used as example, such as NTSC (National Television Systems Committee) or SECAM (*Séquentiel Couleur avec Mémoire*) both also described in [BT.470], the same conclusion would have been reached.

The “bandwidth explosion” challenge (if a feasible modulation is used) is a problem that experts have been successfully trying to solve for the last two decades, by developing compression techniques that allow a more efficient digital representation of video data. Nowadays, thanks to the great efforts that have been made, the required data storage capacity or bandwidth have been so drastically reduced that it is possible to store or transmit digital video on media that were traditionally used for much less demanding data, such as voice or audio. This is clearly illustrated by the several available (frame-based) video coding³ standards that have arrived to the market, targeting different digital video services, such as the ITU-T H.261 Recommendation [H.261] for video telephony and video conference over Integrated Services Digital Networks (ISDN), the ITU-T H.263 Recommendation [H.263] for video telephony and other video applications over wireless networks and Public Switched Telephone Networks (PSTN), the ISO/IEC MPEG-1 Video standard [MPEG1] for storage of digital video on Compact Disc (CD) and the ISO/IEC MPEG-2 Video standard [MPEG2] for digital television and video storage. As an example, consider MPEG-2 encoded video, which is currently used for television broadcast but also for storage in Digital Versatile Discs (DVD). With a bit rate of only 4 Mbps, it is possible to achieve a very similar quality to the original studio master [Taylor98]. In fact, with a few simple calculations, one can see that with this bit rate it is already possible to fit several digital video signals in the same bandwidth used before for a single analog PAL signal by using a reasonable digital modulation such as 64-PSK or 64-QAM. Today, progress is still being made and the same

¹ In the United Kingdom, the PAL I system is used; in this system, the baseband video signal bandwidth is 5.5 MHz, which leads to 6.75 MHz after VSB modulation.

² With a 25 Hz progressive scan, the same bit rate would be obtained.

³ In this Thesis, unless specifically said so, the term “coding” is used as a synonym of compression or source coding, not to be confused with cryptographic coding.

1. Introduction

DVD-quality can be achieved with only 1 Mbps, as recently shown by VideoLocus Inc. with their highly-optimized H.264/AVC⁴ encoder implementation [D023].

More recently, a new video data representation model has been introduced: the object-based model. In this model, video data is no longer seen as a sequence of frames or fields, but consists of several independent (semantically) relevant video objects that together build the *video scene*. Object-based video coding schemes can also be called content-based video coding schemes because the representation entities in the model — *the objects* — are now very close to the video content since by having semantic value they can be subjected to semantically meaningful actions. This new representation approach allows, in addition to the advantages already provided by the digital frame-based representation, new and improved functionalities in terms of interactivity, coding efficiency and universal access since, for the first time, the content is not only selectively processed but also independently represented, accessible and consumed. This video representation model was first introduced in a large scale with the MPEG-4 standard [MPEG4].

Independently of the video representation model that is adopted, when lossless compression techniques are used, the information is kept the same before and after coding but the number of bits necessary to represent it is reduced in comparison with the corresponding PCM coding. However, it is quite common for some of the original information being compressed to be perceptually irrelevant, which means that bits are wasted on information that the final user cannot see. To avoid this, lossy compression techniques have been developed. These techniques, which correspond to the most important ones in use today, notably in all the standards mentioned before, allow only the perceptually relevant information to be encoded, thus leading to an even higher compression ratio. This way, after compression, either lossless or lossy, each bit carries more information than in the corresponding PCM coding and, therefore, is more sensitive to errors. This can be a problem if real error-prone storage or transmission media are targeted. Being more sensitive to errors means that, if a bit is lost or received in error, the negative subjective impact in the quality of the decoded video is larger (on average). This is why channel coding is typically used after video compression (i.e., source coding), in order to help the receiver to detect and correct as many channel errors as possible. However, the amount of channel errors that is possible to detect and correct is directly related to the amount of additional bit rate that is introduced by the channel coder. This basically means that, in transmission and storage media with severe channel error characteristics and rather limited bandwidth, many errors will typically go uncorrected and be passed on to the video source decoder. This is the reason why such a huge amount of research effort has been devoted, and still is, to developing error resilient video coding techniques that allow compressed video to be decoded with acceptable quality even in the presence of errors.

To better understand the effects that uncorrected channel errors can have on the decoded video quality, a typical frame-based video coding system, relying on (motion compensated) texture predictive coding followed by entropy coding, can be considered. In this type of system, since entropy coding is used, errors in the bitstream will cause the decoder to lose synchronization with the encoder and, if appropriate action is not taken to regain synchronization, the decoder may never recover. Due to this loss of synchronization, visual artifacts will appear in the texture of the corresponding decoded frame (i.e., texture errors). Since predictive coding is used to code the texture within the frame, this means that the errors will propagate to neighbor pixels (i.e., spatial error propagation). If nothing is done, a very

⁴ The emerging H.264/AVC standard represents the state-of-the-art for (frame-based) video coding and was developed by a joint effort of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Motion Picture Experts Group (MPEG) [Schafer03].

large part of the frame may be corrupted. In addition to this, since motion compensation and texture temporal prediction is used to encode the texture of successive frames, these errors will also propagate to the following frames (i.e., temporal error propagation), which can lead to a very negative subjective impact on the final user.

With the emerging of object-based video, new relevant problems have appeared in terms of error resilience, since new types of data, such as the shape and the scene composition information have to be transmitted in addition to the motion and texture already used in previous frame-based coding systems. Moreover, new types of networks are being used to transmit digital video, such as mobile networks and the Internet, which are especially error-prone and can prove to be quite a challenge to anyone working on error resilience and aiming at avoiding unacceptable decoded video quality.

Before going any further, the meaning of some essential terms for this Thesis has to be clearly defined in order to avoid any misunderstandings. When error resilience techniques for video systems are discussed, two terms will invariably appear in the literature. These two terms are: *resilience* and *concealment*. However, their meaning is not always clear and can lead to some confusion. Therefore, definitions are given below, which shall be used throughout the Thesis:

- **Resilience** – According to the *Merriam-Webster's Collegiate Dictionary* [Mish95], *resilience* is “the capability of a strained body to recover its size and shape after deformation caused especially by compressive stress” or “an ability to recover from or adjust easily to misfortune or change.” From these definitions, the meaning of error resilience in the context of video coding systems can be extrapolated to be *the capability of such a system to withstand errors and still perform acceptably*, thus providing a video quality at the decoder that is not much worse than the corresponding quality if no errors were present. Therefore, in this Thesis, when an error resilience technique is mentioned, the intended meaning is that of a technique that improves the capability of the video coding system to withstand errors, either applied at the encoder or at the decoder.
- **Concealment** – According to the same dictionary [Mish95], to *conceal* means “to prevent disclosure or recognition.” Therefore, similarly to what was done above, the meaning of error concealment in the context of video coding systems can be extrapolated to be *the action of hiding the effects of errors*. It has a more “cosmetic” meaning than resilience. Therefore, in this Thesis, when an error concealment technique is mentioned, the intended meaning is that of a technique used at the decoder, which based on the available (correctly decoded as well as corrupted) data received from the encoder will try to hide and minimize the effects of channel errors (residual channel errors if channel decoding is previously used), maximizing the final subjective impact.

Given the definitions above, one can see that the definition of *resilience* is broader than that of *concealment*. In fact, one can speak of a resilient video communication system or simply of resilient video whereas, error concealment is something more specific performed at the decoder to improve the quality of the displayed video. This means that an error resilient video communication system will almost for sure include some error concealment techniques at the decoder, but also other techniques will have to be used to improve its resilience, such as adding synchronization markers in the bitstream or reducing the use of predictive coding.

1.1.1 The Object-based Representation Approach

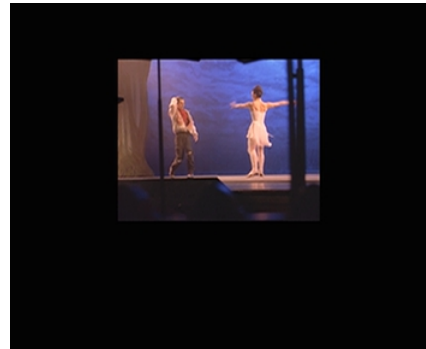
To better understand the concept of video object and object-based coding⁵, imagine a news program, as the one shown in Figure 1.1 (a), represented using an object-based approach instead of a frame-based approach (i.e., the scene is represented as a composition of several objects and not as a periodic sequence of rectangular matrices of pixels).



(a)



(b)



(c)



(d)



(e)

Figure 1.1 – News sequence and its video objects: (a) Composed scene; (b) Background; (c) Dancers object; (d) Speakers object; (e) Logo object

At first sight, this appears to be just a regular news program, consisting of a sequence of frames, with two anchorpersons in a studio reading the news, a large monitor in the back

⁵ In reality, when the object-based representation approach is used, the objects that make up the scene can be video objects, audio objects or even audiovisual objects. However, in this Thesis, only video objects are considered and, therefore, the other types of objects will not be taken into account.

showing video footage related to the news being read and a textual logo on the bottom-left corner of the screen. However, it is not so. This scene is really the composition of four different video objects, also shown in Figure 1.1. Going from left to right, top to bottom, the first object is just a still image, corresponding to the background. The second object is the video footage of two ballet dancers, which are the subject of the news being read. The third object corresponds to two anchorpersons. And finally, the fourth object is a simple text overlay. With the object-based approach, these four objects are independently coded and transmitted to the receiver, along with the composition information, which will allow the receiver to take the four independent objects, decode them and compose the scene that can be displayed on a screen.

If the scene obtained at the receiver is apparently the same as the one that could be obtained with frame-based video, one might ask what is the advantage of using the object-based representation model. The answer is simple and lies in all the new and improved functionalities that can be provided due to separation and not “gluing” approach, which were not possible with frame-based video. For instance, if the user finds that the logo object on the bottom-left corner is annoying, he/she can simply remove it, obtaining the scene in Figure 1.2 (a). Alternatively, if the user does not want to eliminate the logo, he/she can simply drag it to a position that is considered less annoying, as well as remove the ballet dancers in the back, as shown in Figure 1.2 (b). After these changes, if the user is still not satisfied, he/she can replace the logo with more useful information by including a new object such as a stock exchange ticker bar on top of the screen, as can be seen in Figure 1.2 (c). If a more drastic change is desired, the two speakers can be moved to a completely different setting, which is shown in Figure 1.2 (d). Additionally, hyperlinks can be provided allowing the user to access more information about the ballet dancers or the anchorpersons’ biography, depending on where the user clicks on the screen. Moreover, the object-based approach allows the different objects to be encoded with techniques that are more adequate to their nature, which did not happen for frame-based systems. For instance, the text overlay can be coded with text coding techniques instead of using video coding techniques (which are very inefficient to code text), as was (and still is) the case in frame-based systems, such as MPEG-2 Video, where everything is coded together, using the same coding tools.



(a)



(b)

1. Introduction



Figure 1.2 – Possible changes to the News sequence: (a) Logo is removed; (b) Logo changes position; (c) Logo is removed and ticker bar is added; (d) Speakers are placed on a different setting

The object-based video representation model, which was adopted by the ISO/IEC MPEG-4 standard [MPEG4], is the cornerstone of all the new functionalities provided and represents the greatest conceptual difference with respect to previous frame-based standards, namely previous MPEG standards such as MPEG-1 [MPEG1] and MPEG-2 [MPEG2].

In Figure 1.3, a simplified version of the architecture of an object-based coding system is presented. At the transmitter, the various objects in the scene are separately encoded; moreover the composition information is also created. The generated elementary streams (in addition to locally stored elementary streams) are then multiplexed to form a single (multiplexed) bitstream that is sent through the channel and includes all the information about the scene. At the receiver, the received bitstream is demultiplexed in order to obtain the various elementary streams corresponding to the objects and the composition information of the scene. The elementary streams are then decoded and passed on to the compositor, which will compose the scene based on the composition information. Additionally, at the receiver, local coded and uncoded objects can be added to the scene by the compositor; of course, the local coded objects have to be decoded before the compositor adds them to the scene. Moreover, since the several objects in the scene are independently coded, the end-user can also interact with them, performing operations such as changing the spatial position or the size of objects, adding or deleting objects, triggering pre-defined behaviors, following hyperlinks, or even changing color properties. Depending on the type of interaction, the necessary action will be taken either locally (i.e., at the decoder) or remotely (i.e., at the encoder). For instance, if the user chooses to change the spatial position of a given object in the scene, this can be simply taken care of by the compositor. On the other hand, if the user chooses to delete one object, this can be taken care of more efficiently by the encoder, which does not have to send it any longer, thus saving bit rate resources.

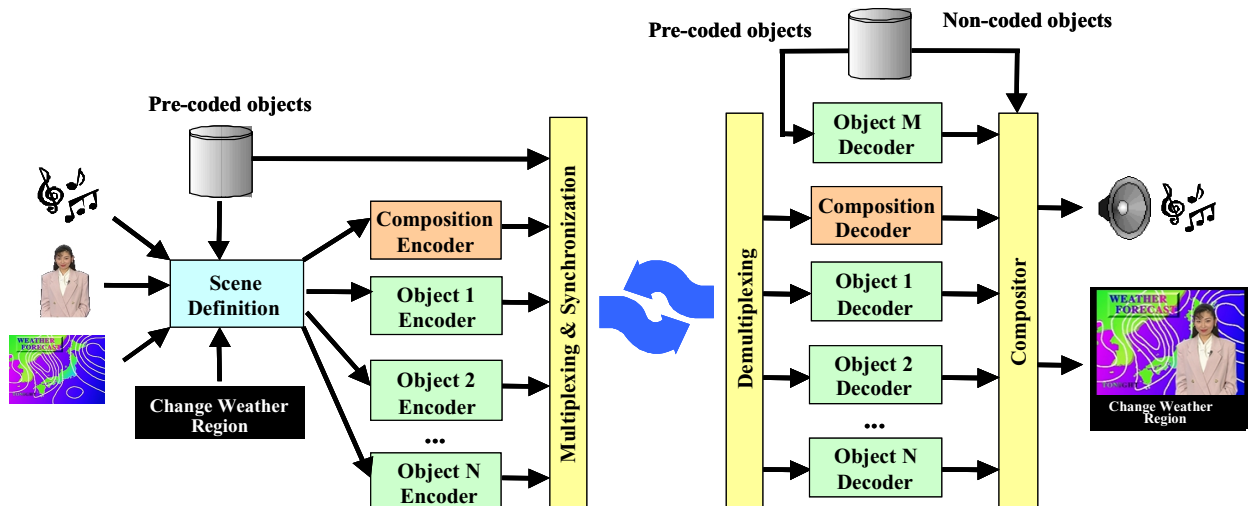


Figure 1.3 – Object-based system architecture [Pereira02]

The fact that a given scene can be modeled as a composition of semantically relevant objects (i.e. relevant objects in the context of a given application) introduces a number of new technical and functional possibilities, namely:

- Selective processing and coding of objects** – With the object-based coding model, it is possible to process and encode different objects with tools that are adequate to their different natures. This alone is enough to make the use of the object-based representation approach worth it because it can bring very significant coding gains or bit rate reductions. For instance, in older standards, subtitles were often encoded using video coding tools. Since video coding tools are not optimized to code text, in order to achieve a reasonable quality, necessary to help the viewers read the subtitles, a lot of bits had to be spent on the subtitles alone. If a fixed bit rate was used, this inevitably resulted in a decrease of the global video quality. With the new object-based representation, subtitles can be independently encoded as text characters, such as Unicode [UNICODE], in addition to some formatting information. This will save an enormous amount of bits that can be spent elsewhere, providing at the same time a better quality. Of course, this does not apply only to subtitles and can also bring benefits when other types of objects are used, such as television station logos, graphics or, more generally, synthetic content.
- Reusability of objects** – With object-based coding, since an independent bitstream is generated for each object, the objects in a given scene are independently accessible, which makes it very easy to reuse a given object from a scene in another scene. This offers many new possibilities in terms of content creation and makes it easier than ever. For instance, it becomes now possible to create new content based only on already existing objects, eliminating the need to film new similar ones, thus saving a lot of time and money (although maybe creating some new intellectual property management and protection issues). This way, by using objects stored in large databases and possibly adding new ones, if the means to create them are available (e.g., digital cameras), anyone may become a content creator, capable of creating rather complex and high quality content that can be easily distributed and published using the Internet.
- Integration of synthetic and natural content** – With the object-based model, it is possible to efficiently integrate in the same scene natural (i.e., shot with a camera) and synthetic objects. For instance, it is possible to have scenes where cartoon characters coexist with live actors, each being encoded with adequate tools. The synthetic characters

can be encoded as three-dimensional (3D) wire frame models, thus allowing new possibilities in terms of interaction, typically not available with natural objects. For instance, a synthetic 3D object can be easily rotated, thus uncovering regions of the object that were not previously visible. This is possible because the synthetic object is simply a fully specified 3D model, which is then rendered in the chosen position.

- **Interaction with and between objects** – Since the objects are independently coded, the user can interact with the scene in several ways. For instance, the user can change the spatial position of a given object, find more information about it by clicking on it, or even modify the perspective with which he/she is looking at the (3D) scene. In addition, objects can also interact with each other. For instance, it is possible for an object to have its trajectory changed due to a collision with another object. Therefore, with this type of interaction with and between objects, if some of the objects in the scene are buttons, it becomes possible to create many interesting applications where everything is encoded with a single standard (e.g., MPEG-4); to trigger a predefined behavior, all the user has to do is to click on the corresponding button.
- **Universal accessibility to the content** – With the object-based model, true universal accessibility may finally become a reality, meaning that the content is available through any type of network and accessed with any type of terminal. Since some of the targeted networks, such as mobile networks, have very critical channel error characteristics, sufficient error robustness has to be provided. However, error resilience always comes at a price in terms of additional bit rate. Since in many of these networks the bit rate is also very limited, this usually meant for frame-based systems that in order to improve error resilience the uncorrupted quality had to be sacrificed, sometimes below acceptable levels. With object-based coding systems, other approaches can be used. For instance, error resilience can be selectively introduced, making more robust to errors the more important objects, without sacrificing their quality at all; the quality of less important objects can be decreased, instead. Still another possibility is to consider content scalability, which basically means that only the more important objects will be sent to the receiver, in order to guarantee that the received content has acceptable quality. This last approach may also be the solution when terminals with reduced complexity are used, which do not have enough resources to decode the whole scene.

Due to these and other technical and functional possibilities, many new applications are starting to appear, which are based on the new object-based representation approach. The next section will show how important error resilience tools are for the deployment of object-based video applications.

1.1.2 Error Resilience for Object-based Video Applications

The advantages of the new object-based representation model are independent of the bit rate used for coding, which can go from the very low bit rates typically used in mobile communications to the high bit rates necessary for studio applications. This makes the new object-based coding approach useful for a great diversity of applications, many of which rely on underlying error-prone networks to deliver the encoded data and, therefore, need adequate error resilience techniques. Below, some examples are given of applications benefiting from the use of the object-based approach where the use of error resilience techniques is more or less important, but is always needed:

- **Mobile personal video communications** – With the emergence of third generation (3G) mobile communication systems, it is expected that personal video

communications, such as video telephony and video conferencing, will finally become available to the mobile user. By using an object-based coding approach, the user experience for this type of applications can be greatly enhanced. For instance, in typical mobile personal video communications it is expected that the background will change much faster than the person actually speaking (e.g., urban scenarios with many people in the back), thus making it harder to code and requiring most of the available bits. However, by encoding the person and the background as separate objects, a much higher bit rate can be assigned to the person (i.e., the most important object), thus increasing its quality at the expense of the background quality. If this is still not enough to achieve an acceptable decoded quality for the person, the background can be simply discarded and not transmitted at all; the decoder can also locally add a static or moving background, which comes for free in terms of transmission resources. Additionally, since these applications are surely among the most demanding ones in terms of error resilience, some resources will have to be used for error resilience. However, since the two objects have been separately encoded and the person is the most important one, most of the error resilience resources can be assigned to it. There are three main reasons that make these applications so demanding in terms of error resilience. The first one is the very high bit error rate associated with mobile networks. Secondly, the available bit rate in this type of networks is very limited, which means both that each bit carries a lot of information (due to the very high compression ratios that are used) and that extra data introduced for error resilience purposes has to be used sparingly. And, finally, due to the fact that these real-time communications are bi-directional, there are also very strict delay constraints, meaning that error resilience techniques which need to delay the arrival of the video encoded data at the decoder cannot be used; otherwise interactivity between the people communicating may be compromised. In addition to this, the computational resources available in the mobile terminal are typically limited (in part due to the limited battery life), making the situation even harder to cope with. A prototype of a mobile video telephone, similar to those that will be used in the near future for video telephony and video conference is shown in Figure 1.4.



Figure 1.4 – Example of a mobile video telephone [Nokia]

- **Mobile multimedia broadcast** – The enormous popularity of mobile phones and personal digital assistant (PDA) terminals shows the interest that users have for this

1. Introduction

type of equipment and the associated applications. With the emergence of third generation mobile communications systems, the users can expect many new multimedia applications to be available in their mobile terminals. A typical example, which in fact is already starting to appear, is the broadcast of multimedia content, such as news programs, weather reports, sports events and music videos. In this type of applications, the use of object-based representation is greatly beneficial. For instance, if the broadcast of a music video is considered, with the object-based representation approach, it is possible to encode all the musicians on the video as separate objects. This allows the user to interact with the video in many new ways, such as accessing information on each musician by clicking on it with a pointing device, changing the position of such musician in the video or even changing the volume of the instrument the musician is playing. Additionally, similarly to what was said for the mobile personal video communications application, if the available bit rate is small, it is possible to transmit only the most important musicians (e.g., the one playing a solo) or to enhance the quality of those musicians at the expense of others, as well as to improve their resilience to errors. In terms of error resilience, due to the unidirectional nature of this application, the delay constraints are not as severe as for the mobile personal communications since a larger initial delay is acceptable. Nevertheless, the use of error resilience techniques is extremely important since the remaining problems typically associated with mobile networks still have to be considered: highly unreliable channels due to the very high bit error rate, limited available bit rate, limited computation resources and limited battery life. The fact that the Third Generation Partnership Project (3GPP) consortium recently adopted MPEG-4 video coding technology is a first indication that MPEG-4 content will have a predominant role in the future Universal Mobile Telecommunication System (UMTS). Examples of terminals that can be used to receive multimedia broadcasts are given in Figure 1.5, including a mobile phone showing a news program in Figure 1.5 (a), a PDA terminal showing a music video in Figure 1.5 (b) and a car video display in Figure 1.5 (c).



Figure 1.5 – Examples of terminals for receiving mobile multimedia broadcasts: (a) Mobile phone [News14]; (b) PDA [Casio]; (c) Car terminal [Clarion]

- **Personal video communications over the Internet** – Video transmission over the Internet is becoming increasingly important. In fact, it is becoming quite normal to use a typical personal computer (PC) as a video telephone or a video conference terminal. Similarly to what happens for mobile channels, these applications over the Internet, which represent some of the most demanding applications in terms of error resilience, can greatly benefit from the use of object-based coding. In the Internet, due to network congestion, the bandwidth varies a lot over time in addition to still being

rather limited. Since for this application example the communication is bi-directional and thus has very strict delay constraints, this can lead to packet losses during transmission. Therefore, if the speaker and the background are encoded as separate objects, the sending terminal will be able to adaptively decide the amount of bit rate assigned to each one, as well as the error resilience resources, while favoring the most important object (i.e., the speaker). In terms of error resilience, an important difference between this type of personal video communications and those typically used over mobile networks is the greater computational resources available in a PC, allowing more complex error resilience techniques to be implemented. Figure 1.6 shows an example of a PC-based personal communication system. While Figure 1.6 (a) shows the whole setup, Figure 1.6 (b) shows the computer window used for the video communication with more detail.



Figure 1.6 – Example of Internet-based personal communication system: (a) Typical PC setup [CSE]; (b) Computer window used for the video communication [Connectix]

- Video broadcast over the Internet** – Video transmission over the Internet is not only becoming important for personal communications, as can be seen by the popularity of some broadcast programs that are transmitted, such as news, movies and live concerts. The benefits that the object-based representation approach can bring to these applications are basically the same that were already mentioned for mobile multimedia broadcasts. In terms of error resilience, similarly to personal communications over the Internet, this application is also very demanding, even though it does not have the same severe delay constraints and some higher initial delay is acceptable. On the other hand, the user expects for this type of applications a better video quality than what is typically achieved for personal communications. Therefore, increased error resilience capabilities can be very beneficial in terms of the received video quality. In Figure 1.7 (a) an example is given of a news broadcast over the Internet, while in Figure 1.7 (b) a broadcast of the weather forecast is shown.

1. Introduction

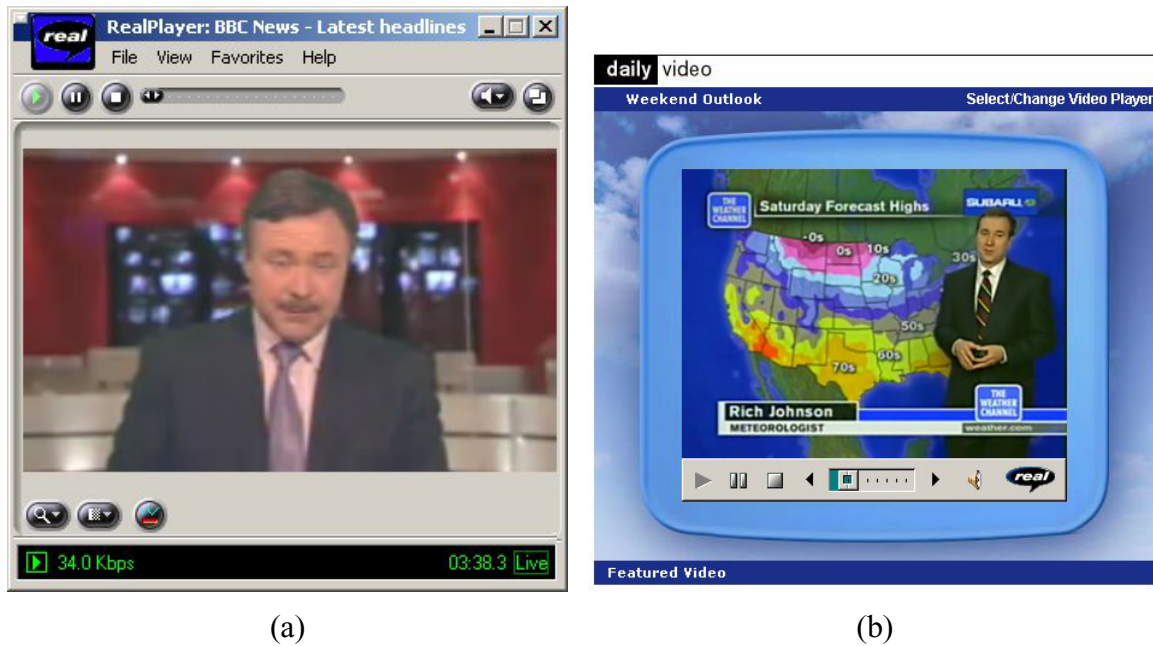


Figure 1.7 – Example of video broadcast over the Internet: (a) News program [BBC]; (b) Weather forecast [Weather]

- **Digital interactive television** – The widespread use of the Internet has created an increasing interest for interaction with the content, which can also be provided by digital television. This way, it becomes possible for the user to manipulate text, images, video, audio and graphics in order to customize the content. Examples of possible manipulations include adding or removing objects, altering the spatial positioning of the objects in the screen and adding extra information to the program at hand. The information to be added can be either related or totally unrelated to the program, such as the actors' biographies, advertisements for different groups of users, sports statistics or even real-time stock exchange quotes. In addition, hyperlinking mechanisms can be added to allow the user to access data, for example, about a given player in a sports broadcast or about an actor in a movie, by accessing his personal web page (using the back-channel) or additional objects that contain this information (text and graphics objects are usually very efficient in terms of bits since specific coding tools can be used). Although the typical channels used for this type of service are not among the most critical in terms of channel error and bandwidth characteristics (e.g., cable networks), the user expects here a very high quality and therefore error resilience techniques are still important (notably for satellite and terrestrial networks). In Figure 1.8, an example is given of a shopping channel on interactive digital television. Here the user can browse a clothing catalog and watch a small clip of the piece he/she is interested in buying. To select a given clip, the user simply has to click on one of the available buttons on the screen, which are also objects with a given predefined behavior (i.e., to play a given clip when clicked on). Additionally, if the clips also use object-based coding, the user can interact with the content. For instance, the user can click on a given sweater and change its color to any of those available in stock. In Figure 1.9, an example of an interactive motorcycling racing application is shown. In this application, a viewer can use his remote control to customize his entertainment experience by accessing real-time information on various motorcycling teams, such as the current position in the track (see Figure 1.9 (a)) or telemetry data being sent from the motorcycles (see Figure 1.9 (b)): which gear the

rider is in, how fast he is going and how many revolutions per minute the motorcycle engine is making.



Figure 1.8 – Examples of a shopping channel on interactive television [TCDesigns]

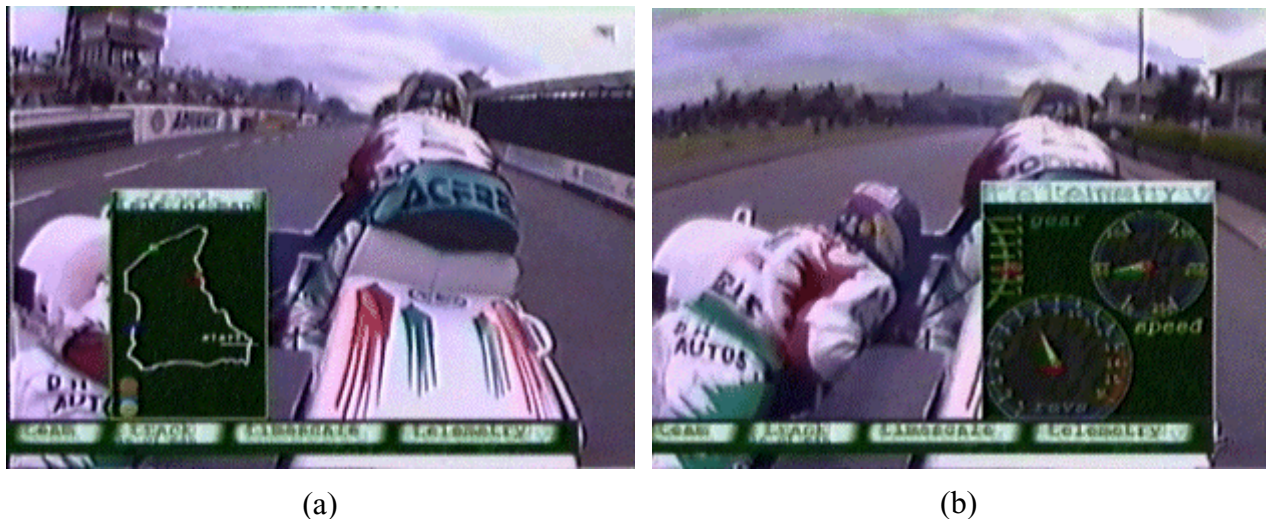
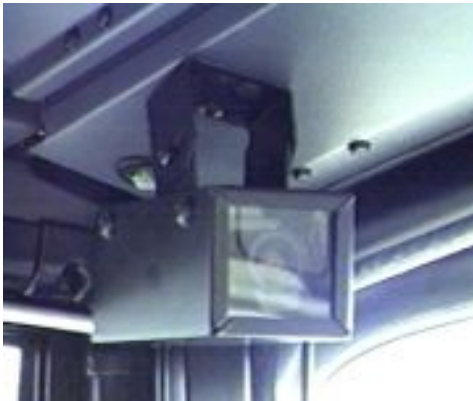


Figure 1.9 – Example of interactive digital television application [Philips]: (a) Position in the track; (b) Speed, gear and rotations per minute

- Remote surveillance** – Error resilience can also be very helpful in remote surveillance applications. In this type of applications, remote cameras transmit video data to a control center. Although in some cases, the transmission will be made over a guided medium such as coaxial cable, other cases exist where the transmission is made over a medium with much more severe error characteristics such as a wireless link (e.g., surveillance cameras on city buses). Also for this type of applications, the use of object-based coding can bring benefits. For instance, a surveillance camera in a bus can be used (with the help of a human operator or not) to identify a suspicious subject traveling onboard, which can then be encoded as a different object from the rest. Since the available bit rate is typically limited in these situations, most of the bit rate and the error resilience resources should be assigned to the suspicious subject in order to achieve an acceptable quality that will later allow the police to identify the suspect. In Figure 1.10 (a), an example is shown of a surveillance camera in a bus in the city of Paris. Another example of a surveillance application is shown in Figure

1. Introduction

1.10 (b), which is the use of surveillance cameras to help young mothers at work monitor their babies and the babysitter.



(a)



(b)

Figure 1.10 – Examples of surveillance cameras - (a) Surveillance camera in a bus [TransBus]; (b) Babycam surveillance camera [iWL]

1.2 Objectives and Original Contributions of this Thesis

The representation of a video scene as a composition of several objects, each one with different properties and associated interactive behavior, is assuming an increasingly important role in emerging multimedia applications. In addition to this, today's user wants to have access to those applications anywhere and at anytime, which means that the video content may have to be delivered over a multitude of networks. Since some of the targeted networks have very critical channel error and bandwidth characteristics, very challenging situations in terms of error resilience are created, leading to the focal point of this Thesis: error resilience for object-based video coding systems. This subject is important because, without error resilience algorithms, object-based content cannot be delivered over error-prone networks, while achieving an acceptable decoded video quality at the receiver. Simply put, error resilience techniques can make the difference between being able to transmit object-based content over error-prone networks or not.

In this context, it should be noticed that the global improvement of error resilience for video coding systems can be achieved by improving or adding error resilience techniques at two different places in the communication chain:

- **Error resilience at the encoder** – At the encoder side of the communication chain, the target of error resilience techniques is to make the coded bitstream more resilient to errors in order to allow the decoder to better recover in case errors occur; these techniques may be called *preventive error resilience techniques*. If a back-channel exists from the decoder, some of the encoder error resilience decisions can be taken based on the feedback information.
- **Error resilience at the decoder** – At the decoder side of the communication chain, on the other hand, the target of error resilience techniques is to take all the available received data (correct and corrupted) and decode it with the best possible video quality, thus minimizing the negative subjective impact of the errors in the video quality offered to the user; these techniques may be called *corrective resilience techniques*.

In order to have a powerful error resilience solution, video coding systems should include both types of error resilience techniques. This is important because, by working together, the encoder and the decoder will be able to achieve the best possible results. In fact, this is so much true that, if not appropriately helped by the encoder, a decoder will typically be helpless and achieve only very poor decoded video quality.

Keeping in mind what has been said above, the objectives of this Thesis have been defined to include both types of techniques. Below, the major objectives of this Thesis are listed; following each objective, a brief summary of the original contribution provided in this Thesis is made:

- **Proposal of error resilience metrics for object-based encoders** – The first place to start thinking about error resilience is clearly at the encoder. However, in object-based video, the various objects present in a given video scene may have very different characteristics in terms of error resilience needs (e.g., some objects may be inherently very easy to conceal in case of errors, while others may be hard) and, therefore, may need different (error resilience) treatment. This way, if the encoder has a set of metrics to evaluate the error resilience needs of each object, the performance of the object-based video coding system can be greatly improved. With this set of metrics, whose definition was an objective of this Thesis, the encoder should be able to decide the adequate amount of error resilience that should be introduced in each one of the generated individual bitstreams. In predictive coding systems, such as those in use today, a possible way to introduce error resilience in a bitstream is to perform frequent intra coding refreshments, in order to refresh the decoded video quality and avoid long error propagation. With this in mind, the first contribution of this Thesis is a set of encoder metrics that can be used by the encoder to decide how much error resilience should be introduced in the coded elementary bitstreams, in the form of intra coding refreshment. Since the visible video data is basically the shape and texture of the various objects in the scene, these metrics are proposed for both the shape and texture data of video objects and are, respectively, called *shape refreshment need* and *texture refreshment need*. The shape refreshment need metric is based on two other metrics: the *shape error vulnerability*, which basically measures the probability of losing part of the shape data, and the *shape concealment difficulty*, which measures how hard it is to conceal the shape data in case it is corrupted. Similarly, the texture refreshment need metric is based on the *texture error vulnerability* and the *texture concealment difficulty*. At the time of writing, these metrics are the only ones of their kind available for error resilient object-based video coding. The publications related to this work are [Soares01][Soares02][Soares02a][Soares03].
- **Proposal of error resilience techniques for object-based encoders** – Based on the error resilience metrics described above, the encoder should be able to efficiently decide the amount of error resilience that should be included in each object elementary stream in order to make it sufficiently robust against errors. Therefore, an objective of this Thesis was to define an algorithm to make these decisions and define an adequate allocation of error resilience resources. This way, based on the encoder metrics mentioned above, which consider the introduction of error resilience in the form of intra coding refreshment, an intra coding refreshment scheme for object-based video coding is proposed. At each time instant, this scheme decides which video objects present in the audiovisual scene should have their error resilience improved by intra refreshing the shape and/or texture data, possibly at the expense of other objects that can tolerate not to have their shape and/or texture data refreshed so much without

significantly affecting their error resilience. As for the previous metrics, the proposed intra coding scheme is the only one available in the literature dealing with both shape and texture data. This scheme and the metrics above are the contributions made in this Thesis regarding encoder error resilience. The publications related to this work are [Soares02b][Soares04b].

- **Proposal of shape error resilience techniques for object-based decoders** – After the encoder has added an adequate amount of error resilience to the coded bitstreams and transmitted them, it is time for the decoder to go to work and start decoding them, while minimizing the negative impact of any channel errors that might have corrupted the bitstream. In order to do this, the decoder must use error resilience techniques allowing it to improve the decoded video quality. While a lot of texture error resilience techniques exist, which have been developed for frame-based video coding systems but can be easily adapted to work also for object-based video coding systems, almost no shape error resilience techniques exist in the literature. Therefore, the third objective of this Thesis was to propose effective shape concealment techniques. In fact, two different shape error concealment techniques are proposed. While the first one is a spatial concealment technique, in the sense that it takes advantage only of the information at the time instant in question, the second one is a temporal concealment scheme, which is able to use the shape information available from previous time instants. The spatial concealment technique is based on the interpolation of broken contours (due to channel errors) with Bézier curves. As for the temporal concealment technique, it is based on the global motion compensation of the shape data from previous time instants, followed by a local motion correction. Additionally, a spatio-temporal technique is also proposed, which basically combines in a quality efficient way the spatial and temporal techniques mentioned above. The publications related to this work are [Soares04][Soares04a][Soares04c][Soares04d][Soares04e]. In parallel to this work, another leading research group has also followed a similar line of research in terms of spatial shape concealment, which also led to several publications [Li02][Li02a][Schuster03][Schuster04]; this helps further emphasize the importance of shape concealment techniques.

Since MPEG-4 is the only available object-based video coding standard, the techniques proposed in this Thesis are integrated and tested in the context of an MPEG-4 video encoder and decoder. However, any other object-based coding system could have been used; only small adjustments to the proposed error vulnerability metrics (on which the refreshment need metrics depend) would be necessary, since they are directly related to the used coding syntax. Additionally, since the proposed techniques only deal with non-standardized parts of the video coding system, MPEG-4 compatibility is in no way compromised and all the generated bitstreams are fully compliant (as they would be compliant if another coding solution would have been used).

In addition to the work briefly described above, some other work was done during this Thesis. Since this work does not directly fit in the major objectives of the Thesis, as presented above, and for the sake of space, only a brief description will be made here. This work includes:

- **Study of the error resilience and concealment performance of frame-based MPEG-4 video coding** – The performance of the MPEG-4 error resilient frame-based syntax was studied with a meaningful set of test conditions and three different concealment schemes. This is a relevant case in terms of MPEG-4 since it corresponds to the Simple Visual Profile [MPEG4], addressing simple mobile

communications (no arbitrarily shaped video objects are supported). The results have shown that MPEG-4 can provide very acceptable quality in environments with very severe error characteristics, such as mobile environments and, therefore, MPEG-4 provides also an adequate frame-based solution for video over wireless channels. This work was published in [Soares99].

- **Performance tests using a complete MPEG-4 frame-based video coding system** – In June 1998, MPEG organized a set of video error resilience tests in order to show the world that (frame-based) MPEG-4 video could be used in error-prone mobile environments [N2165]. For these tests, a complete MPEG-4 video communication system was implemented, including all the necessary layers of the system from the video encoder/decoder down to the physical layer. For these tests, Instituto Superior Técnico (IST) and Motorola Corporation (in the US) have created a partnership, where the author of this Thesis was responsible for all video coding and decoding, while Motorola was responsible for the lower layers, such as network adaptation, multiplexing and simulation of the physical layer. While the results of these tests are presented in [N2604], a more specific description of the work done by the author of this Thesis can be found in [Soares98].
- **Proposal of an alternative syntax to the MPEG-4 object-based error resilient video syntax** – Due to the very tight schedule running MPEG activities, the MPEG-4 error resilient video syntax⁶ for object-based video has been defined by simply extending the frame-based video syntax, which already existed at the time. In frame-based video, the syntax is such that the motion and the texture data are transmitted separately (each type of data in its own partition). For object-based, the shape data has been simply added to the motion partition, resulting in one partition for the shape and motion data, and another partition for the texture data. In [Soares98a], an alternative syntax with three different partitions, one for each type of data (i.e., shape, motion and texture), has been proposed for object-based video. The presented results have shown this syntax to be superior in terms of error resilience to the one currently available for MPEG-4 object-based error resilient video.
- **Study of the influence of error resilience encoder parameters on the decoded video quality** – In the informative annexes of the MPEG-4 standard [MPEG-4], several error resilience tools are described, which can be used at the encoder to improve the error resilience of the whole system. In particular, there is a (texture) intra coding refreshment scheme that can be used for MPEG-4 frame-based video. However, to be able to effectively use this intra coding algorithm, it is important to know how to adequately configure its parameters according to the error resilience needs of the situation. Therefore, in [Soares99a] and [Soares00], a study was conducted to understand the influence these parameters can have on the decoded video quality of (frame-based) MPEG-4 video and what values should be assigned to them in different situations. For this, W-CDMA mobile networks were considered, due to their importance in emerging UMTS systems. As for the work, it basically consisted in varying the encoder parameters within reasonable ranges and observing what happened at the decoder, after sending the encoded bitstreams through a simulated W-CDMA mobile network. To simulate the network, a set of realistic error

⁶ In MPEG-4 video, the encoded data can be arranged according to one of two possible modes: the combined mode and the data partitioning mode. Since the data partitioning mode is more error resilient, it is the one typically used in error-prone situations. Therefore, when the MPEG-4 error resilient video syntax is mentioned here, the data partitioning mode of the MPEG-4 video syntax is intended. The difference between the two modes shall be explained later in the Thesis.

patterns provided by 3GPP was used. This work was performed during a 4 month stay at NTT DoCoMo in Japan.

1.3 Outline of this Thesis

This Thesis deals with error resilience for object-based video coding systems. The context and motivation for this work are presented here in Chapter 1, along with the definition of the objectives of the Thesis, a summary of the main original contributions and an outline of the work.

In Chapter 2, the problem of error resilience is dissected and organized. This is first done in a general fashion in order to be able to consider both frame-based and object-based coding techniques. Then, the specific problem of error resilience for object-based video coding systems is studied. In particular, the architecture of an error resilient object-based coding system, including all its components, is analyzed, as well as the problems it faces and what has to be done to solve them.

A review of the most relevant available error resilience techniques is provided in Chapter 3 for both frame-based and object-based video coding systems. The structure adopted for this chapter is very similar to the one of Chapter 2 to help the reader better understand under which category a given error resilience technique is included.

In Chapter 4, the shape and texture refreshment need metrics, already mentioned in the previous section, are derived with the target of expressing the necessity of refreshing the shape and texture data of a given video object. These metrics are based on metrics that measure the vulnerability of the data to channel errors and the difficulty of concealing this data when it is corrupted.

In Chapter 5, a shape and texture intra coding refreshment scheme to be used at the encoder, in order to efficiently improve the error resilience of the generated bitstreams that need it the most, is proposed. This scheme, based on the metrics defined in the previous chapter, will greatly help the decoder to recover when errors occur in the transmission or storage medium, substantially improving the final video quality.

After the decoder has received the encoded bitstreams, which have been made adequately error resilient by the encoder, it is time to deal with the errors that may have occurred in the transmission or storage medium. For this, error concealment techniques, among others, are needed. In Chapter 6, two different error concealment techniques are proposed for the shape data. As mentioned in the previous section, one technique is a spatial technique based on contour interpolation with Bézier curves and the other is a temporal technique based on the global motion compensation of the shape data from previous time instants, followed by a local motion correction. Additionally, a spatio-temporal concealment technique, which basically combines the spatial and temporal techniques above, is proposed.

Finally, Chapter 7 discusses the achievements of this Thesis and identifies some directions for future work.

After the closing chapter, one annex is included (i.e., Annex A). This annex includes a description of all the sequences that have been used throughout the Thesis in order to help the reader rapidly find information on a sequence every time it is referred to in the Thesis.

Chapter 2

Error Resilient Video Coding: Organizing and Classifying

2.1 Introduction

In a Thesis dealing with error resilience for video coding systems, it is of the utmost importance that, before trying to propose any new solutions for the existing problems, these problems be fully understood. For this, the different parts of a video coding system should be analyzed, while trying to understand the error resilience problems that affect each of these parts, how they can be solved and their influence on the complete system. In particular, since the Thesis is about error resilience for object-based video coding systems, the specific problems associated with this type of systems should also be well understood because, after all, they can be significantly different from the ones found in frame-based systems.

This way, in Chapter 2, the problems associated with the error resilience of video coding systems are dissected and the possible ways to solve them are classified according to several categories. This is first done in a general fashion in order to be able to include frame-based as well as object-based coding techniques. Then, the specific problems of error resilience for object-based video coding systems are studied. In particular, the architecture of an error resilient object-based coding system, including all its components, is analyzed, as well as the problems it faces and what has to be done to solve them. However, the actual existing solutions for specific error resilience problems are not mentioned yet; this shall be done later in Chapter 3.

2.2 Maximizing the Video Subjective Impact

One of the main objectives of all video communication systems is to maximize the subjective impact of the decoded video, shown at the receiver. However, this objective may have to be reached simultaneously with other objectives, sometimes contradictory, such as maintaining a constant bit rate or providing content-based interactivity by means of independently coded objects, and thus a compromise has to be looked for.

Additionally, in any real communication, whether the data is being stored or transmitted, errors are most of the times inevitable and, therefore, so are error resilience techniques. Moreover, in the case of a video communication, the data is usually in a compressed format with a high compression ratio and reduced redundancy. This means that the coded video signal is even more vulnerable to errors than its uncompressed version and, thus, an error will affect a larger portion of the original information. Therefore, in error prone conditions, both the sending and receiving terminals have to “work” for the maximization of the subjective impact of the decoded material, in other words for the minimization of the negative impact of the (transmission or storage) media errors. The techniques used for the maximization of the subjective impact in the presence of errors have the main target of minimizing the negative impact of errors, which means bringing the subjective quality as close as possible to the subjective quality reachable if no errors were present.

In order to develop error resilient tools that minimize the negative subjective impact of the errors, a good subjective quality metric should be used. Although the human user is generally considered as the best subjective quality measurement system [BT.500][Pereira97], the subjective quality of a video sequence is many times measured in terms of its peak signal-to-noise ratio (PSNR). This objective analytical metric is undoubtedly simple to compute but it is unfortunately not adequate for many situations, and for sure even less adequate when error corrupted video sequences are being evaluated. For instance, a certain decoded sequence can have a high PSNR and a very negative subjective impact and vice-versa. On the other hand, the PSNR can increase indefinitely although the subjective impact is a limited function and thus cannot go beyond a certain limit. In conclusion, this metric does not perfectly reflect the behavior of the *Human Visual System* (HVS) and other measures that try to better match the HVS behavior have been tested by the Video Quality Experts Group (VQEG) in [VQEG00]. However, it was found that the performance of the tested measures is still statistically equivalent to the PSNR, which explains why the PSNR measure is still generally used to evaluate the different error resilient tools (e.g., within MPEG [N1646]). The general topic of quality measurement shall be treated in the next section.

2.2.1 Quality Assessment

Quality assessment is the procedure by which the quality of an image or video sequence is evaluated. These techniques can be used to evaluate the performance of a new coding scheme, but they can also be used to evaluate the performance of a complete video communication system. Two distinct procedures are available for image and video quality assessment:

- **Subjective assessment** – This type of assessment consists on the evaluation of the image or video quality by a panel of human subjects, for which there are well defined and tested methodologies [VQEG].

2. Error Resilient Video Coding: Organizing and Classifying

- **Objective assessment** – These methods evaluate the quality of a given image or video sequence by means of automatic tools, thus producing an objective quality figure [VQEG]. In general, objective metrics try to model the reaction of the HVS.

These two types of assessment methods have different advantages and drawbacks, which are described next.

2.2.1.1 Subjective Assessment

As said above, subjective assessment methods are based on the evaluation of the decoded image or video content by a panel of human subjects. This type of techniques will decisively produce the best results because the end-users of the video coding systems are used, the humans. However, these techniques also have serious drawbacks, notably the fact that it is very time-consuming and expensive to perform this kind of tests. For the results to be statistically relevant, a carefully selected group of human evaluators with a significant number of members has to be used. Due to the very nature of the tests, results will vary and two different groups will produce different results. To avoid this, the assessment must follow very precise methodologies, both in terms of test environment setup and grading techniques. The most common methodologies in use today (e.g., in MPEG) are the ones described in Recommendations ITU-R BT.500 [BT.500] and ITU-T P.910 [P.910].

Subjective assessment methodologies can be of two types, depending on whether a reference image or video sequence is used or not:

- **Single stimulus subjective evaluation** – In this case, the quality of the image or video sequence being evaluated is determined on its own, with no comparison to a reference. This approach can be used either because a reference image or video sequence does not exist or to simulate a real image/video communication (i.e., the end-user does not have a visible reference). For instance, in the MPEG-4 video verification tests [Pereira02], the Single Stimulus (SS) method proposed in [BT.500] was one of the used methods. In this method, the human subjects are presented with a processed video sequence. After the presentation, a gray frame is shown for some time, during which the subjects have to evaluate the quality of the sequence they have just watched according to a scale also recommended by the ITU. This procedure is then repeated for various video sequences.
- **Double stimulus subjective evaluation** – In this case, the quality evaluation is done by comparing the image or video at hand with the reference. This approach can only be used if the reference exists and it is typically used in situations where explicit comparison is desirable, such as when demonstrating the differences between two competing techniques. During the MPEG-4 video verification tests [Pereira02], several of these methods have been used, of which the most important is the Double Stimulus Continuous Quality Scale (DSCQS) method [BT.500]. In this method, the human subjects are presented with pairs of video sequences. One is the processed video sequence under evaluation and the other is the reference; the order is random and is not disclosed to the subjects. Each pair is presented twice, separated by a short period where a gray frame is displayed. After the second time, the subjects are asked to evaluate the quality of the two video sequences they have just watched on a continuous scale. This procedure is then repeated for various video sequences.

Since the subjective evaluation is so time-consuming and cannot be performed easily everyday, it appears to be adequate for tests at the end of the development stage of some new technology. For example, it is common to have a large battery of subjective tests at the end of

the development of a new MPEG coding standard (e.g., the MPEG-4 video error resilience verification tests [N2165][N2604]). Its main target is to show the world, and more specifically the interested industry, that the new technology can satisfy the established requirements in terms of subjective quality, because after all video coding systems are built for human users.

2.2.1.2 Objective Assessment

As opposed to subjective assessment, objective assessment does not rely on the use of human subjects. Instead, it relies on the use of some analytical expressions, which are used to compute the video quality figures. The fact that objective metrics do not rely on human subjects represents a great advantage because it makes them much less time-consuming and expensive to use.

This type of video quality assessment techniques tries to reproduce the results that would be obtained with subjective testing and, therefore, may not be so good in terms of evaluating the impact on the end-user. However, it has some clear advantages. The main one is that these techniques can be implemented automatically and therefore should be much faster and easier to obtain results; this makes frequent quality evaluation possible during a development process.

Similarly to what was done for subjective assessment, objective assessment methodologies can be of two types, depending on whether a reference image or video sequence is used or not:

- **Single stimulus objective evaluation** – In this case, a reference image or video sequence is not used and the quality metrics of the image or video sequence being evaluated are computed on its own, with no comparison to the reference.
- **Double stimulus objective evaluation** – In this case, a reference image or video sequence is used and the quality metrics are computed by comparing the image or video at hand with the reference.

Since objective evaluation methodologies are fully automatic, only requiring computational power, they can be used even on an everyday basis, as opposed to subjective assessment. This makes objective quality assessment very adequate to be used during the development stage of a new coding or error resilience scheme. In this Thesis, for the reasons highlighted, objective quality assessment will also be the used method to evaluate the proposed techniques. However, to complement the objective results, visual results will be presented whenever this is justified to allow the reader to make a subjective evaluation of the results. As for formal subjective evaluation, it would just be too time-consuming and expensive to be used in the context of a Thesis.

2.2.2 Improving Error Resilience in Video Communication Systems

In order to better understand how the video subjective impact can be maximized at the decoder in the presence of channel errors, a block diagram of a video communication system, such as the one in Figure 2.1, should be first considered.

2. Error Resilient Video Coding: Organizing and Classifying

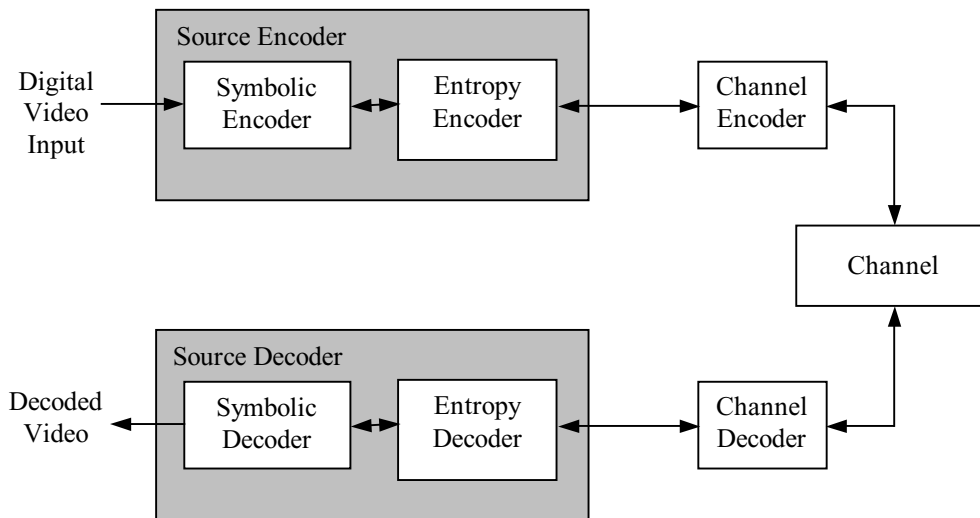


Figure 2.1 – Simplified architecture of a video communication system

In a video communication system, the input digital video is first compressed by the source encoder, which basically consists of a symbolic encoder followed by an entropy encoder. The symbolic encoder generates symbols based on a symbolic representation model for the digital video samples that it receives as input, while eliminating most of the irrelevance and redundancy that the video data may have. Then, the statistical properties of these symbols are exploited in the entropy encoder. Finally, the compressed data output by the source encoder is channel encoded in order to make it more robust to error-prone channel transmission. At the receiver side of the communication chain, the inverse operations are performed in order to obtain the reconstructed video that will then be displayed. Double arrows are used in Figure 2.1 to emphasize the fact that a back-channel may exist to convey feedback information from the decoder to the encoder. When no back-channel exists, a simple arrow should be considered going from the encoder to the decoder.

This way, the maximization of the video subjective impact in the presence of errors or the minimization of the negative impact of the errors may be reached by means of three main types of techniques:

- **Error resilient source coding** – These techniques deal with the conversion of the available digital video (PCM) samples into a suitable (according to the application requirements), efficient, and resilient digital representation. By choosing a more error resilient representation, usually at the expense of some compression efficiency, the negative subjective impact of errors can be minimized. One of the problems of these techniques is that they typically influence the coding syntax that the encoder and the decoder use to communicate and, therefore, have to be standardized. This usually means that after a standard is defined, they can no longer be changed and, thus, remain rather unchangeable. However, there are also some techniques that do not have to be standardized, which are related to the intelligent use of the specified syntax.
- **Channel coding and decoding** – These techniques involve the systematic insertion of extra (redundant) bits in the bitstream. These bits convey no new source information by themselves, but make it possible to detect and correct errors in the bitstream. Channel coding and decoding usually work a layer below source coding and decoding

and these processes can, most of the times, be seen as serial and independent. However, it is largely recognized that joint source-channel coding may bring significant improvements in subjective quality [Wang98]. This means, for example, that different coded data within the same source coding stream, e.g. shape data, motion vectors or DCT coefficients, can be differently protected (in terms of channel coding) since their error sensitivity is different. Since channel coding clearly affects the syntax that the encoder and the decoder use to communicate, it has to be standardized. However, it is often left out of the video coding standards and it is standardized elsewhere (e.g., transport standards). In this case, this gives it some flexibility because the video coding standard and the transport standard can be selected separately.

- **Error concealment** – These techniques cope with the errors introduced by the channel, but not corrected by the channel decoding process or any other means such as retransmission (if available). They process and exploit the available decoded information, both correct and erroneous, with the target of minimizing the video negative impact at the receiver. Since this type of techniques does not influence the coding syntax or the decoding process, it is usually not subject to standardization and it is left as one of the open areas for competition and further improvement.

The techniques mentioned above can be independently used and, typically, a good error resilient solution implies the simultaneous and balanced use of the three types of techniques, depending on the error characteristics, available bandwidth, computational resources, etc. At first, this may sound strange and one may be tempted to think that a channel coding solution alone would be enough, since the channel decoder is able to detect and correct channel errors. Although this is theoretically possible, the amount of errors that the channel decoder is able to correct depends on the amount of extra bits inserted in the bitstream by the channel encoder. Therefore, in environments with severe channel error characteristics, the needed amount of extra bit rate resources may become prohibitive. Additionally, channel coding techniques are especially adequate when the channel errors are uniformly distributed, which is not always the case. Therefore, additional techniques such as interleaving may be needed, which introduce extra delay and need more computational resources.

The three types of techniques referred above clearly highlight two major error resilience coding principles, notably:

- **Essential role of the encoder** – The encoder terminal has a main role on how much error concealment can be done at the receiving terminal since this type of processing has to rely on the received data. The more careful the encoder is with error resilience, the easier the task of the decoder will be. Depending on the applications, the encoder may introduce error resilience based on certain pre-fixed rules or be asked by the receiver to perform actions to prevent or solve specific error related problems.
- **Intelligence of the receiver terminal** – The decoder may use and process the available decoded data (both correct and erred) in a more or less “clever” way. This means that the same corrupted bitstream may result in decoded and concealed video sequences with very different subjective impacts depending on the degree of sophistication of the error processing techniques available.

In the following sections, the techniques presented above will be studied with more detail.

2.3 Error Resilient Encoding

As stated in Section 2.2, the encoding terminal has a major role on how much error concealment can be done at the receiving terminal, since this type of processing has to rely on the received data; the more resilient the received data is, the better the concealment performance at the decoder will be. Therefore, the first place to start thinking about error resilience is at the encoder side of the communication chain, as already illustrated in Figure 2.1. In fact, the more careful the encoder is with error resilience, the easier the task of the decoder will be.

Depending on the applications and on the existence of a back-channel, the encoder may introduce error resilience based on certain pre-fixed rules or be asked by the receiver to perform actions to prevent or solve specific error related problems. This way, two kinds of systems can be identified:

- **Forward systems** – In this type of system, the encoder simply adds error resilience based on a given set of predefined rules and relies on no information coming from the decoder.
- **Back-channeling systems** – In these systems, a back-channel between the decoder and the encoder exists, which means that the decoder can ask the encoder to perform specific actions that will help in the decoding task.

These two types of systems will be further detailed in the following sections.

2.3.1 Forward Error Resilient Systems

In forward error resilient systems, the encoder receives no feedback from the decoder and, therefore, all the encoding decisions are taken based only on *a priori* knowledge. The resilience introduced in this type of systems could be called *preventive resilience*, as opposed to *corrective resilience*, since the encoder is not aiming at helping the decoder to correct a particular error, but generally creating the conditions to help the decoder to recover in the event of an error. This type of resilience involves the intelligent design of the bitstream syntax and of the encoder. Through careful design of these two aspects, it is possible to make significantly easier the task of the decoder in terms of dealing with errors. In fact, the performance of the same decoder can vary greatly depending on the amount of resilience help provided by the encoder. Additionally, by carefully designing the bitstream syntax, it is possible to create a more resilient bitstream with the same amount of bits.

This approach is, in principle, less resilient than the back-channeling approach because the encoder does not know what is happening at the decoder and, therefore, cannot adapt its coding strategy to the current situation. However, in many situations it is the only solution, either because no back-channel exists, such as in video broadcasting, or because the delay constraints are too severe, as in real-time bi-directional communications.

Since the encoder is clearly divided in two blocks, the source encoder and the channel encoder, it is possible to add error resilience at these two levels. In addition, it is also possible to consider another strategy that involves the joint design of the source and channel encoders. These three different approaches will be analyzed next.

2.3.1.1 Source Coding

Even when channel coding protection is used (or to decrease its weight), it is wise to start thinking about error protection at the source coding level, since the encoder has the big advantage of knowing about the coded data involved as well as about its different sensitivity to errors. This means that source coding is the perfect place to introduce error resilience in a bitstream, with a fine granularity, depending on the different data sensitivity, and on each region/object relevance/priority. Similar objectives may be reached with channel coding, although usually with less flexibility and, typically, in a more bit-wise expensive way.

The increasing of error resilience, at the source coding stage, may be reached at two different levels: the symbolic encoder and the entropy encoder. Independently of the level where error resilience is added, these techniques have a cost in terms of coding efficiency, depending on the amount of resilience introduced. This means that the video quality achieved for a certain bit rate without error resilience is lower than the video quality achieved for the same bit rate if error resilience is introduced.

As explained in Section 2.2, the symbolic encoder is the module of the source encoder responsible for converting the digital video samples into symbols, such as motion vectors or DCT coefficients, while removing irrelevance and redundancy from the video data. At this level, it is possible to add error resilience in the following two ways:

- **Reducing the exploitation of video redundancy** – The original video data has usually a very high degree of spatial and temporal redundancy. In order to transmit or store the video material in an efficient way, this redundancy has to be reduced as much as possible. However, redundancy reduction has a price to pay as far as error resilience is concerned, since the entropy value of each bit is higher and thus the video signal becomes more vulnerable to errors. This also usually means that an error in the bitstream will affect a larger portion of the decoded information. To improve the error robustness of the coded video signal, a possible technique is to exploit less effectively the redundancy of the original signal. This can be made in a selective way, also within a frame or object, if some parts are considered more critical or subjectively more important than others. One simple way of applying this technique is by decreasing the use of differential coding, e.g. by avoiding differential shape coding or by periodically intra refreshing all or just the more important macroblocks in an object. In this way, error propagation can be controlled.
- **Using a more resilient prediction model** – By using temporal prediction, high compression gains can be obtained, since typical video sequences have a lot of temporal redundancy. However, if an error occurs in a macroblock that is used for prediction, the error will propagate until the affected area is intra refreshed or predicted from non-erred parts of the image. In order to avoid the long propagation of errors, a more resilient prediction model can be used. Examples of this solution are the Overlapped Block Motion Compensation (OBMC), used in the H.263 standard [H.263], where each pixel's prediction is a weighted sum of predictions using different motion vectors, and leaky-difference schemes where the prediction is weighted with a smaller than one factor, making the dependency on past pixel values disappear after a certain number of frames [Haskell92].

As for the entropy encoder, which is the second module of the source encoder, as explained in Section 2.2, it is responsible for exploiting the statistical properties of the symbols generated by the symbolic encoder. Here, to add error resilience two solutions are possible:

2. Error Resilient Video Coding: Organizing and Classifying

- **Increasing the amount of synchronization data** – One problem that results from the use of entropy coding (which strongly improves coding efficiency) is the possible loss of word synchronization when errors occur. Synchronization will be recovered at the price of introducing resynchronization markers. Since the longer the decoder is out of synchronization, the more data will be lost, the “out of synchronization” time needs to be minimized in order to minimize the lost coded data. To deal with this problem, the amount of synchronization data in the bitstream has to be increased, possibly depending on the data currently being coded. This means that by wisely choosing the spacing between resynchronization markers, as well as the marker length, it is possible to protect better the subjectively more important information. One possible way of doing this is by placing the resynchronization markers periodically, after a certain number of macroblocks (or any spatial entities). However, with this type of spacing, the information will be very unevenly protected since macroblocks may use a very different number of bits. A better way of doing it is by choosing the inter-marker spacing based on the number of bits used, independently of the number of macroblocks that they correspond to. Special markers may also be introduced before very important data types, e.g. shape or motion markers, to assure that synchronization is present or recovered before this type of data is decoded. The markers to be used should be different from any possible combination of valid words in order to avoid unexpected emulations of the markers. However, if a greater resilience of the marker is needed, then a Hamming distance higher than 1 should be used. This will allow the markers to take some errors and still be recognized as a marker with the number of errors allowed depending on the Hamming distance between the marker and any possible combination of valid words.
- **Using error resilient entropy coding** – Entropy coding is a very powerful way to achieve high compression gains. However, since there are no fixed length codewords, an error in the bitstream may have a disastrous effect since, most of the times, synchronization will be lost. One possible way to solve this problem is by designing a more error resilient type of entropy coding. A technique that may improve the error resilience at the entropy coding stage is Reversible Variable Length Codes (RVLC), which have been first proposed in [Takishima95] and are currently used in the MPEG-4 standard [MPEG4]. These codes can be decoded, both from left to right and vice-versa. With RVLC, if an error occurs, all the data up to the next marker can be skipped and then decoded backwards. This helps to locate the errors as well as to minimize the amount of lost coded information and the negative impact due to the errors (ideally one single macroblock is affected in macroblock based coding systems).

While the first three techniques do not require the decoder to do any special processing, in the sense that the same decoding procedure used for the error free case would still work, the exploitation of the last technique requires the decoder to specially process (forward and backward) the received information, if benefit is to be taken from it. Moreover, while the first two techniques increase the bitstream error resilience capabilities by reducing the dependencies in time and space, the third and forth techniques have the main objectives of reducing the resynchronization time (and thus the amount of erred data), and improving the decoder error localization capabilities, respectively.

2.3.1.2 Channel Coding

Since the reduction of redundancy performed by the source encoder increases the video data error sensitivity, reintroducing some redundancy, in a controlled manner, may increase its robustness. One possible, and very common, way to do this is by using channel coding. If channel coding is done in a completely independent way from source coding, then it cannot take into account the different sensitivity of the coded data being protected, which is clearly a limitation. If the channel encoder “sees more than just bits” and “knows what is behind the bits,” the domain of joint source-channel coding is entered. The channel and source coding bit rate budget has to be carefully decided based on the type of bit rate resources (e.g., constant or variable), on its statistical characterization, on the channel error behavior and on the type of service.

While all channel coding techniques ask for bit rate resources that most service providers would like to use in a different way (and these resources may be substantial, as in mobile environments), they differ in terms of what they can do with existing errors. Depending on the amount of redundancy introduced, some techniques are designed to allow error detection, while others also offer error correction capabilities, which may be essential for some applications (notably real-time). If channel coding is to be used, the type of error handling (detection or correction) has to be decided, although the decoder may later decide to take more or less benefit of the associated data, for instance, depending on the computational resources available.

With forward error correcting (FEC) schemes, the encoder (always) adds redundant data to the bitstream. This technique allows the decoder to correct errors occurring in the bitstream, without asking for the re-transmission of any data. Although numerous FEC schemes exist, they can be classified in two main types:

- **Block codes** – When using block codes, the channel encoder accepts information in successive k -bit blocks and for each k bits generates a block with n bits, where $n \geq k$. For instance, Bose-Chaudhuri-Hocquenghem (BCH) block codes are recommended both in H.261 and H.263 standards [H.261][H.263].
- **Convolutional codes** – If a convolutional code is used, the encoder accepts the information as a continuous stream and generates a continuous stream of encoded bits at a higher rate. This type of codes is extensively used in mobile environments [Redl95]. In Recommendation H.223/A [H.223/A], “Multiplexing Protocol for Low Bitrate Mobile Multimedia Communication”, a Rate Compatible Punctured Convolutional (RCPC) code is recommended.

To further increase the performance of the error correction scheme, it is possible to concatenate several codes (i.e., apply them one after the other). Concatenation can be done by applying two or more codes of the same type (e.g., block codes), but it can also be done by applying codes of different types. The latter is the case in the Digital Video Broadcasting (DVB) standard, which uses a concatenation of a block code (i.e., Reed-Solomon code) followed by a convolutional code (i.e., RCPC code) [Reimers01].

In addition to these channel coding techniques, others can be used like interleaving and error framing. The idea of interleaving is to scramble the bitstream, guaranteeing that the successive bits transmitted are widely separated in the data to be decoded. In practice, this means that burst errors in the bitstream will be transformed into more evenly distributed errors in the data to be decoded and, thus, more easily dealt with by the channel decoders. This technique, used in conjunction with channel coding, is very powerful and widely used in

2. Error Resilient Video Coding: Organizing and Classifying

mobile environments such as GSM [Redl95]; more recently, this approach has also been adopted by the DVB standard [Reimers01]. However, if channel decoding is not used, interleaving would be prejudicial, since uniformly distributed errors typically lead to a more negative subjective impact than burst errors, for the same average bit error rate (BER), in the case of variable length coding (VLC) [Talluri97]. This happens because, instead of having one packet with many errors, a lot of packets will have few errors, which may be disastrous when variable length coding is used.

As for the framing technique, a framing bit is inserted in each frame or packet with the constraint of giving a specific pattern to the framing bits in a sequence of packets. If, at the receiver, the pattern is not the one expected, something must have gone wrong, such as the occurrence of an error in the framing bits, or the loss of synchronization. Framing can be very useful to check that packet synchronization is not lost even if word synchronization is lost. Both the H.261 and the H.263 standards use a fixed-length framing technique [H.261] [H.263].

In terms of this chapter, channel coding provides two capabilities, error detection and error correction. While error correction may give an end to the problems since the correct data is recovered, error detection just provides very useful information for the next stages: error localization and concealment.

Forward error correcting codes have been widely used in all types of communications and are well known for their error detection and correction capabilities. However, since their use increases the necessary signal bandwidth, they have to be used sparingly.

2.3.1.3 Joint Source and Channel Coding

Joint source-channel coding corresponds to a more global approach to coding. Instead of considering that source coding and channel coding are done separately and independently, source and channel coding are done together. This way the channel encoder will have access to privileged information that was previously only available to the source encoder and vice-versa, thus allowing the encoder to globally exploit the characteristics of the video data, which would otherwise be impossible.

In terms of joint source-channel coding, two different approaches are possible, depending on the type of information from the source encoder that the channel encoder is allowed to use:

- **High-level joint source-channel coding** – In this type of joint source-channel coding, the channel encoder is allowed to take into account the different sensitivity of the bits generated by the source encoder and add more or less protection accordingly, as is typically done when unequal error protection schemes are used (e.g., the header data has more error protection than motion data, which has more error protection than texture data [Heinzelman99][Budagavi00]). If object-based video representation schemes are targeted, the various objects in the scene will be represented in a fully independent way, and may have different (semantic) priorities. Thus, a hierarchy based on object priorities can be established in a given scene, leading, if desired, to different degrees of channel coding protection.¹

¹ In MPEG-4, audiovisual (AV) object data is conveyed in one or more Elementary Streams (more than one, if scalability is used). The streams are characterized by the Quality of Service (QoS) they request for transmission, as well as other parameters, including stream type information and the precision for encoding timing information. The FlexMux multiplex sub-layer allows grouping together Elementary Streams with similar QoS requirements [MPEG4-S].

- **Low-level joint source-channel coding** – In this type of joint source-channel coding, the approach is different and it basically involves the joint design of the quantizer, the entropy encoder and the error correcting codes. By designing these stages at the same time, while taking into account the error characteristics of a given channel, the effect of transmission errors can be minimized. This approach was first proposed for image coding in [Modestino79][Modestino81], but many others have also used it for video. For instance, in [Ramchandran93], the use of low-level joint source-channel coding is proposed for the broadcast of digital high-definition television (HDTV).

As could be seen, even though both approaches belong to the category of joint source-channel coding, the ideas behind them are rather different. On one hand, low-level joint source-channel coding schemes are based on the principle that, by jointly designing the quantizer, the entropy encoder and the error correcting codes, it is possible to (optimally) minimize the distortion at the decoder for given channel error characteristics. However, this can only be applied to data types, which are sequentially quantized, entropy coded and channel coded, which is usually the case for texture data (e.g., H.263 [H.263], MPEG-2 [MPEG2]). However, in typical video coding systems, other types of data exist, which may not be subject to quantizing or entropy coding. For instance, in most existing video coding standards, while header data is typically not quantized or even entropy coded, motion data is typically not quantized. Additionally, in the only object-based video coding standard (i.e., MPEG-4 [MPEG4]), shape data is also not quantized. On the other hand, in high-level joint source-channel coding schemes, the channel encoder has the advantage of having a global understanding of all the different types of data involved and their different sensitivities to errors, as well as the different sensitivity to errors of the various bits describing the same type of data. Therefore, it can apply different amounts of channel coding to the different types of data (motion, texture and shape, if object-based coding is considered) according to their sensitivity, as well as to the different bits that describe the same type of data (e.g., low-frequency transform coefficients versus high-frequency transform coefficients). The disadvantage of the high-level joint source-channel coding scheme is that the source encoder does not adapt to any information sent from the channel encoder (e.g., when coding the texture, the quantizer and the entropy encoder are designed separately from the channel encoder), thus leading to a sub-optimal solution. However, since the low-level and high-level approaches are not really alternatives, it should be possible to combine their advantages by using them both.

2.3.2 Back-Channeling Systems

Back-channeling systems are characterized by the fact that a back-channel from the decoder is available and, therefore, the encoder can receive feedback from the decoder “asking for help”. Based on this information, the encoder can take the most appropriate encoding action in order to help the decoder to minimize the negative impact of the errors. This could be called *corrective resilience*, as opposed to the preventive resilience of the forward error resilient systems described above.

Since the encoder is clearly divided in two major encoding modules, the feedback provided by the decoder can be used to take encoding action either at the source coding level or at the channel coding level. In addition, if joint source-channel coding is used at the encoder, the feedback provided by the decoder can also be used to take appropriate action in terms of joint source-channel coding. These approaches are described below in the corresponding sections.

2.3.2.1 Source Coding

The decoder can provide feedback to the source coding part of the encoder, which will take the necessary actions in terms of source coding techniques to help the decoder. This way, when one or more errors occur, the decoder signals the encoder, using the back-channel, that one or more errors were detected and, possibly, where. Based on this information, the encoder may have one of three reactions:

- **Resend corrupted information** – When this strategy is used, if an error is detected at the decoder, the encoder will be immediately signaled and asked to resend the corrupted information. This approach has the advantage that errors will effectively be corrected (even if more than one retransmission may be needed). However, if the system has severe delay constraints, such as for real-time services, this technique might be unusable. Additionally, the remaining available bandwidth can suffer a considerable reduction if the number of retransmissions is large.
- **Send additional side information** – With this strategy, when an error occurs, the decoder signals the encoder. Upon this, the encoder will send additional side information that will help the decoder to conceal the corrupted data. For instance, when the encoder is signaled for help, it can send additional motion vectors to allow the decoder to have a good prediction for the corrupted texture data and perform temporal concealment. The advantage this strategy has over the previous one is that the side information is typically much smaller than the original (repeated) information that is supposed to help the decoder. Therefore, the reduction in the remaining available bandwidth will be much smaller. Of course, if the roundtrip delay is very large, this approach might also not be usable for real-time systems.
- **Selective coding** – When selective encoding is used, the decoder uses the back-channel to signal the encoder of what errors occurred and, possibly, where, as for the previous two strategies. Based on this information, the encoder will proceed with the encoding but without ever using corrupted parts of the image for prediction. This will effectively stop error propagation at the decoder, which in fact is the big problem that errors may cause. After all, if an error persists for only one or two frames, no much harm will be done because the HVS will be able to effectively filter that effect. The big problems happen when errors persist during a rather long time.

2.3.2.2 Channel Coding

At the channel coding level, when a back-channel exists to convey feedback from the decoder, automatic repeat request (ARQ) schemes can be used. These schemes are based on the introduction of redundancy in the bitstream with the purpose of allowing the decoder to detect errors. When an error occurs, the decoder signals the encoder that an error was detected and, possibly, where; for this, the back-channel is used. The encoder's reaction differs, depending on the type of ARQ technique used, notably:

- **ARQ-type I** – The encoder resends to the decoder the erroneously received packet. This scheme can guarantee a very good quality but the real-time requirements are difficult to fulfill and, therefore, it is typically not usable in real-time communications, if the round-trip delay is very large.
- **ARQ-type II** – The encoder sends to the decoder some extra (redundant) data that should allow the decoder to recover from the errors. The size of the redundant data to be transmitted has to be carefully designed, taking into account the desired

capabilities in terms of error correction. As an example, suppose that a very simple code, such as a parity-check code, is used. After each k information bits, the encoder adds a set of r parity-check bits, which is derived from the k information bits. The set of r bits is chosen such that only error detection can be performed at the decoder. If an error is detected, the encoder is signaled and sends a larger set of parity-check bits that should now allow the decoder to correct the errors.

2.3.2.3 Joint Source-Channel Coding

At this level, the feedback received from the decoder, through the back-channel, can be used by the encoder to decide on any necessary encoding action in terms of joint source-channel coding to help the decoder. As in Section 2.3.1.3, high-level and low-level approaches can be used for the joint source-channel coding. For instance, if the high-level approach is followed, the encoder can upon being signaled for help, resend the corrupted information. However, since this data is being resent, the encoder should also readjust the amount of channel coding protection that is used, which should probably be higher than the first time it was sent. If on the other hand, the low-level approach is used, a possible strategy is to use the feedback information to estimate changes in the channel conditions. With the updated channel conditions, the encoder will be able to reconfigure the joint source-channel coding parameters (i.e., of the quantizer, entropy encoder and channel encoder).

2.4 Error Resilient Decoding and Concealment

Error concealment includes all the techniques that may allow the receiving terminal to minimize the negative subjective impact of errors by using the available corrupted and correctly decoded data. When content-based coding is used, content-based concealment is applied in the sense that each object (corresponding to an independent elementary bitstream) is error concealed using only its own data; scene level concealment may also have to be applied to reduce the error effects, for instance, in the way the various concealed arbitrarily shaped objects fit together. Since concealment techniques involve the “creative playing” with the corrupted and correct decoded data, very much depending on the specific content and application, they are not subject to standardization and remain as one of the areas, within the standards, for industry competition. This means all video coding standards define its normative decoding process in error free conditions even if they include coding techniques with the purpose to increase the resilience of the bitstreams.

In order to have efficient error concealment, the decoder usually goes through three different processing steps. First of all, the decoder has to know that an error occurred. Then it has to know (in the most precise possible way) where it happened and, finally, it will try to dissimulate – *conceal* – it. In conclusion, error concealment in a broad sense can be divided into the following three stages, detailed below:

- **Error detection techniques** – All the techniques that allow the decoder to know that one or more errors occurred.
- **Error localization techniques** – With these techniques, the decoder tries to find, with the best possible precision, where the detected error(s) occurred.
- **Error concealment techniques** – Finally, by using different dissimulation techniques, the decoder tries to decrease the negative subjective impact of the errors that could not be corrected (by channel decoding).

2.4.1 Error Detection

The first step in error resilient decoding is the detection of the errors. While the detection of errors may ease its concealment, non-detected errors may also be concealed, e.g. by post-processing of the decoded images after the error concealment techniques have been applied. Since most of the information is encoded with *Variable Length Coding* (VLC), usually either Huffman or arithmetic encoding, when an error occurs, synchronization is, almost always, lost and errors can propagate for a long time. Therefore, it is important to detect errors as soon as possible. The error detection techniques can be grouped according to three types, but typically many techniques shall be combined to improve the error detection capabilities.

2.4.1.1 Channel Decoding

This includes the channel coding techniques with error detection capabilities. In terms of concealment, only the errors detected but not corrected are relevant. For example, channel coding can be applied to packets of a certain length, allowing the decoder to know which packets have errors. The first localization of errors is usually poor in precision. Some examples of channel coding schemes have been given in Section 2.3.1.2 dedicated to this topic.

2.4.1.2 Syntactic Inconsistency

Here, the detection of errors results from the occurrence of syntactic inconsistencies in the source coded bitstream. For Huffman coding, the simplest syntactic inconsistency is the appearance of an “impossible” codeword; this means that, in the decoding process, a codeword that cannot be the beginning of any valid codeword occurs. This will only happen if incomplete VLC tables are used, making some codewords illegal at the cost of some compression efficiency. The more incomplete the table is, the better the error detection capability and the worse the coding efficiency will be. Another case of syntactic inconsistency is the unexpected appearance of a marker in the middle of data, meaning either that an error occurred in the data and the decoder continued up to the marker without detecting it, or that an error occurred and a marker was emulated. A similar syntactic inconsistency is the absence of a marker where expected (e.g. at the end of a block of data).

2.4.1.3 Semantic Inconsistency

The detection of errors results from the occurrence of semantic inconsistencies in the source coded bitstream. Examples of semantic inconsistencies are the decoding of DCT coefficient with order higher than 64 in an 8×8 pixels DCT context, or the detection that the current running macroblock count is different from the one indicated after a resynchronization marker.

2.4.2 Error Localization

The best possible localization of errors is essential for their concealment since the better the error localization, the smaller the amount of correct data discarded. Unfortunately most of the error detection techniques are very imprecise in terms of error localization: while channel decoding usually gives an indication of error for a block of data, syntactic and semantic inconsistency detection gives an indication of error for a bit or a codeword but typically much

after the error effectively happened (depending on the degree of completeness of the code). This is why, in order to reach the best error localization results, based on an error detection indication, several error localization techniques are often used simultaneously. In some other cases, a very simple solution is adopted, just assuming that the error happened N bits or macroblocks before it was detected, where N is determined based on extensive experimentation for the relevant conditions.

2.4.2.1 Error Resilient Entropy Coding

Error localization can also be performed by exploiting the error resilient entropy coding. For instance, RVLC codes can be used to further localize the errors, as illustrated in Figure 2.2. If RVLC codes are being used, and an error occurs, all the data up to the next marker is skipped and then a backward decoding process is started. This helps to better locate the occurrence of the error(s), minimizing the information discarded (ideally one single macroblock in macroblock-based schemes), and thus the negative impact of the errors.

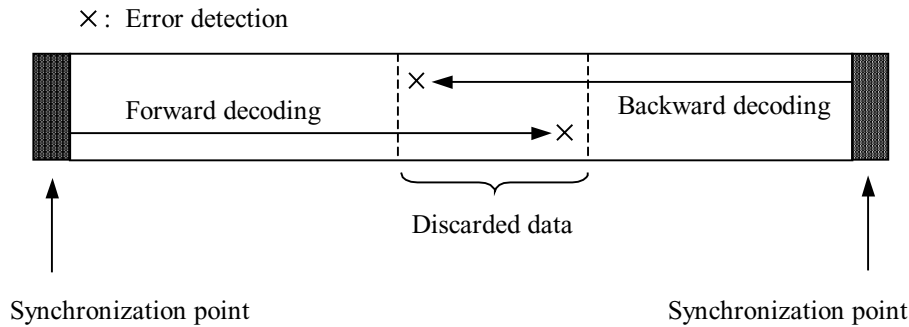


Figure 2.2 – Error localization capability of RVLC codes

2.4.2.2 Decoded Parameters Analysis

When decoding all the video parameters, the decoder should be kept alert in order to detect strange parameter values. Typically, very sudden and large changes in parameters may mean that an error has occurred. Thus, by determining where these changes occurred in the parameter values, the decoder will usually be able to improve the error localization.

2.4.2.3 Image/Video Analysis

Since errors are many times detected much after they occur, some corrupted texture data is (correctly) decoded, leading sometimes to very strange and visible image artifacts, such as light green 8×8 blocks aligned with the 8×8 DCT block grid in a dark background. These artifacts and the associated errors may be detected, improving error localization, by means of post-processing techniques. For example, for DCT-based coding schemes, it is possible to improve the localization of a detected error by neighborhood continuity processing. This means looking for strong image discontinuities aligned with the DCT coding grid, starting by the last decoded image area (e.g., a macroblock), and supposing that it is highly unlikely that strong (usually very colorful) image edges coincide with the coding grid. The blocks fulfilling certain strong discontinuity criteria will be considered as corrupted, and the position

2. Error Resilient Video Coding: Organizing and Classifying

of the error correspondingly moved backwards (in terms of the decoding direction). Another way of detecting errors by texture post-processing is by accepting some constraints, e.g. in terms of color saturation. For instance, in a video telephone application, it is reasonable to assume that 8×8 blocks with highly saturated colors (e.g., of pink) are unlikely, and hence can also be considered erroneous. However criteria that might be reasonable for one application may not be for another. For example, 8×8 blocks with saturated colors might exist in an application including synthetic video content. These post-processing techniques can be switched off if desired since they always involved a (very low) probability of detecting as erred, correct data.

When object-based coding schemes are used, some corrupted shape data may also be decoded and lead, in principle, to evident shape artifacts, as it happens for texture. For shape, it is difficult to define in a simple and generic way what is an artifact, but this may be possible in the context of specific applications. If, for example, it is known that shapes are smooth without sharp edges, shape errors may be detected if this condition is not verified.

2.4.3 Error Concealment

After the errors are detected and as much as possible precisely localized, it is time to start concealment, which basically means to hide their negative subjective effects. The decoder tries to minimize the negative impact of the errors in the decoded images just by using the available decoded data (motion and texture or also shape) or in some conditions additional data may be requested from the encoder. The concealment process is always based on some sensible assumptions, which may depend on the type of application and content involved.

Depending on the data that is used, the decoder error concealment techniques can be classified into the following three categories:

- **Spatial error concealment** – In this type of concealment, only the data from the current time instant is used to perform the concealment operation.
- **Temporal error concealment** – In this case, the data from other time instants is used to perform the concealment. The most common approach is to use the data from the previous time instant.
- **Spatio-temporal error concealment** – This approach is basically a combination of the previous two, in the sense that data from both the current time instant and other time instants is used. This is typically the most powerful approach, but also the most complex.

These three different approaches will be further detailed in the following sections.

2.4.3.1 Spatial Error Concealment

As mentioned above, in spatial error concealment techniques, only the data from the current time instant is used to perform the concealment. This basically means that corrupted areas in a given image are recovered by interpolating the data from the surrounding correctly decoded areas of the same image. As it is easy to understand, this approach can have serious problems if the corrupted area of the image that is being concealed represents quite a large part of the whole image, especially if it is very inhomogeneous. This is the reason why these techniques are mostly used for large images, where the corrupted areas are typically relatively small when compared to the size of the whole image. In particular, this technique works relatively

well for homogenous areas. In addition, these techniques are also suggested for sequences where there is very little temporal redundancy.

2.4.3.2 Temporal Error Concealment

As the name suggests, in temporal concealment techniques, data from time instants other than the time instant being considered is used to perform the concealment. Although the most common approach is to use data from the immediately preceding time instant, this does not have to be so and other time instants may be used. When this technique is applied to sequences where a large temporal redundancy exists, which corresponds to most of the cases, the results will typically be very good. Otherwise, serious problems can appear if the decoder tries to conceal the image in the current time instant with data from images in surrounding time instants that have very little to do with it, for example in a scene cut.

2.4.3.3 Spatio-Temporal Error Concealment

The idea behind the spatio-temporal concealment techniques is to have the best of both worlds by combining the spatial and the temporal concealment techniques. This way, some parts of the image might be concealed using spatial concealment, others by using temporal concealment and still others by using a little bit of both. Ideally, each of these solutions should be used for the parts of the image that are spatially, temporally or both spatially and temporally very homogeneous, respectively. For this, adequate image analysis techniques have to be used.

2.4.4 Post-Processing Techniques

Post-processing techniques are very commonly used tools for artifact reduction on the decoded images. For the purpose of this Thesis, these tools, which may also be called filters, can be classified in two classes depending on the type of artifact reduction that has to be performed:

- **Filters for coding artifact reduction** – These techniques try to remove the coding artifacts in the decoded (after concealment) images, such as the DCT blocking artifacts. Good examples of such filters are the deblocking and deringing filters, specified in Annex F of the MPEG-4 Visual standard [MPEG4-V]. In Figure 2.3, an example is shown where the improvement achieved with a deblocking filter is very clear.
- **Filters for error artifact reduction** – These techniques try to remove the image artifacts resulting from undetected (and thus unconcealed) channel errors by using some sensible assumptions about the image content. These filters may be useful since the error concealment techniques described above are usually applied when errors are detected, but unfortunately it also happens that errors are not detected at all by any of the detection techniques mentioned. The probability of an error not being detected mainly depends on the bitstream syntax and on the degree of completeness of the Huffman code used. Anyway, either because some errors have not been detected or were detected but not adequately localized and concealed, it is possible that some artifacts still remain in the decoded images after all the concealment stages above described are applied.



Figure 2.3 – Images before (a) and after a deblocking filter has been applied (b) [Schafer03]

In the context of this Thesis, only the post-processing techniques for error artifacts reduction will be studied, since the other type of post-processing, although very important for the final subjective impact, is not exclusively related to channel errors (and it is in fact also used in error free environments). The post-processing filters should detect error artifacts, defined as unlikely decoded data in the context of the relevant application. However, since there is always a residual probability that the detected “unlikely decoded data” is good data and not an artifact, *the use of these techniques is optional and they may be switched off at the receiver*. After an artifact is detected, one of the error concealment techniques already described is used.

As examples, two possible texture error artifacts filters will be presented in the following:

- **Pixel value post-processing** – This technique makes some assumptions about the values that texture can take, for instance, within a macroblock. For example, macroblocks with a significant number of light green or bright pink pixels can be considered artifacts in the context of a video telephone conversation. Also macroblocks for which a high number of pixels are subject to clipping, this means the decoded values are higher than 255, are very likely a sign of an artifact. In Figure 2.4, macroblocks with saturated pink are removed, which may be a wrong decision in the context of applications with synthetic content. Since most error detection post-processing filters are content-dependent in the sense that they may lead to wrong decisions if the content does not follow the basic assumptions behind the error detection post-processing filter used, the choice of the filters to use must be carefully made, depending on the relevant application (and thus on the corresponding typical content).
- **Continuity post-processing** – This technique assumes that a good continuity exists at the boundaries of blocks, in the context of 8×8 DCT coding schemes. A filter able to detect high discontinuities along 8×8 blocks aligned with the DCT coding grid can very likely signal error artifacts (depending on the discontinuity thresholds set). In Figure 2.5, the discontinuities at the border of the various macroblocks are processed to detect “out of context” macroblocks, according to a certain threshold. This

technique may lead to wrong decisions if the video material contains discontinuities aligned with the DCT coding grid.



Figure 2.4 – Images before (a) and after pixel value post-processing (b)

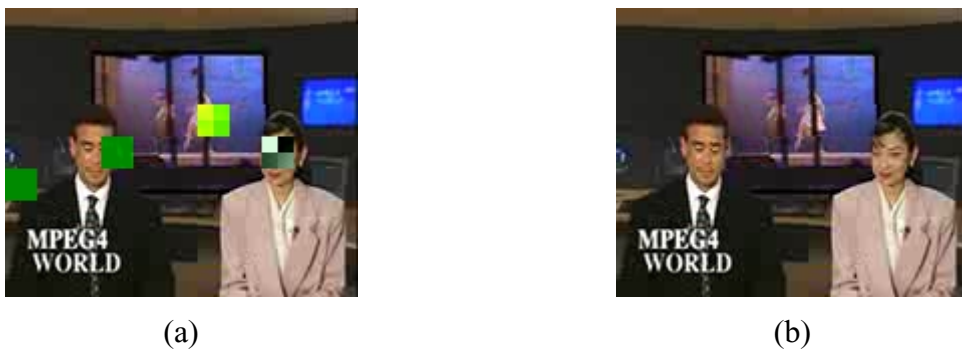


Figure 2.5 – Images before (a) and after continuity post-processing (b)

The post-processing stage to remove error artifacts may significantly improve the video subjective impact since very evident artifacts can be removed from the images. However, this impact depends on the coding scheme used and on its resilience to errors, e.g. a very resilient coding scheme can avoid the strong artifacts that post-processing is supposed to remove. Other types of post-processing concealment besides those described here may also be performed, both for texture and shape. These techniques are not subject to standardization but may lead to significant quality differences between terminals using the same video coding standard, notably for critical error conditions.

Similar post-processing techniques may be applied to shape information although the definition of shape artifacts is more difficult and may only make sense in the context of specific applications with clear limitations on the shape characteristics.

2.5 Error Resilience in Object-based Video Coding Systems

In the previous sections, a general organization of the video coding error resilience problem has been given. However, emerging object-based video coding systems have specific error resilience problems that have to be dealt with, which will be addressed in this section. But, before trying to cope with these problems, it is important to understand why the development of error resilience tools for object-based video coding systems is so important. The answer to this question is straightforward and lies in the fact that an increasing diversity of networks,

2. Error Resilient Video Coding: Organizing and Classifying

some of them with critical bandwidth and channel error characteristics, such as mobile networks and the Internet, are being used for video communications. Therefore, if object-based video is to be transmitted over such networks, error resilience tools will have to be developed.

As it will be seen in Chapter 3, these error resilience issues have been exhaustively discussed for frame-based video, but they are a very new topic when it comes to object-based video. However, the general problem of error resilience that has to be solved is still, in part, the same as with frame-based systems: showing the user the best possible video quality with the available network resources, while taking into account the channel error characteristics of the used network. Additionally, in object-based systems, it is also important to make sure that some of the object-based functionalities are not compromised due to channel errors.

Another issue that needs to be addressed is the performance evaluation of object-based video coding systems in the presence of errors. In order to specify new error resilience tools and compare them, adequate quality evaluation methods are needed, since the usual ones based solely on the texture of frames cannot be used in this context.

This way, in order to better understand the error resilience problems associated with object-based video coding systems, it is important to first understand how object-based systems work. For that, the architecture of such a video coding system will be presented in detail in the following section.

2.5.1 Error Resilient Object-based Video Coding Architecture

Object-based video coding systems represent the latest development in video coding technology, and follow the currently available frame-based video coding systems. In these new systems, a new data representation model is adopted. While in frame-based systems video data was represented as a sequence of rectangular matrices of picture elements (*pixels*), in object-based systems, video data is modeled as a composition of (semantically) relevant 2D arbitrarily shaped video objects. These objects can have one or more non-connected parts, which are typically referred to as regions. Therefore, in order to represent each video object, the traditional motion and texture data is no longer enough and a new video data type has to be added — *the shape*. This way, the complete data set required to represent a video object consists of shape, motion and texture data. In addition, some composition data is necessary to create a video scene where a few objects may co-exist [Pereira02].

This change in the video representation model has a significant impact in terms of error resilience, since new solutions will have to be found to deal with transmission or storage errors in the bitstreams. For instance, in frame-based systems, the problem of error concealment always corresponded to the concealment of rectangular objects, since the shape of the object is *a priori* known to be a rectangle with fixed dimensions. In object-based systems, however, arbitrary shape is added, as well as scene composition information, which brings an additional dimension to the problem of video error concealment. This means that, in these new systems, when errors occur, the shape data may have to be concealed as well as the data that describes the scene composition (i.e., scene composition concealment). Therefore, error resilience in frame-based systems can be seen as a subset of the general problem of object-based video error resilience in the sense that most techniques available may still be useful (even if some adaptation may be required) but new techniques, notably for shape and scene composition concealment, have to be developed. Frame-based resilience techniques have been extensively studied and much work has been published; a good review of this work can be found in [Wang98]. However, very few papers are known dealing with object-based

video error resilience and, therefore, much work still remains to be done to solve the general problem of object-based video coding error resilience.

To better understand this problem, the general architecture of an error resilient object-based video coding system should be considered. This system, which includes many different modules responsible for very specific tasks, is illustrated in Figure 2.6.

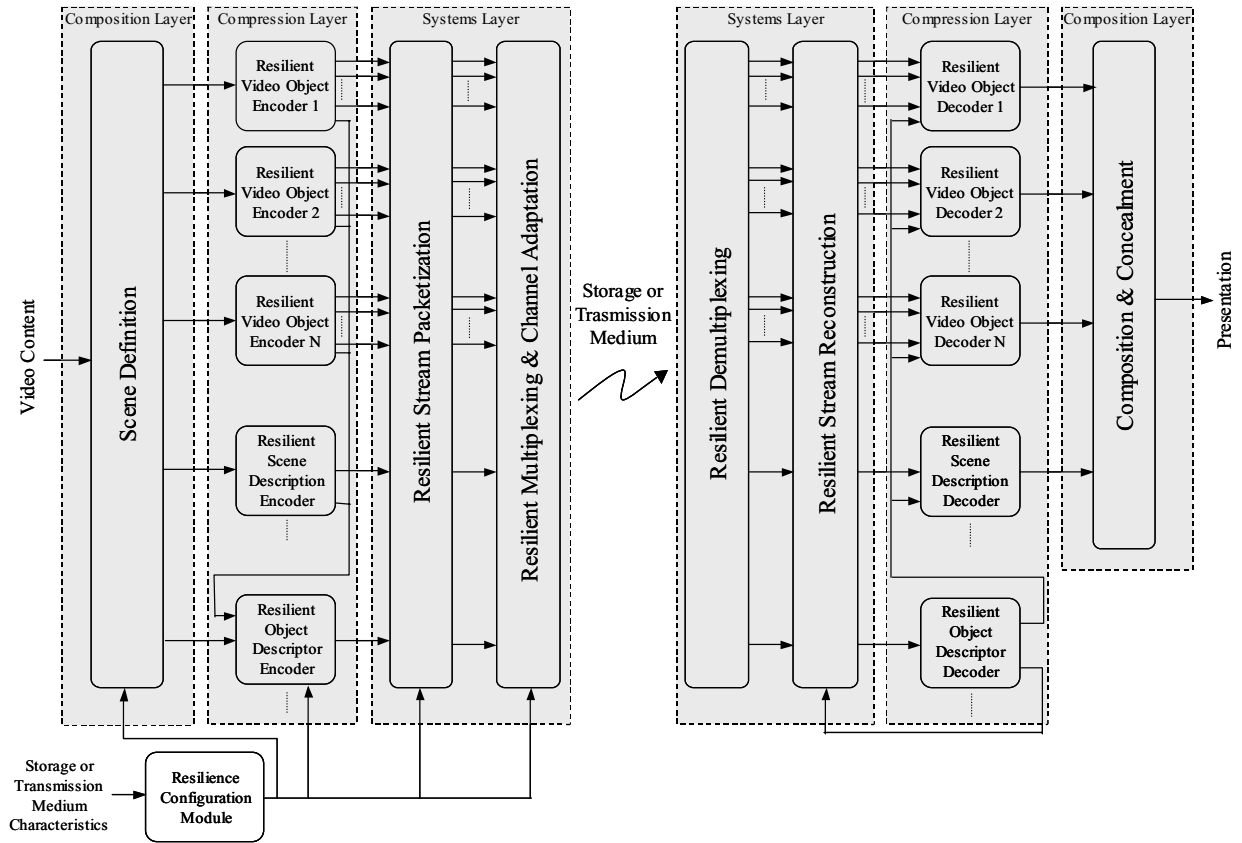


Figure 2.6 – Example of an error resilient object-based video coding architecture

On the encoder side, the first step is to take the input video content and define a video scene with it. This can be done by segmenting an existing video sequence, by composing it with already existing pre-segmented video objects or a mixture of both. After the video scene to be coded has been defined, the elements that compose the scene are sent to the corresponding encoders. This means that each video object is sent to a video object encoder (corresponding to a certain video object type), and the information that positions them in space and time is sent to the scene description encoder. In addition to this, the object descriptor data, which creates the link between the various elementary streams and the objects in the scene description structure, is sent to the object descriptor encoder. The output of all these encoders corresponds to various elementary streams containing encoded data, which will then be packetized, multiplexed and, finally, adapted to the storage or transmission channel(s) in question. All the previously described operations, are supervised by the *resilience configuration module*, which is basically responsible for choosing the most adequate (in terms of resilience) coding parameters. This module is necessary because the encoding side of the communication chain plays an important part in error resilience; in fact, the decoding performance will depend a lot on the kind of “protective action” the encoder has taken. As for the scene definition module, it typically plays no part in error resilience since its role is to

2. Error Resilient Video Coding: Organizing and Classifying

define the scene, possibly based only on semantic criteria. However, it can also help. For instance, by slightly smoothing the video objects, concealment may become easier at the decoder.

At the decoding side, the procedure is basically the opposite but, instead of having a scene definition module, a scene composition module is used. This way, the decoder starts by demultiplexing the received bitstream(s) from the storage or transmission channel(s) and then reconstructing the various elementary streams, which are sent to the corresponding decoders. After all the elementary streams have been decoded and the various scene elements recovered, they are sent to the composition module (except for the object descriptor data). The function of this module is to take all the decoded video objects and the information that relates them in space and time and build a scene that can be presented to the end-user. At the decoder, while performing all the described tasks, “defensive actions” have to be taken in order to minimize the negative subjective impact of channel errors in the presentation.

As was seen in the description above, the general problem of error resilience is a distributed one, in the sense that each module in the object-based video coding system plays its part in the error resilience of the whole system. This basically means that the problem of error resilience cannot be solved at only one specific location in the system and, therefore, to increase the error resilience of the whole system, the error resilience of individual modules should be improved. However, since some of the modules have conceptually similar functions and problems that need solving (at the encoder and the decoder), they can be grouped together into layers, thus helping to better structure the problem. The following three layers, which are highlighted in Figure 2.6, have been identified:

- **Systems layer** – This layer is responsible for sending/receiving the various elementary streams. The tasks include packetization, multiplexing, adding time information and channel adaptation, at the encoder side, and the inverse operations at the decoder side.
- **Compression layer** – This layer is responsible for encoding/decoding all the necessary elements that make up the scene, according to their type (e.g., video objects, scene description data). It considers as elementary streams the video objects, the scene composition data and the object descriptor data.
- **Composition layer** – This layer is responsible for defining/composing the video scene. At the encoder, it receives the video content and it outputs the various video objects that make up the scene, in addition to the scene description information that relates them in time and space. At the decoder side, it receives the various video objects, as well as the scene description information, and builds a scene to be presented to the end-user.

In the following sections, the error resilience problems associated with the various modules included in these three layers will be discussed. But first, the problem of object-based quality assessment will be discussed.

2.5.2 Object-based Video Quality Metrics

At this point, it is important to mention that in order to develop adequate error resilience schemes, reliable quality assessment techniques are needed for comparison purposes. After all, it is necessary to compare the various concealed versions (the images obtained by concealment with different techniques) of the same corrupted video material to decide which one is closer to the original and, in principle, the one with the best subjective impact. For this,

as explained back in Section 2.2.1, two types of quality assessment techniques can be used: subjective or objective. However, since the subjective techniques are so time-consuming and expensive, only objective quality assessment will be considered here. Thus, the fidelity of the concealed video with respect to the original has to be measured by an objective metric related to a certain quality criterion. The quality criteria have to be chosen such that the corresponding metrics measure the subjective impact of the video on the human visual system (HVS) in a fairly reliable way.

For frame-based video systems, the problem is basically reduced to finding a way of evaluating how subjectively close is the texture of a given video frame to the corresponding original. Many objective metrics have been developed that can more or less reflect the impact of impaired rectangular textures on the HVS. Some of these metrics can be fully described by a computable analytical model, which makes them very easy to use. This is the case of the peak signal-to-noise ratio² (PSNR), which is certainly the most used metric for the evaluation of frame-based video quality.

In object-based video systems, however, the problem is more complex, because an additional quality dimension exists. The quality can thus be evaluated at two different levels:

- **Object level metrics** – These metrics measure the quality of an object as a stand-alone entity that has no relationship with other surrounding objects. This means that these metrics, when assessing the quality of a given object, will not take into account how this object fits in the scene to which it belongs and how it relates to the surrounding objects; these quality metrics regard the fidelity of the decoded texture and shape for that single object.
- **Scene level metrics** – These metrics measure the quality of a scene as a whole, considering the various objects that make up the video scene and their composition. This means that these metrics have to take into account, not only the quality of individual objects, but also the quality and type of the “relationships” between them, such as how they fit together and their relative importance in the scene.

These two types of metrics have to deal with both shape and texture, instead of just texture as in the old frame-based systems, because the subjective impact of an object or a scene depends, not only on the texture quality, but also on the quality of the shape of the various objects. Additionally, in the case of scenes, the subjective impact also depends on how the various objects fit together in the scene. Therefore, for object-based systems, metrics have to take shape and texture quality into account when assessing the quality of an object, and also the relationship between objects in the case of a scene. In this context, the following three elementary types of metrics can be defined:

- **Object shape metrics** – These metrics measure the fidelity of the shape for a single object, and ideally reflect its subjective impact on the user. The fidelity of the shape is very important, not only because of the subjective impact of the shape itself, but also because the shape of one object affects the way it fits with the surrounding ones (for video segmented objects). Shape fidelity is very critical for easily recognizable shapes, such as squares, circles, letters, etc. and when objects are to be reused in other scenes.
- **Object texture metrics** – These metrics measure the fidelity of the texture for a single object, and ideally reflect its subjective impact on the user. The fidelity of texture seems to be less critical than the fidelity of shape, because the texture inside a given object does not influence as much as the shape the surrounding objects.

² $PSNR = 10 \log_{10} (255^2 / MSE)$, where MSE is the mean square error.

2. Error Resilient Video Coding: Organizing and Classifying

- **Relational metrics** – These metrics are used for video scenes and measure how well the different objects build the scene. For instance, this is associated to aspects such as the fitting of the shapes of neighboring objects (if this is relevant), the relative texture and shape quality of the objects in the scene, their capability of calling the user attention, and their position in the scene.

When assessing the quality of a single, stand-alone object, only object shape and texture metrics should be used. This means that object metrics will basically be a function of shape and texture metrics. On the other hand, when assessing the quality of a scene, shape, texture and relational metrics should be used and, therefore, scene metrics will become a function of the three mentioned metrics. How these metrics are defined is a rather complex problem on its own and will be dealt with later in the Thesis (see Chapter 5).

2.5.3 Error Resilient Object-based Encoding

Similarly to what was explained in Section 2.3, the best place to start thinking about error resilience in object-based video coding systems is at the encoder. At the encoder, it is possible to introduce error resilience at the compression layer, at the systems layer and at the composition layer. However, the contribution of the composition layer to the error resilience of the whole system is rather limited and, therefore, it will not be considered here. Thus, at the encoder side, only the compression and systems layers will be considered and the most important error resilience aspects in each one of these layers discussed.

2.5.3.1 Compression Layer Resilience

As was seen in Section 2.5.1, the compression layer includes three types of elementary encoders: the video object encoder, the scene description encoder and the object descriptor encoder. The error resilience issues that each one of these encoders has to deal with will be considered in the following subsections.

2.5.3.1.1 Video Object Encoder Resilience

In frame-based video coding systems, the following different types of data were needed to describe one rectangular video object—*the frame*: header data, motion data and texture data. In object-based systems, however, since the scene is divided into various (semantically) relevant arbitrarily shaped objects, this information alone is not enough to describe the whole scene. A new type of video coded data is needed: *the shape*. Therefore, to represent a video object in object-based systems the following types of data are needed:

- **Header data** – Header data is by far the most important information for one object. Without it, correctly decoding of the video object is not possible. This type of data is also the hardest to conceal, because of its very low redundancy. This information includes fields such as the temporal references of objects, which already existed in frame-based systems, and new fields such as the position of an object within a video scene or the size of the object. Since in object-based systems, the video objects can change position and size over time, it is important to make sure that this information is correctly decoded, otherwise very strange object behaviors can happen at the decoding side.
- **Shape data** – Shape data is the information used to describe the arbitrary shape of video objects and did not exist for frame-based systems. After the header data, it is

probably the most important of the various types of data needed to represent a video object because its loss can create artifacts that can severely degrade the subjective quality of the whole object.

- **Motion data** – Next to shape data, motion data is the most important information, since its loss can be very difficult to overcome, especially in fast moving objects, and if erroneously decoded, very strange artifacts may be created.
- **Texture data** – Texture is the least important information and its concealment is usually easier than for the other types of data. Usually, if the shape and motion data are known, a good prediction can be obtained from the texture in the previous time instants, thus allowing the decoder to perform some type of concealment. However, some other concealment schemes are also possible.

By adding the shape data, new problems related to the loss of this new type of data will appear, which did not exist in frame-based systems. Therefore, it is expected that the field of shape data error resilience is where most innovation can be achieved. In addition, error resilience techniques related to the other types of data (i.e., motion and texture) may also need revision since motion and texture coding have to be slightly changed with respect to the corresponding solutions used in previous frame-based video coding systems. Here, however, not as much innovation is expected as for the shape data.

2.5.3.1.2 Scene Description Information

The scene description information is what allows the various video objects to be organized in a scene, in terms of spatial and temporal positioning. This description can be modified by performing scene description updates [MPEG4-S], which correspond to a stream that also has to go through a transmission or storage medium. This means that they are subject to possible transmission or storage errors, which have to be accounted for. The scene description updates stream carries the information needed to modify a set of properties of the scene at a given time. The scene update information is used to build the scene and the implications of errors in this information are quite clear. On one hand, undetected errors can cause very strange effects in the scene composition, since objects can be misplaced or simply removed from the scene. On the other hand, detected errors can also be quite harmful in another way, since the strange effects due to erroneous interpreted information will be avoided but the lost information will be very hard to conceal, if not impossible. Therefore this information should be very well protected. It can be either transported through a channel with virtually no errors or schemes should be designed to allow it to be easily recovered. In this Thesis, it will be considered that this information has been sent through an error-free channel.

2.5.3.1.3 Object Description Information

Elementary streams are one of the major new conceptual elements in object-based video coding systems. And in order for these streams to be accessible they must be described by an object descriptor. These object descriptors are conveyed between peer terminals in the form of object descriptor streams. It can be anticipated that the information carried in these streams is very important and its loss can be disastrous. In fact, either detected or undetected errors will make it so that essential streams become unavailable and this information will be impossible to conceal. Therefore, these streams should be very carefully designed since the information they carry is essential and protective schemes will definitively have to be implemented at the systems layer, in order to guarantee that these streams are error-free (e.g.,

2. Error Resilient Video Coding: Organizing and Classifying

by applying strong channel coding). In this Thesis, it will be considered that this information has been sent through an error-free channel.

2.5.3.2 Systems Layer Resilience

In the previous video coding systems, the systems layer was a rather simple layer, which basically provided synchronization between the video and the audio channels, by inserting timing information in the bitstream. In object-based systems, however, this layer has evolved to become quite complex. It not only does the usual synchronization job but, due to the possibly great number of elementary streams, it also provides tools that allow multiplexing together streams with similar requirements in terms of transport channel characteristics. Therefore, the data that has to be dealt with in the systems layer can be grouped in the following categories:

- **Timing and synchronization information** – The timing and synchronization information have the responsibility of guaranteeing the temporal synchronization between the various elementary streams. This information basically consists of time-stamps and clock references. The clock references allow the decoding terminal to know the current time and the time-stamps allow the terminal to know when certain events should take place. Undetected as well as detected errors in this kind of information will make the decoding terminal lose synchronization with the encoder but the recovery should be fairly fast, since this information is inserted very frequently in the streams exactly to avoid losing synchronization for long periods of time.
- **Multiplexing data** – Multiplexing data is the information inserted in the final streams that will allow the decoding terminal to recover the various elementary channels that are multiplexed in one single stream. A loss in this information will cause the decoding terminal to mix up the different channels and produce a lot of erroneous information. This should be clearly avoided by using framing mechanisms that allow the decoder to distinguish between the several channels without mixing up all the information.

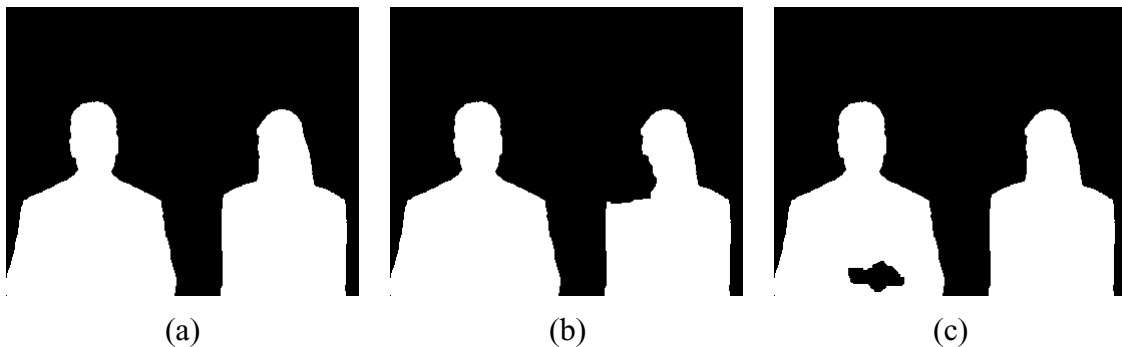
Since timing and synchronization data, as well as multiplexing data, are subject to errors, their error resilience provides very challenging problems that also need solving. However, in this Thesis, these problems will not be dealt with. Instead, it will be considered that the errors that affect this type of data will reflect themselves in the video object elementary streams in the form of residual errors.

2.5.4 Error Resilient Object-based Decoding and Concealment

In object-based video coding systems, bitstream errors can lead to shape, motion and texture artifacts. However, since motion and texture artifacts should be similar to the artifacts that already occurred in frame-based systems, shape artifacts will be given special consideration here. As was seen above, shape data is used to describe the arbitrary shape of video objects. For binary shapes (a pixel is either in or out of the object), as well as for grayscale shapes (each pixel inside the object has a transparency value associated with it), there is a direct correspondence between the shape of an object (or its support, in the case of grayscale shapes) and its contour, which is basically its border. Since it is widely accepted that human beings are very sensitive to edges [Marr82] and the contour is a very special edge (as will be seen later in Chapter 6), it can be envisioned that this type of information will be of the

utmost importance in the reconstruction of object-based video scenes. In fact, shape data might well be the most important type of data of the three required (shape, motion, and texture) because the correct use of motion and texture data may be highly dependent on the availability of the correct shape data, as is the case for MPEG-4 video coding [MPEG4-V]. Therefore, the loss of the shape data can create artifacts able to severely degrade the subjective quality of the (decoded) video content. Such artifacts, which are illustrated in Figure 2.7, may fall under the following three categories:

- **Shape distortion with an unchanged number of regions** — the shape of the various regions (or the only region) of the object is distorted, but the number of object regions remains unchanged; this includes the following two more specific cases:
 - *Severe shape distortion* — when severely corrupted, some shapes might become unrecognizable, which is especially true with elementary geometric shapes such as circles or rectangles;
 - *Appearance of holes in the object* — due to channel errors, a small area within a given region of the object can be turned into an area that does not belong to the object, thus creating holes inside the video object.
- **Shape distortion with an increased number of regions** — the shape errors are such that the number of object regions increases; this includes the following two more specific cases:
 - *Appearance of new non-connected regions* — due to the errors, an area (usually rather small) not belonging to the video object can be turned into an area that belongs to the object (although not connected to it), thus creating new non-connected regions belonging to the same video object;
 - *Splitting of regions* — due to the channel errors, one (or more) of the object regions can be split into two or more regions, thus increasing the number of object regions.
- **Shape distortion with a decreased number of regions** — the errors in the shape data are such that the number of object regions decreases; this includes the following two more specific cases:
 - *Disappearance of regions* — one or more entire object regions may “disappear” due to errors, thus decreasing the number of object regions;
 - *Merging of non-connected regions* — due to errors in the shape data, two or more non-connected regions can be merged together, thus creating a unified object region.



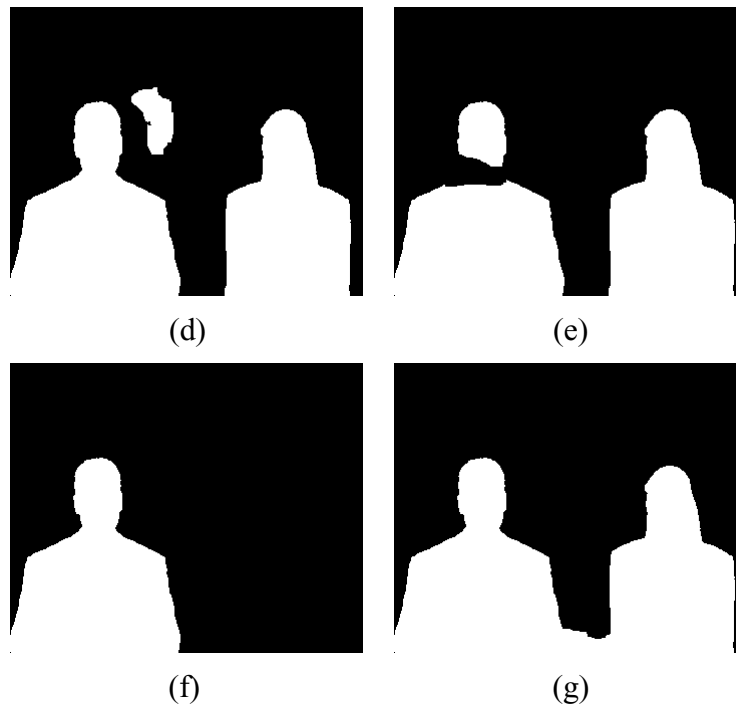


Figure 2.7 - Examples of shape artifacts due to errors – (a) Original of an object with two regions; (b) Severe shape distortion in one of the object regions; (c) Appearance of a hole in one of the object regions; (d) Appearance of a new non-connected region; (e) Splitting of a region; (f) Disappearance of one region; (g) Merging of two regions.

Due to these shape artifacts, subjectively very poor video scenes can be reconstructed, rapidly turning unusable the decoded video material, unless some action is taken to recover from these artifacts. Therefore, in order to guarantee that object-based systems can work properly in error-prone environments, the following tasks have to be addressed, following the approach presented in Section 2.4:

- **Detection of corrupted shape data** — In order to improve the subjective quality of the decoded video, it is essential that shape errors are detected; otherwise corrupted shape data may also be interpreted as being correct, thus leading to very negative results in terms of subjective impact;
- **Localization of corrupted shape data** — Then, it is essential to know as precisely as possible which parts of the shape data were correctly decoded and which were corrupted by localizing the error(s) in the most precise way possible;
- **Concealment of corrupted shape data** — Finally, after the corrupted shape zones have been identified, it is necessary to conceal them to decrease the subjective negative impact of the artifacts by using the available correctly decoded data.

To perform the shape error detection and localization tasks, a similar approach to the one presented in Section 2.4 can be followed. However, as for the error concealment task, some more problems exist, inherently related to the fact that the video is object-based. These aspects are discussed in the following section.

2.5.4.1 Error Concealment in Object-based Video Systems

In object-based video coding systems, video objects are encoded independently, although they together build a scene. This gives rise to the following question: should video objects be concealed independently or should the video scene be concealed as a whole? An absolute answer to this question that is adequate in all situations, does not exist because video objects can be used in many different situations with very different constraints. For instance, in some cases it would be clearly beneficial to use both approaches sequentially. This means that first each video object in the scene would be concealed independently in order to obtain a good estimate of the missing data at the object level. Afterwards, this first estimate would be refined by submitting the various concealed objects to a higher level of concealment considering the scene as a whole. For example, this solution would be adequate for the case of a video scene with two segmented objects that fit together like the pieces in a jigsaw puzzle: their shapes complement each other in their common border and, therefore, this information has to be used in the concealment phase. On the other hand, in some other cases it might be harder to decide if the scene level concealment is really necessary. For instance, if a single arbitrarily shaped object is being used over a uniform rectangular background, independent concealment at the object level might be enough.

In the previous examples, error concealment in object-based systems was presented as a two-stage problem: object level and scene level. This means that in general error concealment in object-based systems can be structured in two levels:

- **Object level concealment** – This level deals with error detection and concealment for each single object. The simplest way of doing this is by using for each object only its own data and no data from other surrounding objects; this means that each object is concealed as a stand-alone object (independently) and no information other than the one decoded for it is used (only its elementary stream is used). However, it is also possible to consider more complex cases where some information relative to the surrounding objects is used.
- **Scene level concealment** – This level deals with error detection and concealment for the whole scene, notably for the composition of the individual objects. This type of concealment has to deal with the relationship between objects as well as with the composition information where their spatial and temporal positioning is encoded. A simple example of scene level concealment is a technique guaranteeing that neighboring video segmented objects perfectly fit together (have complementary shapes) in the zone where they have common borders. Of course, a way would be needed for the encoder to signal the decoder that this is supposed to happen (this is not an obligation) or, alternatively, it can be assumed by the decoder to happen for specific applications/contexts.

In addition to the object/scene dimension, another important dimension exists and can be exploited in terms of error resilience, which is the space/time dimension. This new space/time dimension can be exploited both in the object and scene level concealment. This means that concealment techniques may be classified depending on the information that they use to perform the concealment, notably:

- **Spatial or intra concealment** – In this case, the information used for concealment is obtained, at a given time instant, from spatially surrounding areas in either the video object or the video scene, depending on the concealment level (i.e., object or scene), in order to explore any spatial redundancy that the video object or the video scene, respectively, may have;

- **Temporal or inter concealment** – In this case, the information used for concealment is obtained from other temporal instants, such as previous time instants, either within the same video object or video scene, depending on the concealment level (i.e., object or scene), thus exploring any temporal redundancy the video object or scene, respectively, may have.

These two approaches can be used separately or can be combined in a spatio-temporal concealment scheme. It is up to the decoder to decide which of the three approaches to use (spatial, temporal, or spatio-temporal) depending on its specific constraints, notably on the computational power available for concealment purposes and on the concealment benefits that may result from each approach. This decision in itself is a complex problem and will be dealt with in Chapter 6.

Although both object level and scene level concealment introduce some new challenging problems, scene level concealment deserves some further considerations. These will be made in the next section.

2.5.4.2 Further Considerations on Scene Level Concealment

Composition is a new operation that was not necessary in frame-based video coding systems because the scene was simply a single matrix of pixels. Since in object-based systems the scene is seen as a composition of several arbitrarily shaped objects that have to be used to compose the final scene to be presented to the end-user, some entity has to take all the objects and build a scene with them. This may create many problems in terms of concealment (e.g., segmented objects have typically to fit in perfectly) and has to be considered at a different level because it is not directly related to the insertion of a new type of information.

As mentioned in Section 2.5.1, to define video scenes one of the two approaches shown in Figure 2.8 can be adopted. A scene can be defined either by segmentation of an existing video sequence or by composition of pre-segmented video objects. These two cases will imply different problems and solutions when it comes to error concealment. Additionally, it is also possible to define a scene using a combination of both approaches above.

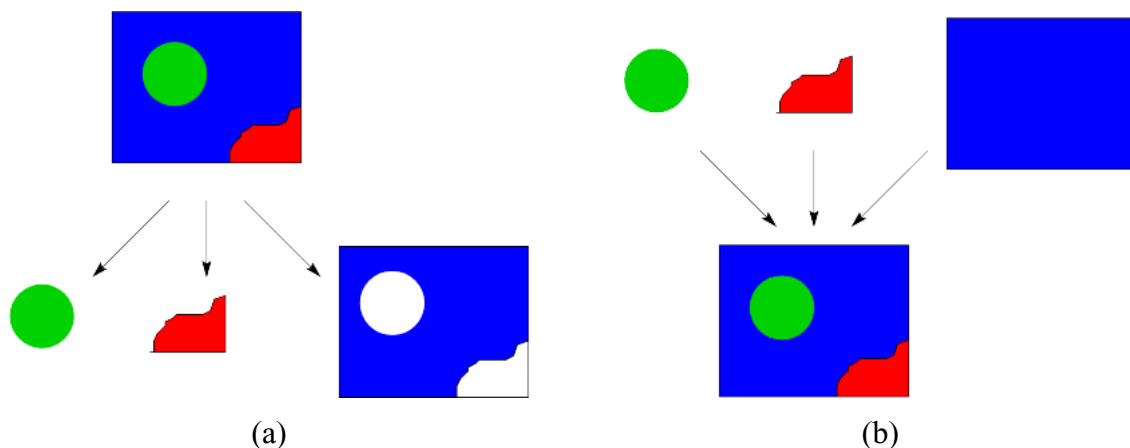


Figure 2.8 – Two different scene types: (a) Segmented scene; (b) Composed scene

The segmented scene case corresponds to the case where the different objects are extracted from a rectangular video sequence. This means that the objects fit perfectly in terms of shape and there is no overlapping between the various objects. The second case corresponds to the case where the scene was built by using existing pre-segmented objects. In this case, partial

or complete overlapping between objects may happen, which is the case in TV weather reports where the speaker walks in front of a weather map. These cases present different kinds of problems and are treated separately in the following.

For segmented scenes, if there are any distortions in the shape data, holes or overlapping of objects will appear. Object overlapping is less critical than holes because it is less visible. Another case that may happen regards the positioning of the objects in the scene: if this position is not correctly decoded, the object can be misplaced, thus leaving an empty zone, which can be very annoying.

However, the scenes of this type have one feature that can be used in their favor, which is the fact that the various objects have to fit in perfectly. This means that, if the shape of a given video object is corrupted, the shape of the surrounding objects (which is complementary) can be used to conceal it. Or, if the object is misplaced, due to an error in its position, the shape of the other objects can also be used to estimate its correct position.

For composed scenes, however, the situation is slightly different. Since there is at least some overlapping between objects, this means that holes will not necessarily appear due to shape distortions in the objects. This is especially true for small amounts of distortion, which means that even by performing error concealment that is not perfect, a good subjective impact may still be obtained (with no holes in the scene). With small misplacements of the video objects, caused by an error in the position, holes will also not necessarily appear. Of course, for large distortions or misplacements, holes may appear.

What was said above clearly represents an advantage over the segmented scenes. However, in composed scenes, since the various video objects do not have to fit in perfectly, the shape of surrounding objects cannot be used to conceal shapes of corrupted ones, as was the case for segmented scenes.

2.6 Final Remarks

In this chapter, the problems associated with the error resilience of video coding systems were dissected and the possible ways to solve them were classified according to several categories. In the first part of the chapter, this was done in a very general way, in order to include frame-based video coding systems as well as object-based ones. Afterwards, in the second part of the chapter, the specific problems of error resilience inherent to object-based video coding systems were discussed. For this, the architecture of an object-based video coding system with all its components was studied. This study has shown that a lot of room exists for innovation, especially in the field of shape error resilience.

It should be noted that the classification of error resilience techniques proposed in this chapter was done independently of any particular video coding solution or standard, even though some of the examples that were given were taken from actual video coding systems. This way, it is expected that the proposed classification will be able to accommodate most, if not all, existing error resilient video coding solutions.

In the following chapter, a review of the available error resilience techniques will be made, both for frame-based and object-based systems. The techniques will be organized according to the classification proposed in the present chapter. In this classification, the first division is between error resilient encoding techniques and error resilient decoding techniques. At the encoder side, error resilience techniques are further divided between forward error resilience techniques (i.e., techniques that do not rely on the existence of feedback from the decoder) and back-channeling techniques (i.e., techniques that rely on feedback sent from the decoder).

2. Error Resilient Video Coding: Organizing and Classifying

At the decoder side, error resilience techniques can be divided into error detection, error localization and error concealment, where error concealment can be either spatial, temporal or a combination of both. As shall be seen, by combining several of these techniques, it is possible to make the difference between being able to transmit video over error-prone environments with an acceptable quality or not.

Chapter 3

Error Resilient Video Coding: A Review

3.1 Introduction

In Chapter 2, the problems associated with the error resilience of video coding systems were dissected and the possible ways to solve them were classified according to several categories. This was first done in a general fashion in order to be able to include frame-based as well as object-based coding techniques. Then, the specific problems of error resilience for object-based video coding systems were studied. In particular, the architecture of an error resilient object-based coding system, including all its components, was analyzed, as well as the problems it faces and the possible ways to solve them. However, the actual existing solutions for specific error resilience problems have not been mentioned yet.

The purpose of Chapter 3 is to review the frame-based and object-based video error resilience techniques available in the literature. To help the reader understand under which category a given error resilience technique is included, the structure adopted for this chapter is similar to the one of Chapter 2. However, due to space limitations, not all categories of techniques are reviewed but only those related to the techniques proposed in this Thesis. This way, at the encoder side, the sections on channel coding and on joint source-channel coding techniques have not been considered for this reviewing, while at the decoder side, the sections on error detection and error localization have also not been addressed. After all, the number of existing error resilience techniques is just too great to describe them all here. Therefore, even for the categories of techniques that will be considered here, a special emphasis will be given to the techniques that are more related to the techniques proposed in this Thesis. This includes techniques on which those proposed in this Thesis are based, as well as competitor techniques against which those proposed in this Thesis can be directly compared. All other techniques will be more briefly reviewed. For the more interested reader, a good review of many of the available (frame-based) techniques can be found in [Wang98] and

[Katsaggelos98]. In terms of object-based techniques, however, very few are known and have all been described in this chapter.

3.2 Error Resilient Encoding

Similarly to what was done in the previous chapter, this section is also divided in two sections, one devoted to forward systems and the other to back-channeling systems. However, in terms of error resilient encoding techniques, only source coding level techniques will be described here in each one of these sections. Otherwise, each section would become excessively large. Additionally, since, in terms of error resilient encoding, the proposed technique in this Thesis is a coding refreshment technique to be used on block-based hybrid coding schemes, a special attention will be given to existing block-based refreshment techniques.

3.2.1 Forward Systems

This section is devoted to the available error resilience techniques that can be used in forward systems, which, as explained in Chapter 2, correspond to unidirectional systems where the encoder receives no feedback from the decoder. In these systems, all the encoding decisions, including those related to error resilience, have to be based only on *a priori* knowledge. However, through the intelligent design of the syntax and of the encoder, it is possible to greatly improve the error resilience of the whole system, thus making the task of the decoder significantly easier.

Due to the importance of coding refreshment techniques in this Thesis, an entire section will be devoted to them (i.e., Section 3.2.1.1). The other available techniques have been grouped into several categories and are only briefly described in the remaining sections.

3.2.1.1 Coding Refreshment Techniques

When designing a video coding refreshment scheme, several issues must be considered in order to achieve the best performance in the context of the application in question. Although the way to refresh the coded video data is entirely up to the encoder, and can be done in many ways, some general requirements and constraints have a special impact on coding refreshment solutions. These requirements and constraints are:

- **Maximization of the video quality** - The video quality degradation due to channel error propagation shall be minimized; this means that error resilience techniques, including coding refreshment, must work so that the decoded quality under the presence of errors is as close as possible to the decoded quality in ideal channel conditions.
- **Minimization of the compression efficiency reduction** – The quality achieved with error resilient uncorrupted streams shall be as close as possible to the quality achieved with non-error resilient uncorrupted streams for similar bit rates; this implies that the bit rate overhead due to the error resilience tools shall be as small as possible so that the quality degradation for a certain bit rate is minimized, even for ideal channel conditions.
- **Minimization of the burstiness** - The bit rate of the coded stream shall be as uniformly distributed over time as possible. This implies that error resilience must not significantly affect the bitstream burstiness (peak to average number of bits ratio for a chosen temporal

window) so that bit rate resources and characteristics do not change significantly over time; this requirement is especially relevant for constant bit rate conditions.

- **Real-time encoding delays** – If video encoding takes place at the same time and pace as the transmission and consumption of the encoded data, the relevant maximum delays shall be respected so that regular video display can be made. On the contrary, off-line encoding systems do not have critical encoding delay requirements and thus the video is encoded taking as much time as needed, possibly having available the complete video data to be encoded; transmission or storage of the encoded data is typically done after the full encoding process is finished.

Notice that while for real-time encoding applications it is possible to take into account time-dependent channel information such as the instantaneous channel error statistics, for off-line encoding applications this cannot be done since this information is not yet known. This is an important factor to take into account when designing a coding refreshment technique, since the possibility to use some knowledge regarding the channel characteristics would surely improve the refreshment scheme performance.

- **Random access** - The decoder shall be able to access any part of the encoded video stream, without exceeding a given amount of time (e.g. as in video storage or video broadcasting). For applications without random access requirements, the decoder only has to be able to decode the bitstream starting from its beginning (e.g. video telephony).

For frame-based applications with random access requirements, a full frame has to be refreshed from time to time, in order to allow the decoder to access any part of the content in a limited amount of time. This is usually done in MPEG-1 [LeGall91] and MPEG-2 [Haskell97] where intra coded frames are periodically introduced. This type of intra refreshment is rather rigid since the encoder is obliged to periodically refresh entire frames (which may be problematic in terms of burstiness). However, for applications without random access requirements, fully intra coded frames are not needed. Therefore, intra refreshment does not have to be applied to whole frames, also because it leads to very non-uniform distributions of bit rate if constant quality is a target. Instead, intra refreshment can be applied to small fractions of each frame, more or less uniformly distributed along the time. This type of intra refreshment solution is more flexible and thus challenging (the periodic full intra refreshment alternative is rather rigid), and this is where most developments have been made.

The requirements above must be fulfilled while taking into account the type and amount of channel errors, the type of coding algorithm used, and the fact that resources such as the bit rate are limited.

3.2.1.1.1 Intra Coding Refreshment Approach

Intra coding refreshment techniques encode a certain amount of video data (at a given time instant) without any reference to past or future time instants, implying that the video data at hand is encoded solely using the information from the current time instant. This also means that there is no need for the decoder to have access to past or future decoded data to recover the current decoded data and therefore “previous” errors will not propagate to this data, i.e. the quality of the decoded video data is refreshed (if this intra coded data has no errors)¹. It is clear that data from future time instants is less relevant in this context since it relates to the so

¹ In H.264/MPEG-4 AVC, this is no longer the case because intra coded blocks can depend on neighboring inter coded blocks and, therefore, past errors may still propagate to intra coded blocks.

called B-frames which typically do not play the role of prediction for any other frames and thus their refreshment is not particularly useful (this has recently changed with the H.264/MPEG-4 AVC standard). This explains why, in the following of this chapter, reference will only be made to time dependencies associated with past time instants.

These techniques can be applied to frame-based video, as well as to object-based video. However, since object-based video coding is just emerging pushed by the availability of the MPEG-4 standard, only frame-based intra coding refreshment techniques are available in the literature. The first intra coding refreshment technique to appear in the literature was developed for the H.261 standard [H.261] and it is very simple. It consists of intra refreshing every macroblock (16×16 pixels of luminance and 8×8 pixels of each chrominance, in a 4:2:0 arrangement), at least once every 132 times it is transmitted. Later on, this technique was slightly changed and also included in the H.263 standard specification [H.263]. This time, each macroblock is intra coded, at least once every 132 times DCT coefficients are sent for this macroblock. After all, a macroblock can be transmitted and not have any DCT coefficients sent for it (e.g., only a motion vector). The major reason for including this technique in both standards was not to avoid channel error propagation but to control the accumulation of the inverse transform mismatch errors at the decoder (in fact both standards include also a channel encoding tool). However, this intra coding refreshment technique is also very useful to avoid error propagation at the decoder due to channel errors and the encoder may very well use it for this purpose, as often as it finds adequate. The major advantage of the H.261 and H.263 intra refreshment solutions as an error resilience tool is their selectivity in the sense that more relevant macroblocks (which are encoded more frequently and have more coefficients transmitted and thus are more prone to errors) are also more frequently refreshed. However, intra refreshment could be further improved by considering factors other than the frequency with which the macroblock is transmitted or coefficients are sent for it. Since these standards specify a maximum (and not a fixed) refreshment period — once every 132 times — the encoder may adapt the refreshment periodicity, e.g. to the characteristics of the channel in terms of bit error rate (BER). This fact allows implementing adequate intra refreshment solutions for a large variety of transmission environments.

In [Haskell92], some other refreshment techniques are suggested. In terms of intra refreshment, the first technique proposed consists of intra coding a different part of the frame each time intra refreshment is made. This way, full refreshment will be obtained after n frames if, for each refreshment, $1/n$ -th of the frame is intra coded. Here the coding refreshment is obtained in a cyclic manner, which explains why this technique later became known as Cyclic Intra Refreshment (CIR).

In [Haskell92], it is also suggested that the intra coding refreshment could be more selective. A possible solution consists in refreshing only parts of the image that would be difficult to conceal in the case that errors occurred. This means that only the blocks that are very different from their co-located² ones in the previous frame would be refreshed; a very simple way to measure this difficulty would be by using the mean squared difference of blocks. This idea would later lead to the Adaptive Intra Refreshment (AIR) technique suggested in the MPEG-4 Visual standard [MPEG4-V] for frame-based video. However, in the AIR technique, the Sum of Absolute Differences (SAD), computed between a given macroblock and the co-located one in the previous frame, is used to determine if the macroblock should be refreshed or not. If the SAD value is greater than a certain threshold, the macroblock is

² A co-located macroblock is a macroblock, which has the same spatial position as the current macroblock but in another frame.

considered to be an important area that should be refreshed. These important macroblocks will be cyclically refreshed as described above for CIR. This technique is based on the fact that fast changing areas are more important to the user and therefore are the ones that should be more frequently refreshed. In addition, macroblocks with greater SAD values will usually spend more bits and therefore are more prone to errors.

Both CIR and AIR have advantages and disadvantages. The main advantage of CIR is that all the macroblocks in the frame will be refreshed, but its main disadvantage is that the most important and the least important macroblocks are treated alike. As for AIR, its main advantage is that resources (i.e. bit rate) are not wasted on the least important macroblocks and only important macroblocks are intra refreshed. On the other hand, its main disadvantage is that the least important macroblocks may never be refreshed and errors in those macroblocks can propagate indefinitely. This is why, in the MPEG-4 video error resilience tests performed in June 1998 [N2165], a combination of CIR and AIR was used as frame-based intra coding refreshment solution. This way, the CIR compensates for the AIR disadvantages and vice-versa. The results obtained are presented in [N2604].

In [Liao96] and [Liao00], another scheme is proposed where the intra refreshment rate for different parts of the image is adaptively adjusted to the channel conditions and image characteristics. This scheme is based on an error sensitivity metric, which is accumulated at the encoder and represents the vulnerability of each macroblock to channel errors. The intra refreshment decisions will be made according to the values of this accumulated vulnerability metric for each macroblock. Two approaches are suggested for the intra coding decisions: the first one is to intra code, in each frame, all the macroblocks whose accumulated vulnerability metric exceeds a predefined threshold; the second one is to intra code the N (user defined) macroblocks with the highest accumulated vulnerability metric. In [Liao00], this technique was compared to the two intra refreshment solutions proposed in [Haskell92] and the results have shown a significant improvement in terms of error recovery time, defined as the average number of elapsed frames before a badly corrupted macroblock is refreshed.

3.2.1.1.2 Inter Coding Refreshment Approach

Inter coding refreshment is similar to intra coding refreshment in the sense that both have the objective to recover as much as possible the error-free quality by avoiding error propagation. However, here the coding refreshment is done while still making reference to past information in order to limit the reduction in terms of compression efficiency. At first, this might seem contradictory but it is, in fact, possible to conceive such a scheme where the encoding is still done with respect to the past but errors do not propagate indefinitely. The solution for this apparent contradiction is to make the contribution of past information fade away with time.

The only such technique of this type known in the literature is the so-called leaky-difference refreshment [Haskell92], based on leaky Differential Pulse Coded Modulation (DPCM). In leaky-DPCM schemes, instead of sending the difference between two samples,

$$diff_i = sample_i - sample_{i-1}, \quad (3.1)$$

a weighted difference is sent,

$$diff_i = sample_i - \alpha \cdot sample_{i-1}. \quad (3.2)$$

This way, the contribution of $sample_{i-1}$ to $diff_i$ is scaled by α . Since α is between 0 and 1, the effect of erred past samples will decay exponentially, instead of propagating forever as in

regular DPCM schemes. The exponential decay can be explained by considering that a given sample, $sample_i$, can be reconstructed at the decoder by using

$$sample_i = diff_i + \alpha \cdot sample_{i-1}, \quad (3.3)$$

where $sample_{i-1}$ is obtained by using

$$sample_{i-1} = diff_{i-1} + \alpha \cdot sample_{i-2}. \quad (3.4)$$

By replacing Equation (3.4) in Equation (3.3), the following result is obtained

$$sample_i = diff_i + \alpha \cdot diff_{i-1} + \alpha^2 \cdot sample_{i-2}, \quad (3.5)$$

which can be generalized to the N previous samples

$$sample_i = diff_i + \alpha \cdot diff_{i-1} + \alpha^2 \cdot diff_{i-2} + \dots + \alpha^{N-1} \cdot diff_{i-(N-1)} + \alpha^N \cdot sample_{i-N}. \quad (3.6)$$

As can be seen from Equation (3.6), the dependency of $sample_i$ on $sample_{i-N}$ is with α^N , showing indeed that the effect of erred past samples decays exponentially.

In [Haskell92], this decaying effect is considered and the idea is applied to a hybrid coding scheme³, which means that instead of sending motion vectors and the difference signal

$$diff_i = frame_i - frame_{i-1}[motion\ compensated], \quad (3.7)$$

the encoder sends motion vectors and a weighted difference signal

$$diff_i = frame_i - \alpha \cdot frame_{i-1}[motion\ compensated]. \quad (3.8)$$

This way, the effects in the current decoded frame of errors in past decoded frames will decay exponentially. In order to choose an appropriate value for α , the usual trade-off between coding efficiency and error resilience has to be considered since as α approaches 0, errors will decay faster but more bits will be needed if the same quality is wanted (less inter coding, lower compression performance).

This leaky-difference refreshment technique can also be used in a more selective way, for instance by using different α values for each block, e.g. depending on its content. In that case, the used α value will have to be transmitted along with the block data itself, unless different α values are pre-fixed, e.g. for each type of macroblock. This way, blocks that are significantly different from the previous co-located ones should, in principle, use lower α values, which will limit degradation due to error propagation from previous frames. On the other hand, blocks that do not change significantly should use α values closer to 1, because they are easier to conceal if needed.

The main advantage of leaky-difference techniques is that errors fade away smoothly over time, whereas with other methods the errors propagate until the affected video data is intra refreshed, erased by motion compensation or error concealed. This means that error concealment becomes less critical because, even if it is not applied at the decoder, the video data will regenerate itself. If an intra refreshment technique is used instead, the fact that the quality of some video areas is refreshed all at once can be annoying to the user in itself. This happens because errors often appear as blocks with a sudden change in luminance (even if some concealment is applied). And when these blocks are refreshed with an intra refreshment method, the luminance will suddenly be replaced with the correct values, which means that

³ A hybrid coding scheme is based on transform coding (e.g. DCT) and motion compensation, combining techniques in the spatio-temporal and frequency domains.

the user will be annoyed by two sudden transitions: when the error occurs and when the error disappears. Even if the luminance transitions are small, they can be annoying to the user because human beings are very sensitive to luminance changes. On the other hand, with the leaky-difference method the user will only notice one sudden transition (when the error occurs), because the effect of the errors will be smoothly erased over time (although not too slowly). As for disadvantages of the leaky-difference methods, three major ones stand out. The first one is that these methods cannot cope with very high BER values because new errors will occur before the effect of the previous ones fades away. The second one is that, when no refreshment is necessary, more bits are usually needed than if a non-leaky difference coding scheme were used. And finally, it is important to stress that while intra coding refreshment does not affect compatibility with currently available coding standards, leaky-difference techniques cannot be used without giving up on standard compatibility (or changing the standards) since the way to decode has to be changed (more precisely, the way temporal predictions are formed).

3.2.1.2 Layered Coding

Since the appearance of digital video, one of the most popular techniques to achieve error resilience has been layered coding, as can be seen by the number of published papers on the subject [Ghanbari89], [Nomura91], [Zhu93], [Ramchandran93], [Zhang94], [Khansari95], [Ayanoglu95], [Khansari96], [Aravind96] and [Tsai97]. In fact, layered coding has even been adopted by international standards such as MPEG-2 [MPEG2][Haskell97] where it can be used for frame-based video or MPEG-4 [MPEG4][Pereira02] where it can be used for both frame-based as well as object-based video.

The idea behind layered coding is to divide the coded video data in two or more layers of different importance. The base layer carries the essential information, without which the other layers are useless, and is enough to recover low-quality video at the decoder side. As for the additional enhancement layers, each one of these layers relies on the layers below it and carries information to provide an increment in terms of decoded video quality, spatial resolution, etc. Due to the hierarchical nature of this coding scheme, best results will be achieved if channels with different channel error characteristics are used for the different layers. For instance, since the base layer is certainly the most important because all other layers depend on it and it is the only layer that can be decoded by itself, it should be sent through the channel with the lowest channel error rate (ideally, the channel should be error-free). As for the enhancement layers, channels with progressively higher channel error rates (or decreasing amounts of channel coding protection) can be used, since the transmitted data is less critical because the layers below can be used to decode lower quality video at the decoder. Thus, by following this approach (i.e., a base layer continuously improved by the enhancement layers), the error resilience of video coding systems can be greatly improved because the decoded video degrades gracefully when errors occur, instead of simply becoming unusable. The architecture of layered coding is illustrated in Figure 3.1 for two layers, which is the most common case.

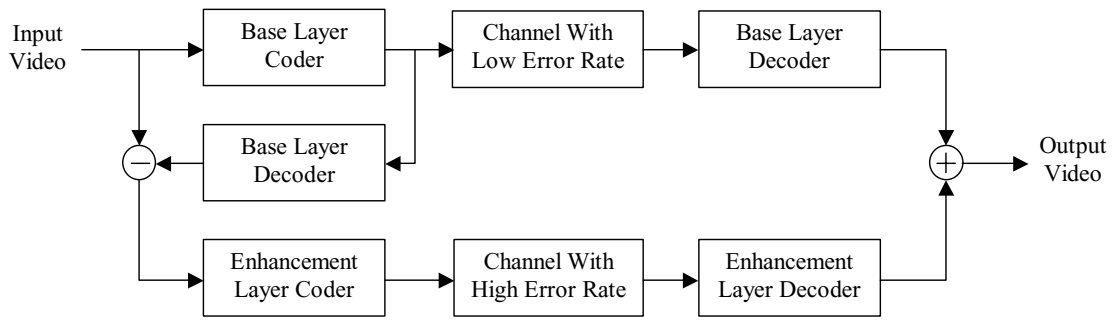


Figure 3.1 – Layered coding system (two layers) where channels with different channel error characteristics are used

3.2.1.3 Multiple Description Coding

In layered coding, it is assumed that an essentially error-free channel, or at least a channel with a smaller amount of errors than the remaining channels⁴, is available to carry the essential information at the base layer. When this is not the case, multiple description coding (MDC) appears as a good replacement. As in layered coding, the coded video data is divided in two or more channels (referred to as descriptions), but here any one of these channels is enough to reconstruct a low-quality copy of the video. This is especially useful when several channels are available but any of them can undergo a long period of errors, such as in channels with similar channel error characteristics (e.g, two different physical paths between the source and the destination in a mobile network). Similarly to layered coding, this type of techniques improves the error resilience of video coding systems by allowing the decoded video to degrade gracefully when errors occur.

To illustrate this concept, the two-description coder shown in Figure 3.2 can be considered. In this case, a low-quality version of the transmitted video can be obtained either from S1 or S2 and a full-quality version can be obtained by combining both S1 and S2. To guarantee an acceptable quality with a single description, each description must carry sufficient information about the original video data, which necessarily implies a certain amount of overlap between the different descriptions. Obviously, this leads to a reduced coding efficiency compared to the conventional single description coder (SDC), but results in an increased error resilience to long burst errors and channel failures. This trade-off has been shown using a rate-distortion analysis for different types of sources in [Wolf80], [Ozarow80] and [Gamal82].

⁴ These channels can either correspond to several different physical channels or several different logical channels (obtained by using different amounts of channel coding) within a single physical channel.

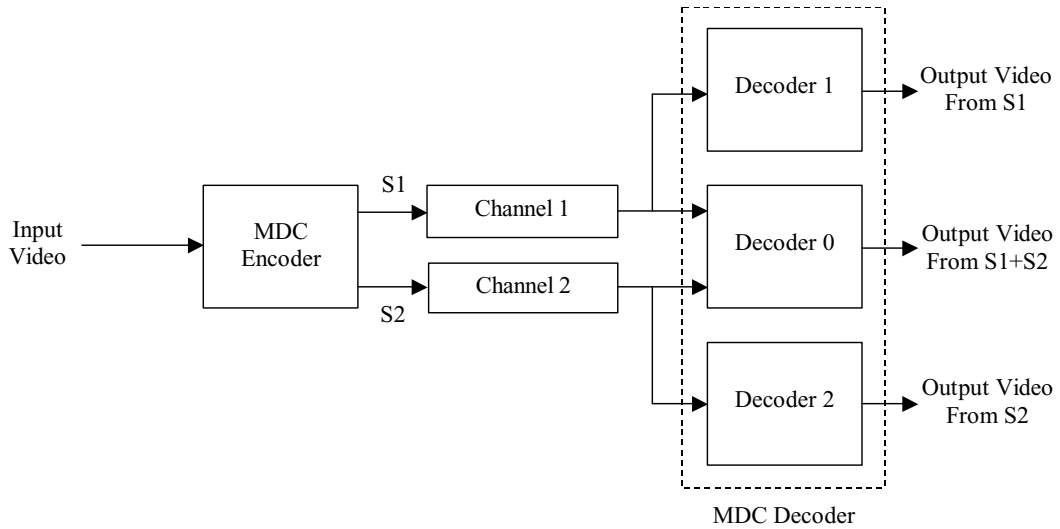


Figure 3.2 – Multiple description coding and decoding (adapted from [Wang98])

To obtain the multiple descriptions many different techniques have been proposed, as can be seen in [Tom91], [Vaishampayan93], [Vaishampayan94], [Vaishampayan96], [Battlo97], [Wang97], [Orchard97], [Hemami96], [Malvar89], [Servetto00] and [Wang01]. The techniques range from the direct splitting of adjacent pixels into several channels (e.g., even rows in one channel and odd rows in another channel) to more complex ones. For instance, a multiple description quantizer has also been proposed. With this quantizer, two descriptions of the same data can be obtained by producing two different indices for each quantized level. If both indices are received correctly, the reconstruction accuracy is equivalent to that of a fine quantizer. If, on the other hand, only one of the indices is received correctly, the reconstruction accuracy is that of coarse quantizer. This can be achieved by using two coarse quantizers whose decision levels are shifted by half of the quantizer interval with respect to each other.

3.2.1.4 Robust Symbolic Coding

The elimination of redundancy during video coding makes recovery at the decoder very difficult when errors occur. A possible solution to this problem is to intentionally leave/add some redundancy in the video data at the symbolic coding stage, hence the name robust symbolic coding. This way, when errors occur, the decoder will be able to use the extra redundancy to recover, thus improving the error resilience of the whole video coding system. Strictly speaking, MDC also belong to this category, since by separating the coded data into two or more channels they are not able to eliminate the existing redundancy as efficiently as when the data is not separated; therefore, some redundancy is still left in the video data. However, in this section, only techniques that do not depend on several channels are considered, which does not mean that they cannot be applied to individual streams in MDC.

A simple way to make symbolic coding more robust is to add auxiliary information. For instance, in MPEG-2 Video, the encoder has the option of sending motion vectors for macroblocks in I-frames, so that I-frames can be more easily recovered [MPEG2] with motion compensation concealment (see Section 3.3.2.1). Of course, in the absence of channel errors, these motion vectors are useless. More complex schemes have been also proposed such as the one in [Hemami94] where the additional information (i.e., a set of weights) allows a damaged block to be interpolated from correctly received neighboring blocks.

Another possibility to make symbolic coding more robust is to restrict the prediction domain in order to reduce the effect of error propagation due to the use of prediction. For instance, in the MPEG-4 standard, pictures are divided into video packets [MPEG4][Wang00]. Any spatial prediction of DCT coefficients and motion vectors is limited to adjacent macroblocks within the same video packet. In terms of the temporal prediction domain, [Wenger97] proposed that input video frames be separated into different groups called threads (e.g., even frames and odd frames) and each thread be coded without using other threads for prediction.

3.2.1.5 Robust Entropy Coding & Synchronization

In the previous section, redundancy was added at the symbolic coding stage. However, redundancy can also be added at the entropy coding level in order to make it more robust to errors; the extra redundancy will help the entropy decoder to recover, thus making the whole system more error resilient. Since the main problem of entropy coding is the typical loss of synchronization when errors occur, a simple solution to prevent this is to frequently insert synchronization code words in the bitstream. Several strategies have been proposed for the insertion of synchronization code words, such as the ones in [Maxted85], [Ferguson84], [Neumann71], [Lei91] and [Lam92], but they are all based on the fact that when the entropy decoder finds a synchronization code, word synchronization will be regained. The synchronization code word itself does not carry any information on the encoded video and, therefore, if additional side information is needed to resynchronize the source decoder, such as spatial and temporal locations, it has to be sent after the synchronization code word. In standard video coding systems such as H.261 [H.261], H.263 [H.263] or MPEG-4 [MPEG4][Pereira02], relatively long synchronization code words are used to avoid code word emulations when bit errors occur.

Another possible approach is to redesign the entropy coding itself in order to make it more error resilient. The Error Resilient Entropy Coding (EREC) methods proposed in [Cheng92][Redmill96] are a good example of this. For instance, the idea behind the method proposed in [Redmill96] is to guarantee that in block-based video coding systems the start of block data in the bitstream occurs at predefined fixed intervals, thus allowing the decoder to regain synchronization at the start of each block; in the paper, this was tested for 8×8 blocks. This way, error propagation is predominant only in higher (less important) frequency coefficients, if coefficients are transmitted in a zig-zag order with low frequency coefficients first. Several applications of the EREC method have been proposed such as those in [Swann96] and [Kawahara96]. Another example of error resilient entropy coding is reversible variable length codes (RVLC), which are designed in such a way that when a synchronization code word is found, the bitstream can also be decoded backwards, thus minimizing the amount of discarded data when errors occur [Takishima95][Wen97]. These codes are currently used in the MPEG-4 standard [MPEG-4][Pereira02].

3.2.2 Back-Channeling Systems

This section is devoted to the error resilience techniques available in the literature that can be used in back-channeling systems. These systems, as explained in Chapter 2, correspond to systems where the encoder can receive feedback from the decoder through a back-channel. In these systems, all the encoding decisions, including those related to error resilience, can be based on the feedback sent from the decoder, thus allowing a better adaptation of the coded bitstream to the transmission conditions. This is a great advantage over forward systems, because the coding decisions are no longer based on assumptions, but rather on facts.

Similarly to what was done with Section 3.2.1, an entire section is devoted to coding refreshment techniques. As for all the other techniques, which are based on the retransmission of previously sent data, they are more briefly described in Section 3.2.2.2.

3.2.2.1 Decoder-Assisted Coding Refreshment Techniques

As explained in Section 3.2.1.1, the objective of coding refreshment techniques is to avoid long error propagation due to the use of prediction. However, in forward systems, the coding refreshment has to be based only on *a priori* knowledge, which can result in a lot of wasted bit rate if the assumptions are too pessimistic. This is where back-channeling systems have a great advantage over forward systems. If the decoder can provide information about the location of the damaged parts to the encoder, then the encoder will precisely know which parts of the images are in need of coding refreshment in order to minimize or eliminate error propagation. This way, a simple technique one can conceive is that whenever the decoder detects an error (while performing source decoding), it signals the encoder. Since motion vectors only have a limited range [Yasuda77][Mukawa84], only part of the next video frame may be affected by the damaged data and, therefore, only that part of it needs to be intra coded. Thus, upon receiving the signal from the decoder, all the encoder has to do is determine which parts of the next video frame will be affected and intra code them. This is sure to stop error propagation in only one round-trip time. However, in environments with very critical channel error characteristics, this can mean that a large percentage of the transmitted data is intra coded, which may lead to an unacceptably high bit rate or, if the bit rate is fixed, to an unacceptably low quality.

To improve coding efficiency, the two schemes proposed in [Wada89] can also be used. When an error is detected at the decoder, the decoder sends a message to the encoder identifying the image blocks that have been affected by the error and performs some kind of concealment on the affected data. At the encoder, two different strategies are proposed. In the first one, to encode the next image, the encoder will simply do it without using the affected area for prediction. Notice that encoding without using the affected area for prediction does not necessarily mean intra coding. In the second method, based on the information sent from the decoder, the encoder performs a local concealment of the damaged blocks (using the same technique as the decoder). Then, since the encoder now has the same previous image to perform prediction as the decoder, it can continue encoding in a normal fashion.

Along this direction, the H.263 standard [H.263] allows the encoder to specify which reference picture it is using for prediction. Since the encoder can use one of several past frames for prediction, it can adapt its coding strategy based on the feedback sent from the decoder. This way, if the decoder signals that an error was detected in a given frame, the encoder will simply proceed by using a different frame for prediction, thus avoiding any error propagation. Of course, the bitstream has to carry information about which frame is being used for prediction in order to guarantee that the decoder will also use the same one for prediction. In very critical situations in terms of channel error characteristics, it is possible that all the potential frames to be used as prediction contain errors. In this case, this scheme can be combined with one of the above. For instance, to encode a given frame, the encoder can start by choosing the reference frame for prediction that has the least errors. Then, encoding can be done without simply using the corrupted areas for prediction.

In the above techniques, the decoder explicitly says what image blocks have been affected by errors and, therefore, have to be refreshed. However, a different strategy is possible such as the one proposed in [Horne93]. Here, the decoder sends back to the encoder information about the cell-loss statistics. Based on this information, the encoder can adapt its encoding

parameters to match the channel conditions. When the cell loss rate increases, the encoder reduces the size of the slices and uses more intra coding. On the other hand, when the cell loss rate decreases the slice size is increased and the amount of intra coding is reduced. The main advantage of this solution over the previous ones is that it does not involve any complex processing either at the encoder or the decoder and, therefore, can be used even for very simple terminals.

3.2.2.2 Back-Channeling Techniques Based on Retransmission

In the above techniques, the encoder adapts its refreshment decisions based on the feedback from the decoder. However, other schemes are possible based on the feedback from the decoder. These are the schemes based on retransmission. In these schemes, the encoder does not have to adapt its coding decisions based on the feedback, but it will have to resend some of the data that has been corrupted. Examples of these schemes are given in [Zhu96] (also described in [Wang98]) and [Ghanbari96]. These schemes, however, are different from typical techniques based on retransmission, where the decoder would have to wait for the retransmitted data. Here, after the decoder has signaled the encoder that a block of data contains errors, the corresponding corrupted part of the image is concealed and decoding continues normally, while tracing the affected data in subsequent frames and storing the associated decoded data (e.g., coding modes, motion vectors). After the retransmitted data has arrived, the decoder will simply decode it, propagate it until the current time instant (by using the stored associated decoded information) and replace the affected data. This stops the error propagation right away and no additional delay is introduced.

3.3 Error Resilient Decoding and Concealment

As described in Chapter 2, three types of error concealment are possible depending on the type of information that is used to perform it: spatial, temporal or spatio-temporal. These three categories of error concealment have been extensively investigated and numerous techniques have been proposed for frame-based video. However, in terms of object-based video, the interest of researchers working in the field is just awakening and it is expected that many new techniques will soon appear in the literature. In fact, a few object-based error concealment techniques are already available. These techniques shall be summarized in the following three sections.

3.3.1 Spatial Error Concealment

As explained in the previous chapter, spatial error concealment techniques only rely on the available information from the current time instant to perform the concealment operation; no information from other time instants is used. Therefore, due to its characteristics, this type of techniques can not only be used to conceal errors in video sequences but also in still images.

Since spatial error concealment techniques are no longer available only for frame-based video coding systems, in the following they will be clearly separated according to the type of system that is targeted. This way, Section 3.3.1.1 is dedicated to the available frame-based error concealment techniques, while Section 3.3.1.2 is devoted to object-based error concealment techniques.

3.3.1.1 Frame-based Spatial Error Concealment Techniques

The number of available error concealment techniques for frame-based video coding systems is extremely large and, therefore, the idea of this review is not to exhaustively describe them all, but rather provide a brief description of only the most important ones. However, there is one exception, which is the texture concealment technique described in Section 3.3.1.1.1. This technique is rather important in the context of this Thesis, since the spatial shape concealment technique proposed later in Chapter 6 is based on the same approach. The other available techniques have been grouped into several categories and are only briefly described in the remaining sections.

3.3.1.1.1 Texture Concealment using a Sketch-based Approach

The shape error concealment technique proposed in this Thesis is related to the sketch-based image coding approach, which has been well studied in [Graham67][Carlsson88][Grattoni90] and is well suited for very low bit rate image coding. The main idea behind sketch-based image coding is that an image can be fairly well represented by its high-frequency content (i.e., edges) alone. In addition, due to the typically very smooth variation of the luminance data encompassed by the edges, the missing gray level information can be simply interpolated from the gray level data on both sides of each edge, and thus recovered. This way, to represent an image, the edges of the image and the gray level texture information immediately on each side of the edges are enough. These two types of data combined are called the *image sketch* and are generally enough to reconstruct a perceptually good replica of the original image. By carefully selecting which edges are used in the image representation, the necessary bit rate to encode the image can be adjusted (e.g., by selecting only the most important edges, the bit rate can be reduced). This technique has later been extended to color images [Cumani91].

In [Atzori98], the sketch-based approach has been used to conceal the errors in the texture of natural images by rebuilding the most important edges and then recovering the low-frequency content. Since block-based coding is assumed⁵, bitstream errors manifest themselves as lost blocks (or blocks that have been declared lost since the decoded information is not considered reliable enough). Therefore, the decoded image will have several missing (texture) blocks, which have to be concealed. For that, the sketch elements from the correctly decoded blocks (blocks that were not lost) are used. For each missing block, the important edges that are going into the missing block from the 8 surrounding blocks are taken. These edges are matched two by two and, finally, those chosen as best fitting according to some criteria are connected across the missing block by simple cubic spline interpolation. The values of the gray level data on each side of the interpolated edge are also interpolated with cubic splines. This technique allows recovering the image sketch. Finally, to recover the texture, each pixel in the lost block that does not belong to the image sketch is assigned the value given by the weighted mean of the four nearest known pixels in the four directions (up, down, left, right). This sketch recovery technique is illustrated in Figure 3.3, where a lost block is shown with two broken edges going into it. These edges will be interpolated in order to connect points A and B.

⁵ Block-based coding understood as coding by dividing the full image or frame in square blocks is very generally used, notably in all currently available image and video coding standards.

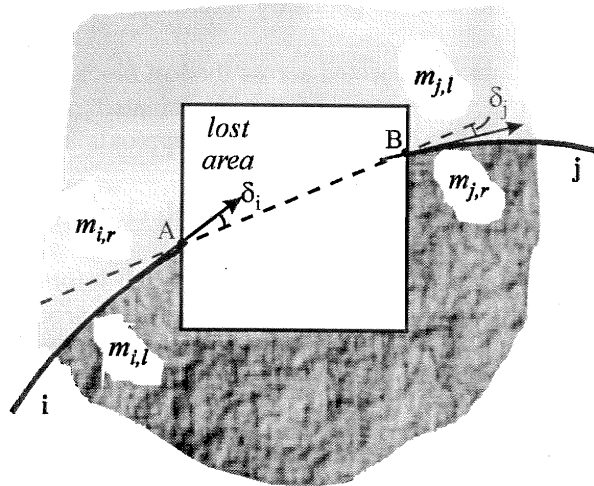
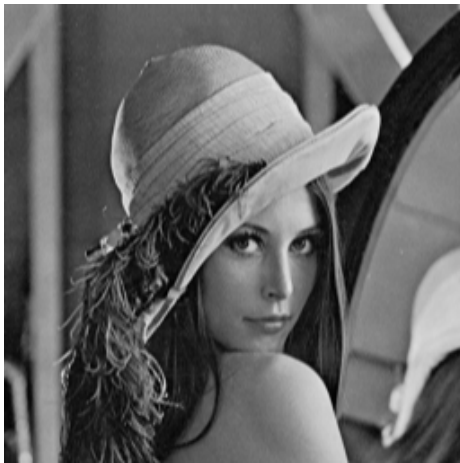
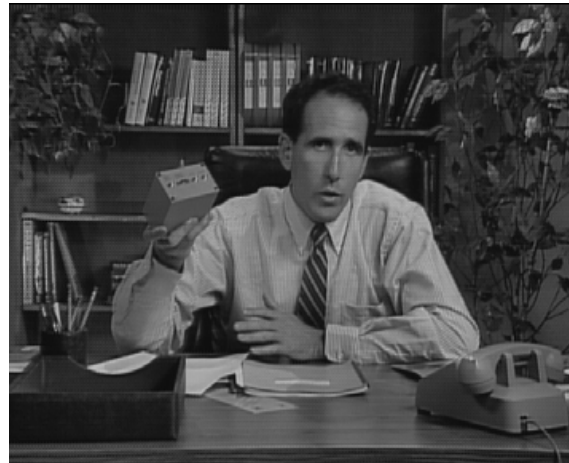


Figure 3.3 – Sketch recovery technique [Atzori98]

In [Atzori98], this method has been tested for several grayscale still images and video frames, with different resolutions and complexity. The errors were simulated by a regular distribution of single lost blocks over the whole image, in order to test the behavior of the concealment technique in areas with different luminance and frequency characteristics. According to the authors [Atzori98], this technique allows typical “head and shoulders” natural images, such as the Lena and Salesman images illustrated in Figure 3.4, corrupted with a 15% block loss rate to be recovered without any noticeable differences with respect to the original. In Figure 3.5, results are given for the Maskerade image. It should be noticed, however, that the used error model is not very realistic because channel errors are more likely to manifest themselves as bursts of consecutive lost blocks, instead of single isolated lost blocks. This aspect seriously compromises the usability of this technique and, therefore, a more realistic error model will be used later in Chapter 6, when a similar approach will be used to conceal corrupted shape data (not texture data).



(a)



(b)

Figure 3.4 – Typical “head and shoulders” natural images: (a) Lena; (b) Salesman

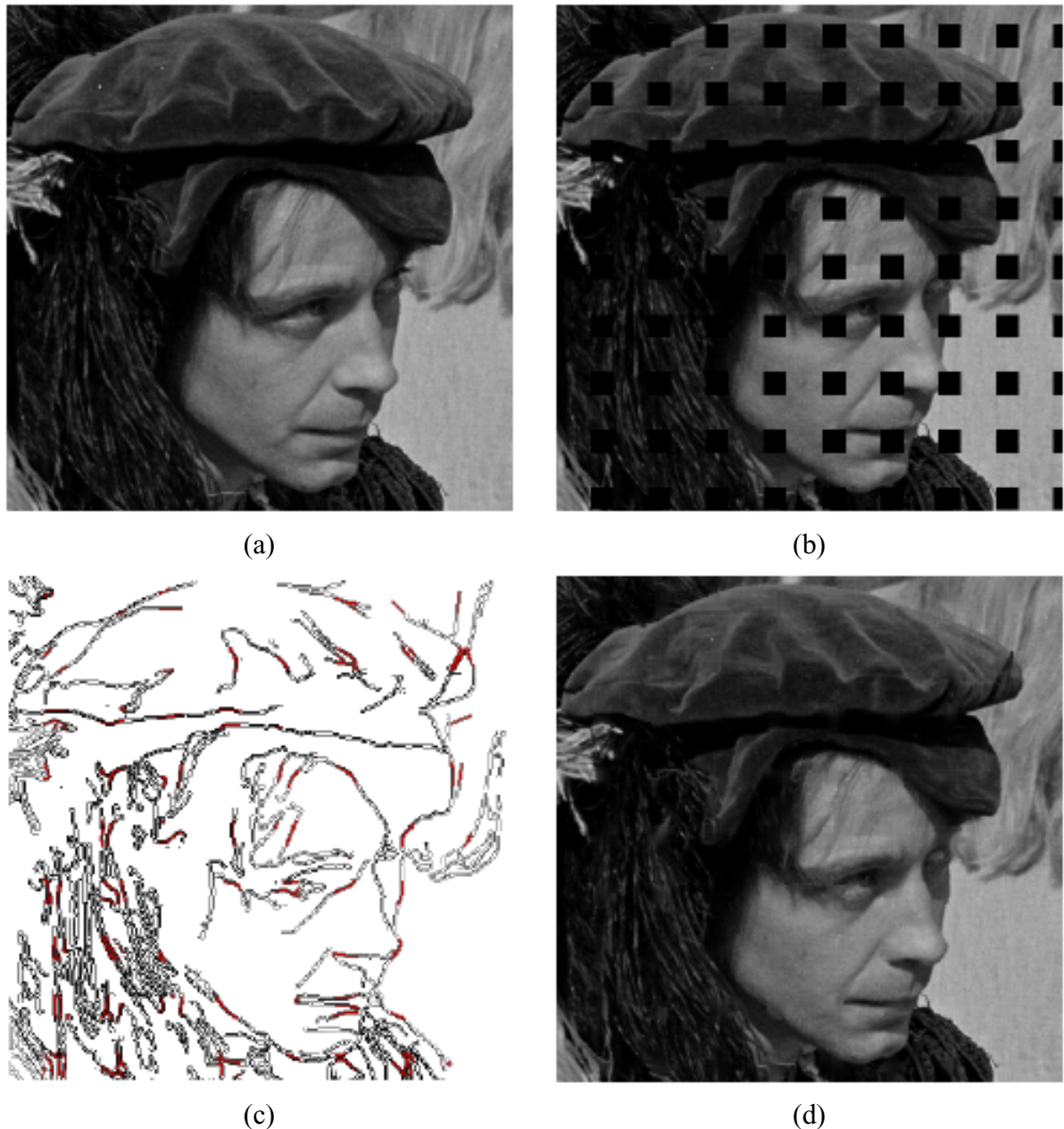


Figure 3.5 – Sketch recovery procedure [Atzori98]: (a) Original Maskerade image (detail); (b) Simulated losses of 16×16 blocks; (c) Recovered sketch; (d) Concealed image

3.3.1.1.2 Maximally Smooth Recovery

The next spatial error concealment technique described here is referred to as *maximally smooth recovery* and was proposed in [Wang93] for frame-based still images, but can also be directly used for frame-based video. This technique targets block-based transform coding systems, where the transform coefficients have been partially lost in some blocks. To illustrate this, the authors consider a two-layer coding system, where the base layer carries the low-frequency transform coefficients and the enhancement layer carries the high-frequency coefficients. In these two-layer systems, when the enhancement layer is lost or corrupted, a lower quality image can still be displayed by using only the low-frequency coefficients in the base layer. However, since it is not always cost-effective to guarantee that the base layer is error-free, the real problems start to appear when the base layer is lost or corrupted. In this

case, the high-frequency coefficients, if they are uncorrupted, are not enough to obtain a reduced quality image and the quality will simply be unacceptable.

In [Wang93], the authors reasonably assume that most images and video frames are smooth and use a constrained energy minimization approach, on a block-by-block basis, to estimate the missing DCT coefficients in each corrupted block and, finally, recover the pixel values of the block. For this, it is necessary to minimize a measure of spatial variation between adjacent pixels in the corrupted block and the spatially neighboring blocks, so that the resulting recovered visual data is as smooth as possible. The spatial variation measure is defined as the sum of squared differences between spatially adjacent pixels, as shown in Figure 3.6. The number of adjacent pixels included in the measure depends on what coefficients are being estimated. On one hand, when only the DC coefficient is lost and needs to be estimated, the authors have shown that the smoothing constraint only needs to be enforced on the border pixels, as shown in Figure 3.6 (a). On the other hand, when additional coefficients have been lost, all the pixels in the damaged block should be included in the smoothing constraint, as shown in Figure 3.6 (b). With this spatial variation measure, the authors have shown that the solution to this problem, which yields the concealed pixel values for the corrupted block, consists basically of two linear interpolations (in the spatial and frequency domains) of the pixels in adjacent image blocks and the received coefficients for the block being concealed. If all the coefficients have been lost, the solution degenerates in a simple spatial interpolation using the pixel values from the surrounding blocks. To further improve the reconstruction results of the maximally smooth recovery method, other smoothness measures have been proposed, such as those in [Zhu95] and [Kwok93]. Since the maximally smooth recovery method can be quite computationally intensive, in [Park97], the authors propose an alternative smoothness measure, which allows the computation of the missing DCT coefficients to be much faster than in [Wang93].

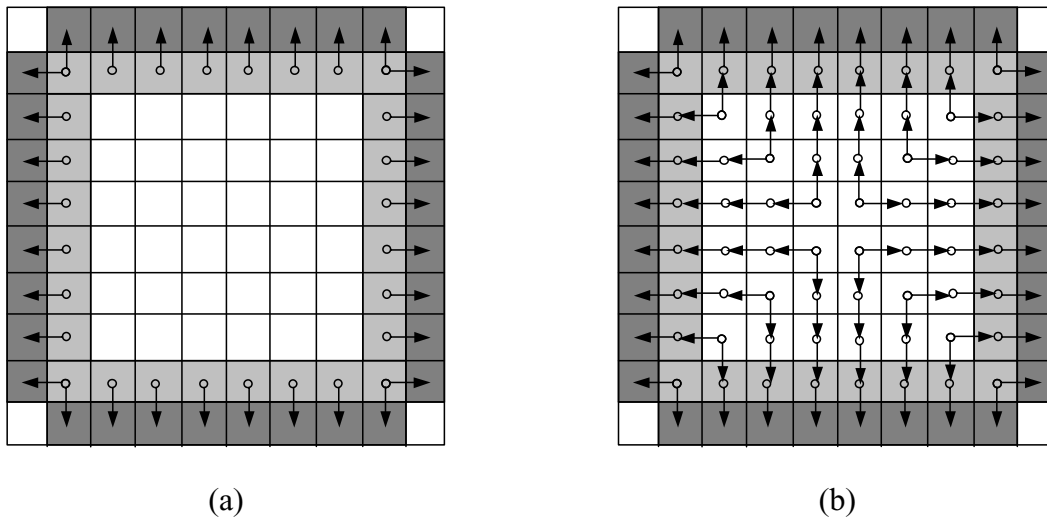


Figure 3.6 – Illustration of smoothing constraints, where an arrow between two samples means that the difference between these two samples occurs in the smoothness measure: (a) Smoothing constraint imposed only on the block boundary; (b) Smoothing constraint imposed on each sample in the direction towards the nearest block boundary [Wang93]

Independently of the smoothness measure that is used, the above concealment procedure is adequate for isolated blocks. However, it is quite common for consecutive blocks to be corrupted. In this case, as described in [Wang93], the blocks are first reconstructed by setting the missing coefficients to some initial values and computing the inverse transform. Then, the

concealment procedure described above is applied, using the values previously obtained as the boundary information necessary to compute the spatial variation measure. This is iteratively repeated until the change in the pixel values is less than a certain threshold. When this happens, the reconstruction for this block is terminated. This process is repeated until all the damaged blocks are reconstructed satisfactorily. Since the performance of the maximally smooth technique when consecutive blocks have been corrupted is not as good as for isolated corrupted blocks, to further improve the reconstruction quality, it is possible to use spatial block interleaving in order to guarantee that adjacent blocks are transmitted separately so that channel errors will only affect spatially apart blocks [Tom91].

3.3.1.1.3 Projection Onto Convex Sets

Instead of using an energy minimization approach to enforce the smoothness of the texture for the concealed block, another possible approach is the one described in [Sun95], which is based on the projection onto convex sets (POCS) method first introduced in [Youla82] [Sezan82]. In [Sun95], the authors propose to use the POCS method to conceal corrupted blocks in frame-based still images and video frames when a block-based transform coder is used. Since this method is based on convex sets⁶, the first step is clearly to define the used convex sets. In [Sun95], the authors define two convex sets. The first set, $C1$, is the set of pixels with luminance values in the $[0, 255]$ range. The second set, $C2$, is the set of pixels with a given band limitedness characteristic (i.e., with a given limited band along a given direction). Based on these two convex sets, the method works as follows to correct a given corrupted block:

1. First, the corrupted block, surrounded by the 8 correctly decoded blocks, is input to an edge classifier block, which determines if any edges should pass through the corrupted block. This way, the block is either classified as a *monotone block* or an *edge block*. The edge orientation is determined and quantized to one of eight directions in the $0-180^\circ$ range.
2. Then, two projection operators are alternately applied to the corrupted block, surrounded by the 8 correctly decoded blocks, as shown in Figure 3.7. The first operator is a band limitedness operator, which depends on the edge classifier output. If the block is monotone, an isotropic band-limitedness constraint is applied in the form of an isotropic low-pass filter. On the other hand, if the block has one of eight edges, then a bandpass filter is applied in that direction. The filtering operation is applied in the Fourier transform domain. The second projection operator is a pixel value limiter and truncates the pixel values to the range $[0, 255]$. The values of pixels in the 8 surrounding blocks, which were correctly decoded, are maintained. The two operators are applied alternately until the concealed block does not change any more and no more projections are needed. It was found that 5 to 10 iterations are usually enough when a good initial estimate is available.

⁶ The geometrical interpretation of a convex set is that of a set where a segment $[p_1, p_2]$ joining any two points of the set, p_1 and p_2 , completely belongs to the set.

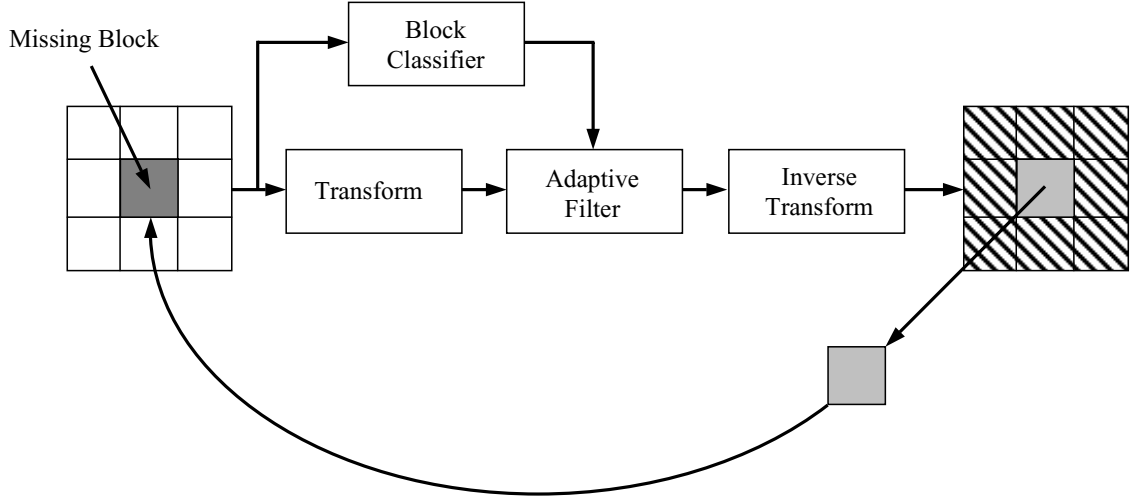


Figure 3.7 – Illustration of the adaptive POCS iterative restoration process [Sun95]

3.3.1.1.4

Pixel Value and Transform Coefficient Interpolation

Due to the smoothness property of video frames, it is expected that the correlation between transform coefficients with the same frequency index be very high in spatially adjacent blocks. Based on this, in [Hemami95], the authors proposed to recover missing transform coefficients by interpolating the corresponding coefficients from the four spatially adjacent blocks. Instead of using simple bi-linear interpolation, the interpolation was done in order to minimize the spatial variation measure proposed in [Wang93]. In the case that all the coefficients are lost and have to be estimated, this frequency-domain interpolation reduces to simply interpolating each lost pixel in the block from the corresponding pixels in the four spatially adjacent blocks. This is done instead of using the nearest available pixels. Since the pixels used to perform the interpolation are eight pixels apart, the spatial correlation is likely to be small. Therefore, in [Aign95], the authors have suggested to improve the results by using the nearest available pixels, which basically corresponds to saying that the interpolation is done by using the four available 1-pixel wide boundaries of the lost block. For this, two methods were proposed. In the first method, referred to as block-based, a given lost pixel is interpolated from the two nearest pixels in the boundaries, as illustrated in Figure 3.8 (a). For instance, the indicated pixel in block b_l is interpolated from the nearest pixel in block 3 of the top macroblock (i.e., b_{T3}) and the nearest pixel in block 2 of the left macroblock (i.e., b_{L2}); the distance to the top and left borders is d_T and d_L , respectively. In the second method, referred to as macroblock-based, a given lost pixel is interpolated from the four nearest pixels in the boundaries, as illustrated in Figure 3.8 (b). For instance, the indicated pixel is interpolated from the nearest pixel in the top macroblock (i.e., mb_T), the nearest pixel in the bottom macroblock (i.e., mb_B), the nearest pixel in the left macroblock (i.e., mb_L) and the nearest pixel in the right macroblock (i.e., mb_R); the distance to the top, bottom, left and right borders is d_T , d_B , d_L and d_R , respectively.

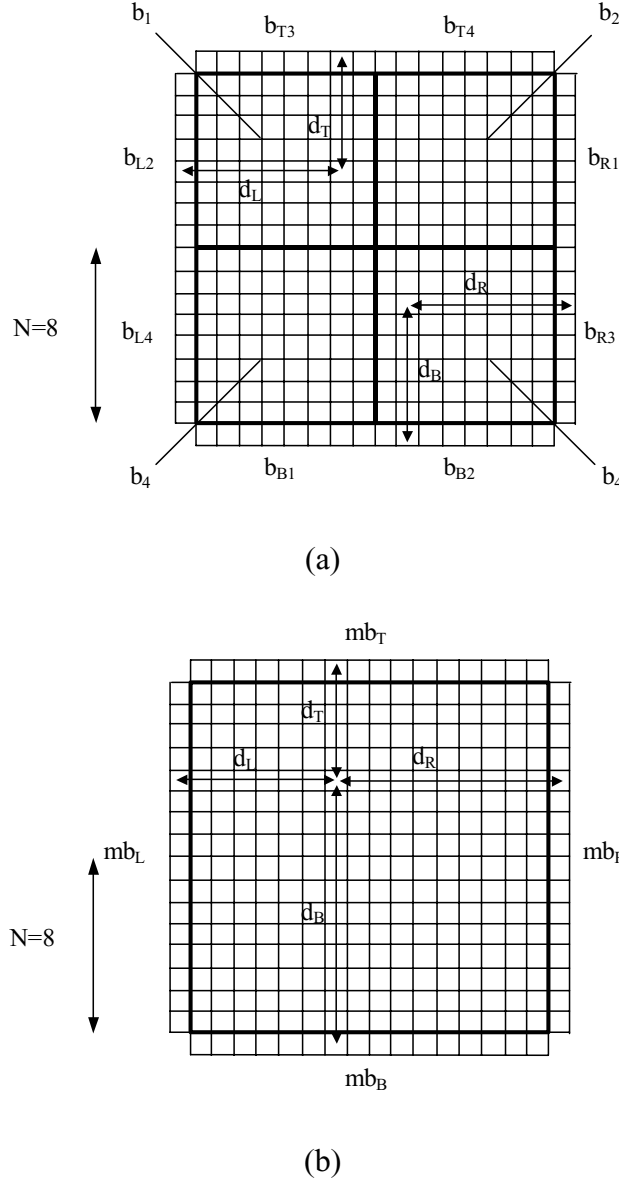


Figure 3.8 – Spatial interpolation for error concealment – (a) Block-based; (b) Macroblock-based (adapted from [Aign95])

The smoothness of video frames usually only guarantees that the lowest order coefficients are correlated to the corresponding ones in the spatially adjacent blocks, but the highest order coefficients remain fairly uncorrelated. However, this is not a major problem since the low-frequency coefficients alone are typically sufficient to recover a reduced quality reconstruction. This is why, in [Sun92], only the DC coefficient and the lowest five AC coefficients of a corrupted block are estimated from the neighboring blocks. As for the remaining AC coefficients, they are simply set to zero. In [Sun92], the DC values are linearly interpolated, while the five AC coefficients are determined according to the method specified in [Pennebaker92]. In this method, to recover the low-frequency AC coefficients, a 3×3 array of blocks, centered on the corrupted block, is considered. Based only on the DC values of these blocks, a quadratic surface is fitted to the 3×3 array of blocks; this surface is then used to estimate all the pixel values in the central corrupted block. Finally, by applying a DCT to these pixel values, a set of equations is obtained relating the AC coefficients required to

reproduce the quadratic surface to the DC coefficients used to determine the quadratic surface; only the first five of these equations are used.

3.3.1.1.5 Fuzzy Logic Reasoning

In addition to the previous techniques, a method has also been proposed to recover lost transform coefficients in block-based (image) coding systems based on fuzzy logic [Lee95]. However, this method has been especially designed to deal with blocks that have significant high-frequency content. For this, the authors consider the fuzzy set theory, which tries to model how human beings make decisions based on fuzzy data, imprecise information, vague rules and common sense. This way, a 3×3 array of blocks, centered on the corrupted block, is considered; each one of these blocks is divided into five different clusters of AC coefficients, called *subspectra*, as illustrated in Figure 3.9. Then, the various spectral features associated with the blocks surrounding the corrupted one are inspected; these features include the energy distribution within each subspectra, the energy centroid of each subspectra and the texture orientation of the block. Based on this set of features, the authors then use fuzzy logic reasoning to reconstruct the corrupted block.

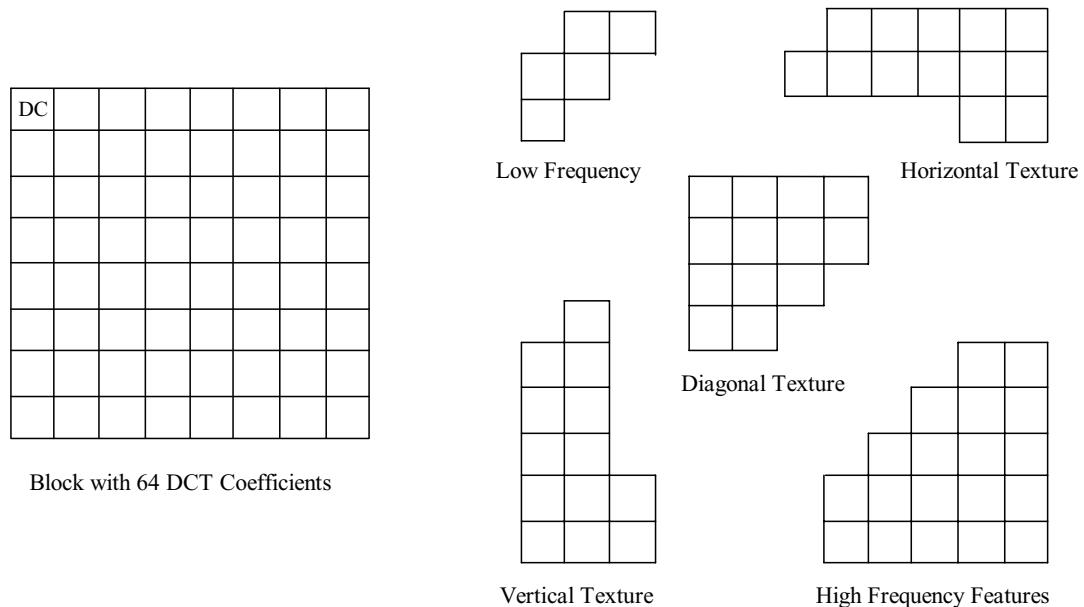


Figure 3.9 – Division of a block with 64 DCT coefficients into five different subspectra

3.3.1.1.6 Error Concealment Techniques for Less Common Video Coding Systems

All of the above methods were developed for block-based coding systems with non-overlapping transforms. However, methods have also been proposed for completely different systems, such as subband coding [Wang91][Hemami97][Rosiles95], lapped orthogonal transform [Tzou89][Haskell90] and Walsh transform [Wong78].

3.3.1.2 Object-based Spatial Error Concealment Techniques

In terms of error concealment techniques for object-based video coding systems, it is important to remind the reader that they have to deal with both shape and texture data

concealment. However, considering that the available (frame-based) texture concealment techniques can most of the times be adapted to work for object-based, researchers are mainly investing in shape concealment techniques. This explains why only shape concealment techniques are beginning to appear in the literature and no texture concealment techniques, specifically designed for object-based video coding systems, are known.

In terms of spatial shape concealment techniques, only a single technique was found in the literature [Shirani00]. In the context of this Thesis, this technique is quite important since it represents the only competitor technique to which the spatial shape concealment technique proposed in Chapter 6 can be compared. The technique proposed in [Shirani00] is based on the use of a binary Markov Random Field (MRF) of shape, in conjunction with a maximum *a posteriori* (MAP) estimation. According to this algorithm, each missing shapel⁷ in a missing 16×16 shape block is estimated as a weighted median of the 16 neighboring shapels (correctly decoded or concealed) shown in Figure 3.10 (a). The weight assigned to each one of the neighboring shapels is adaptively selected based on the likelihood of an edge in that direction. To determine the likelihood of an edge in a given direction, the blocks surrounding the corrupted block are inspected and the number of edges in each of eight possible directions, shown in Figure 3.10 (b), is counted. The rationale behind this selection is to weight more the neighboring shapels in the directions along which shapels have a higher tendency to be the same. For each missing shape block, this procedure is iteratively repeated until the algorithm converges. Additionally, if several consecutive missing blocks have to be concealed, the procedure is recursively applied to all the missing shape blocks.

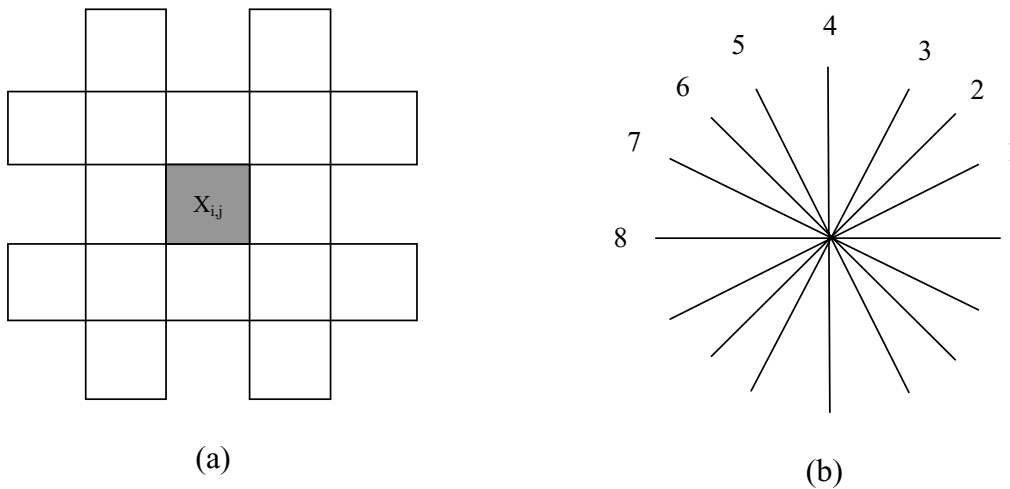


Figure 3.10 – (a) The 16 neighboring shapels used to estimate the value of the lost shapel (in gray) and (b) The 8 possible edge directions used to determine the weight associated with each of the 16 neighboring shapels

In [Shirani00], this technique was tested for relatively simple alpha planes (i.e., CIF versions of Akiyo, Bream and Weather), which have been MPEG-4 encoded with 5 to 10 video packets per VOP. In the shown results, some of these video packets were corrupted, corresponding to a percentage of lost shape blocks between 25 and 30%. Although this technique is able to conceal most of the fully opaque or transparent lost blocks with hardly any visible differences with respect to the original blocks, in most other cases (i.e., border

⁷ A shapel is an element of the alpha plane; its name results from the contraction of the words *shape* and *element*. In a binary alpha plane, the shapels can be either opaque, if they are inside the object, or transparent, if they are outside.

blocks) it can lead to rather poor (subjective and objective) concealment results. This happens because the shapes of a given missing shape block are concealed based on the local shape data statistics of the surrounding shape blocks, which is an adequate solution for isolated lost blocks. However, when several consecutive lost blocks have to be concealed, which actually happens much more often than isolated blocks, many of the lost blocks are going to be concealed based on the statistics of surrounding blocks that have already been concealed themselves. Therefore, when many consecutive lost blocks exist, this technique tends to include severe error propagation. As shall be seen in Chapter 6, this can be solved by using the correctly decoded object contour to perform the shape concealment, instead of using such a block-based approach. This way, when many consecutive shape blocks have been lost, they can be concealed as a whole by considering the available correctly decoded contour, thus avoiding the error propagation and leading to better results. Another problem of the technique proposed in [Shirani00] is that it can be computationally very intensive since it is iterative.

3.3.2 Temporal Error Concealment

As explained in Chapter 2, in temporal error concealment techniques only information from time instants other than the time instant being considered is used to perform the concealment operation. In this category, several techniques have also been proposed, some of which correspond to the most popular concealment solutions, mainly due to their simplicity and the existing high temporal redundancy in typical video sequences.

Since temporal error concealment techniques are also no longer available only for frame-based video coding systems, in the following they will be clearly separated according to the type of system that is targeted, as was done for spatial error concealment techniques. This way, Section 3.3.2.1 is dedicated to the available frame-based error concealment techniques, while Section 3.3.2.2 is devoted to object-based error concealment techniques.

3.3.2.1 Frame-based Temporal Error Concealment Techniques

The simplest way to exploit the existing temporal redundancy in (frame-based) video sequences is by replacing a damaged area with the co-located area in the previous time instant. In macroblock-based schemes this would correspond to replacing a corrupted macroblock with the co-located macroblock in the previous time instant. However, for sequences with significant motion, this method can produce visual artifacts with a very negative subjective impact. Instead, to significantly improve the results, it is possible to replace a corrupted macroblock with the macroblock specified by the motion vector of the damaged block. In fact, due to its simplicity and efficacy, this is probably the most widely used error concealment technique. It is so popular that, in MPEG-2 [MPEG2], it is even possible to send additional motion vectors for intra coded blocks, in order to be able to adequately recover them if they are corrupted during transmission. This clearly shows the importance of the motion data and explains why it is preferred to transmit the motion data in such a way that texture and motion data will not be corrupted at the same time. This can be achieved in several ways. For instance, in [Ghanbari93], a layered coding scheme is considered that includes all the motion information in the base layer, in order to guarantee that it will be available. Another possibility to achieve this is the data partitioning mode defined in MPEG-4 Visual [MPEG4-V][Wang00], where the motion information is separated from the texture, in order to avoid corrupting the motion data when errors corrupt the texture data [Talluri97]. For object-based video, an extended data partitioning scheme has been

suggested [Soares98a], where the shape, the motion and the texture are all sent in different partitions.

Nevertheless, this approach also has some problems that must be dealt with. The first problem is that the motion information has to be available. Therefore, the motion vectors have to be separately sent from the texture, in order to guarantee that they will not be corrupted at the same time. However, there will still be situations where the motion vectors will also be corrupted. In this case, it is necessary to estimate them from the surrounding correctly decoded information (e.g., the motion vectors of the surrounding blocks). This should be carefully done because a significant error in the motion vector can lead to very negative visual artifacts. Another problem with this approach occurs when the original macroblock was coded with intra mode, due to a scene change (i.e., low temporal redundancy). In this case, the concealment method above can lead to very bad results, because the temporal redundancy is low. Several methods have been proposed to estimate missing motion vectors based on the motion vectors from the past time instant, such as the two in [Haskell92]. In the first one, when a motion vector is corrupted, the decoder simply copies the motion vector from the corresponding position in the past time instant. In the second method, all the nearby motion vectors in the past time instant are searched. The one that best moves its associated block into the position of the corrupted block is chosen to replace the corrupted motion vector.

In [Kieu94], a similar motion compensation concealment approach was considered for a layered coder, where the base layer includes the motion vectors and the low-frequency coefficients and the enhancement layer includes the high-frequency coefficients. Instead of proceeding as usual in layered coding, when the enhancement layer is lost, which is simply to ignore that data and use the base layer alone, Kieu and Ngan have used the motion vectors from the base layer to conceal the high-frequency coefficients that have been lost in the enhancement layer. It was shown that this approach can improve the reconstructed quality of the video over the usual approach. For this study, the authors have assumed that the base layer is transmitted without errors. When errors hit the enhancement layer, the motion vectors in the base layer are used to determine the motion compensated macroblock from the past. The DCT is applied to the blocks inside the determined macroblock. The obtained high frequency coefficients are then used together with the low frequency coefficients in the base layer. The inverse DCT is computed and the damaged macroblock is concealed.

3.3.2.2 Object-based Temporal Error Concealment Techniques

As already explained in Section 3.3.1.2, researchers are mainly investing in new shape concealment techniques for object-based video coding systems. In fact, in terms of temporal shape concealment, two techniques are known in the literature. The one described in Section 3.3.2.2.1 is especially important in the context of this Thesis, since the shape concealment technique proposed in Chapter 6 is based on the same global motion compensation approach, although it includes some important changes.

3.3.2.2.1 Shape Concealment by Global Motion Compensation

The temporal shape error concealment technique proposed in this Thesis follows the same global motion compensation concealment approach as previously proposed in [Salama02]. The technique described in [Salama02] assumes that the shape of a given video object, although it can move in time, is rather rigid, meaning that it can only suffer very little deformation in consecutive time instants. This way, the motion of the object shape can be

fairly well described by one of the more common global motion models; this model is also used to conceal the current corrupted shape based on the previous shape.

In [Salama02], since block-based shape coding is assumed, bitstream errors manifest themselves as lost blocks (or blocks that have been declared lost since the decoded information was not considered reliable enough). Therefore, when corrupted, the decoded image will have several missing shape blocks, which have to be concealed. To conceal these lost blocks, the technique proposed in [Salama02] can be divided into four sequential tasks or steps that have to be performed by the decoder, as shown in Figure 3.11.

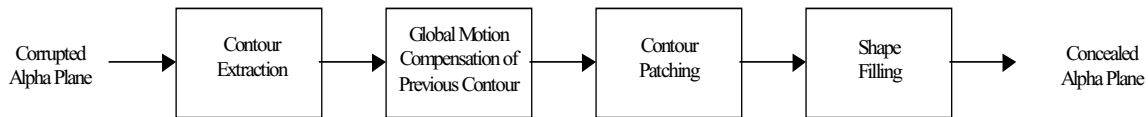


Figure 3.11 – Different modules of the global motion compensation shape concealment method

The four steps that have to be followed in order to conceal the corrupted alpha plane are described next:

1. **Contour extraction** – The first step to conceal the lost blocks in the alpha plane is to extract the contour from the corrupted shape data; the contour will be broken if and only if some of the lost shape blocks were border blocks. This happens because an uncorrupted shape always leads to a closed contour, as will be shown at the beginning of Chapter 6; if the lost blocks do not correspond to border blocks, the contour extracted from the corrupted shape is in no way affected. An example of a broken contour, extracted from the shape of the Bream video object, is shown in Figure 3.12 (a), where two separate contour segments have been identified. Each of these contour segments has two end points, which have also been identified (e.g., for contour segment 1, the end points are labeled 1 and 1').
2. **Global motion compensation of previous contour** – The second step is to take the shape from the previous time instant (correctly decoded or already concealed), extract its closed contour and motion compensate it with the global motion parameters corresponding to the current time instant, which have been computed at the encoder and sent to the decoder. If the small deformations assumption made by the authors is valid, this should lead to a contour that is very close to the current contour and which will be referred to as the *reference contour* from now on. Due to the similarity, the reference contour could be readily used to replace the corrupted contour. However, since some of the corrupted contour was correctly decoded, there is no sense in discarding that part of it because it can be used to improve the temporal concealment results. This leads to the next step in the concealment operation.
3. **Contour patching** – The third step corresponds to patching the broken contour with the available information from the reference contour. This way, the decoder starts by locating in the reference contour the points that correspond to the end points previously identified in the corrupted contour. This is illustrated in Figure 3.12 (b), where the reference contour (corresponding to the corrupted contour shown in Figure 3.12(a)) is shown with the corrupted contour end points on it. This way, all the contour points in the reference contour between two end points with different numbers correspond to missing parts of the corrupted contour (e.g., the points

between 2 and 1). Then, all the decoder has to do is to follow the reference contour in a predefined direction (clockwise or anticlockwise) and every time a missing part of the contour is reached, the corresponding contour points are copied from the reference contour (i.e., the previous motion compensated contour) to the corrupted contour, thus recovering the whole contour.

4. **Shape filling** – Finally, the fourth step, which corresponds to recovering the shape from the contour, it is simply a matter of filling in the shapes while taking into account the recovered contour.

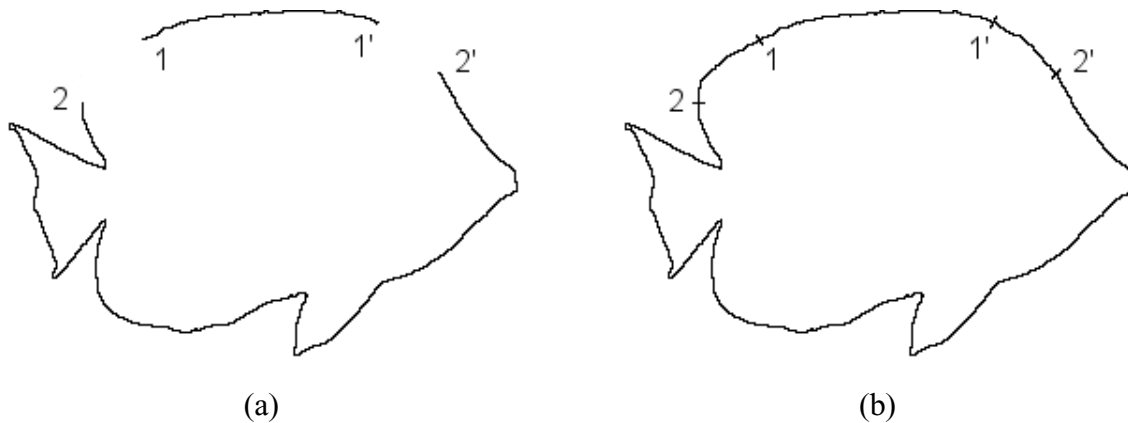
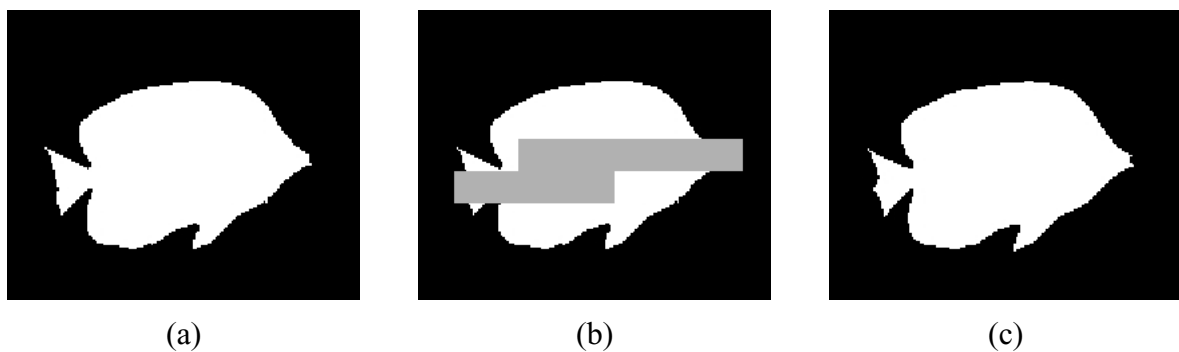


Figure 3.12 – Illustration of the global motion compensation method to recover the object contour – (a) Broken contour extracted from a corrupted shape of the Bream object; (b) Reference contour with overlapped end points from the corrupted contour

With this technique, the authors are able to achieve rather good results for well-behaved video object sequences in terms of global motion, such as the Bream sequence, for which results are shown in Figure 3.13. These results correspond to the QCIF version of the referred video object sequence, MPEG-4 encoded (with lossless shape coding) at 122.4 kbps, using video packet sizes of 384 bytes. These video packets were then packed into ATM cells that were transmitted and subjected to a random packet loss of 3%. Under these conditions, the differences between the original shape shown in Figure 3.13 (b) and the concealed shape shown in Figure 3.13 (d) are very hard to spot by the naked eye, which can be explained by noticing the very little differences between the two shapes shown in Figure 3.13 (f).



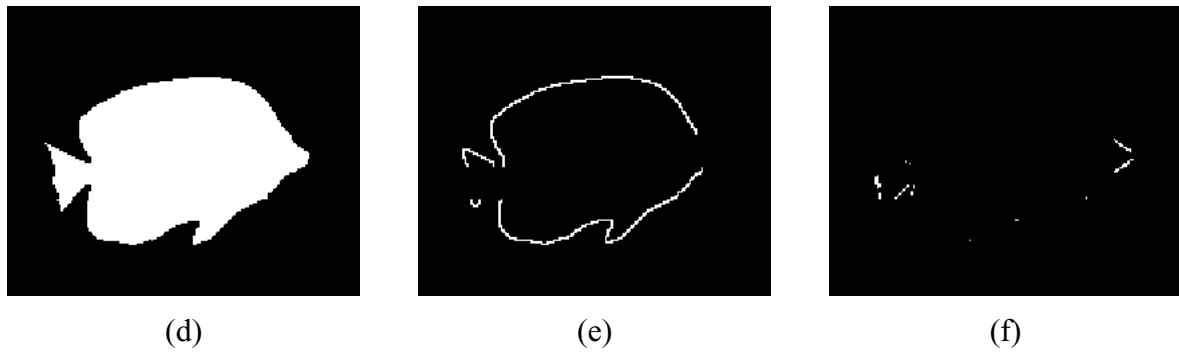


Figure 3.13 – Shape error concealment by global motion compensation as in [Salama02] - (a) Original shape; (b) Corrupted shape; (c) Reference shape; (d) Recovered shape; (e) Contour of the corrupted shape; (f) Differences between original and recovered shapes

The main drawback of this technique is that the global motion parameters are estimated at the encoder, when the sequence is being encoded. Since these parameters are needed at the decoder for the concealment itself, they must be transmitted somehow. To do so, a separate stream, called USER_DATA stream in the MPEG-4 nomenclature [MPEG4-V], is used in [Salama02]; this stream represents a 5% increase in terms of bit rate. This can be a serious limitation because this concealment technique can then only be used if both the encoder and decoder support it, which is very unlikely in such a competition-driven market with many manufacturers (and remind that concealment is non-normative). In addition, other issues would have to be considered, such as how to synchronize this new USER_DATA stream with the visual data stream itself and what should be done if this stream is corrupted. In this context, a concealment technique able to provide similar concealment capabilities without having to rely on specific processing and information received from the encoder would be highly desirable since it would be more generic and efficient. This is the challenge that Chapter 6 intends to answer.

3.3.2.2.2

Block-based Motion Compensated Error Concealment Technique

The motion compensation approach to error concealment described in Section 3.3.2.1 for frame-based systems, which simply corresponds to replacing a corrupted (texture) macroblock with the macroblock specified by the motion vector of the damaged block, has also been applied to object-based video. However, since in object-based video both the shape and texture data can have motion vectors associated to it, this technique can be used for the texture data, as well as for the shape data. To employ this approach to object-based video, it is simply a matter of extrapolating the technique. This way, when errors occur in the shape data, the decoder has to use the shape motion vectors (correctly decoded or estimated based on the correctly decoded information) and copy the indicated areas from the alpha plane in the previous time instant. For the texture data, the same procedure is followed but the texture motion vectors are used to copy the indicated texture area from the previous time instant. The first time this approach was used in object-based video was during the MPEG error resilience core experiments [Brady97] while developing the MPEG-4 standard.

3.3.3 Spatio-Temporal Error Concealment

As explained in Chapter 2, in spatio-temporal error concealment techniques, both the information from the current time instant and the information from other time instants is used to perform the concealment operation. The idea behind this is that, by combining the

information from the current time instant (used in spatial concealment techniques) with the information from other time instants (used in temporal concealment techniques), it should be possible to develop a technique that has the advantages of both types of techniques. In this category, several techniques have been proposed in the literature. The existing techniques are either spatial techniques that have evolved into the temporal domain or temporal techniques that have evolved into the spatial domain. As was done for previous sections, the techniques described in the following are organized according to the type of system they are targeted for.

3.3.3.1 Frame-based Spatio-Temporal Error Concealment Techniques

For instance, the maximally smooth recovery method proposed in [Wang93] started out as a fully spatial concealment technique as described in Section 3.3.1.1.2. However, in [Zhu93], the maximally smooth recovery method was extended to the temporal domain for use on video coders based on motion compensation and transform coding by adding the temporal smoothness measure. In this latter case, the function being minimized becomes a weighted sum of a spatial difference measure and a temporal difference measure, where the spatial and temporal difference measures are defined as the sums of squared differences between spatially and temporally adjacent pixels, respectively. Similarly to what happened when only the spatial domain was considered, the authors have shown that the solution to this problem consists basically of three linear interpolations (in the spatial, temporal and frequency domains) from the pixels in adjacent image blocks that have been reconstructed previously, the prediction block in the previous frame, and the received coefficients for this block, respectively. When all the coefficients are lost in a damaged block, the solution degenerates into a simple spatial and temporal interpolation only.

For the POCS method, described in Section 3.3.1.1.3, it is also possible to add the temporal domain in a similar fashion. Since the POCS method needs an initial estimate for corrupted blocks, one way to make use of the temporal information is by using the motion-compensated block from the previous frame as the initial estimate. Then, the POCS technique would be used to further improve the reconstruction accuracy. Other variations of this technique, where information from several frames is taken into account, also exist [Yu98].

As for the widely used motion compensated concealment technique, which simply consists of replacing a corrupted macroblock with the macroblock specified by the motion vector of the damaged block (see Section 3.3.2.1), it has also eventually evolved into the spatial domain. After all, when the motion vectors needed for the concealment are lost, they can be estimated from the information in the spatially surrounding blocks. For instance in [Haskell92], the authors investigated two possible solutions. The first one is to simply replace the corrupted motion vector with the average of the motion vectors of spatially surrounding blocks. The second one, which achieved better results, is to replace the corrupted motion vector with the median of the motion vectors in the spatially surrounding blocks. In [Lam93], the authors have suggested a method based on block boundary matching to choose a motion vector from several available candidates, including motion vectors from the past as well as from the neighboring blocks. An improved version of this technique is proposed in [Chen97]. Later, in [Hong99], this idea is taken a step further by considering the correctly decoded texture data in surrounding blocks to estimate a new motion vector to replace the lost one.

3.3.3.2 Object-based Spatio-Temporal Error Concealment Techniques

The object-based version of the widely used motion compensated concealment, which was first used in [Brady97] (see Section 3.3.2.2.2), has also undergone a similar evolution. In

[Frater98], the authors consider the concealment of corrupted shape and texture data in object-based video coding systems, when the associated motion data has also been corrupted. Since the motion data is corrupted, the authors propose to locally estimate it based on the surrounding available data. This way, for each corrupted macroblock, the decoder starts by using the correctly decoded shape data in surrounding macroblocks to determine a motion vector for the shape. Then, the correctly decoded texture data in surrounding macroblocks is used to determine a motion vector for the texture. After these two motion vectors have been estimated, the concealment itself can be performed as in [Brady97]. The shape motion vector is used to copy the indicated area from the alpha plane in the previous time instant, while the texture motion vector is used to copy the indicated texture area from the previous time instant.

3.4 Final Remarks

In this chapter, a review of some of the existing error resilience techniques was made. Since the number of available technique is just too great to have an exhaustive description of them, only the more relevant in the context of this Thesis have been described. This includes techniques on which the ones proposed in this Thesis are based, as well as competitor techniques to which the ones proposed in this Thesis can be compared.

Chapter 4

Object-based Refreshment Need Metrics

4.1 Introduction

Error resilience techniques, and especially error concealment, are usually seen as playing a role at the decoder side of the communication chain. This, however, is only partly true since the encoder and the bitstream syntax itself play an important role on what can be achieved at the decoder side in terms of error processing. In fact, even the most advanced decoders in terms of error concealment can do very little if the encoders do not correctly set the relevant encoding parameters, so that the decoder can somehow recover in the case that errors occur. And this is always true, even when there is no back-channel allowing the decoder to communicate with the encoder to explicitly inform or ask for specific error resilience actions.

Although there are several types of techniques that encoders may use to improve error resilience, it is largely recognized that coding refreshment plays a major role. This technique is especially useful for video encoders that rely on predictive (inter) coding to remove temporal redundancy because, in these conditions, the decoded quality can decay very rapidly due to error propagation if errors occur in the transmission or storage of the coded streams. Therefore, in order to avoid error propagation for too long a time, the encoder can use a coding refreshment scheme to refresh the decoding process and stop (spatial and temporal) error propagation. The refreshment process will certainly decrease coding efficiency, but it will significantly improve error resilience at the decoder side, increasing the overall subjective impact. In fact, error resilience always comes at the cost of some coding efficiency; the big question is then which type of error resilience technique(s) to use and how to adjust its settings in order to maximize the error resilience capabilities while minimizing the compression performance reduction.

In order to design an efficient intra coding refreshment scheme for an object-based coding scheme, it would be helpful for the encoder to have a method to determine which components

of the video data (shape and texture) of which objects should be refreshed and when, especially if based on an appropriate *refreshment need* measure. The refreshment need for a given visual entity, such as shape or a texture macroblock for macroblock-based coding schemes, would be a metric able to measure the necessity of refreshing a visual data entity according to some error resilience criteria and could be used by the encoder to decide if a given visual entity should be refreshed or not at a certain point in time. In the following, *shape refreshment need (SRN)* and *texture refreshment need (TRN)* metrics are proposed; later, in Chapter 5, these metrics will be integrated in a complete intra coding refreshment scheme, considering both shape and texture. Since these metrics characterize the video information in terms of refreshment need by analyzing the available data, this is a clear case of video processing to obtain metadata, which is then used for resilient coding (metadata for coding).

In order for the refreshment need metrics to be adequately defined, it is important to first understand what are the factors that influence the error resilience of data and, therefore, should be taken into account when deciding if it should be refreshed or not; these factors are analyzed in Section 4.2. Additionally, since the refreshment need metrics proposed here are to be used by an encoder to decide which data should be refreshed and when, it is also important to take into account any specific refreshment constraints that the targeted system may have. Since the proposed refreshment need metrics will be used in Chapter 5 to develop an intra coding refreshment technique that will be tested on an MPEG-4 video encoder¹, the constraints imposed by MPEG-4 are considered in Section 4.3. In Section 4.4, the refreshment need metrics themselves are proposed and, finally, Section 4.5 concludes the chapter.

4.2 Critical Refreshment Factors

Before defining refreshment need metrics to be used in a coding refreshment scheme, it is important to identify all the factors that influence error resilience and thus should influence the encoder refreshing strategy. Taking these critical factors into account when designing a refreshment strategy should allow reaching a better resilience performance. As can be seen in the following, critical refreshment factors are related to the characteristics of the content itself, to the coding scheme and, of course, to the channel characteristics:

- **Spatial complexity of video data** – This factor expresses the fact that not all types of content are equally difficult to conceal using spatial techniques, if errors occur. For example, video content including very abrupt spatial changes (in shape and/or texture) makes error concealment rather hard to perform with spatial concealment techniques and therefore higher refreshment rates may be needed.
- **Temporal stability of video data** – This factor expresses the fact that not all types of content are equally difficult to conceal using temporal techniques, if errors occur. For example, video content with fast motion makes error concealment rather hard to perform with temporal concealment techniques and therefore higher refreshment rates may be needed.

¹ MPEG-4 was adopted because it represents the only standard object-based video coding solution. However, this does not mean that the refreshment need metrics proposed here and the intra refreshment scheme proposed later in Chapter 5 are only suited for MPEG-4 video, but rather for all object-based video coding systems.

4. Object-based Refreshment Need Metrics

- **Bits spent on each object** – Objects that use more bits, because of their intrinsic characteristics or certain coding choices, are more likely to be hit by errors and therefore may need more refreshing.
- **Bits spent on different time instants** – For each object, the time instants that use more bits, because of their intrinsic characteristics or certain coding choices, are more likely to be hit by errors and therefore may need to accelerate the refreshment process.
- **Bits spent on different data** – Data classes, e.g. shape or texture, that use more bits are more likely to be hit by errors and therefore may need more refreshing. Shape refreshing is typically very important due to the very high negative subjective impact of shape errors; moreover MPEG-4 shape coding is particularly sensitive to errors, as it will be seen in the following. Therefore, if limited resources exist and only one type of refreshment can be applied (shape versus texture), shape refreshment will always get the preference.
- **Type and amount of channel errors** – Depending on the type and amount of channel errors that are going to hit the bits, more or less coding refreshment is needed.

The critical refreshment factors described above will be taken into account in Section 4.4 to develop a set of refreshment need metrics that can be used to build powerful object-based coding refreshment schemes.

4.3 MPEG-4 Refreshment Constraints

Besides the general constraints expressed by the critical factors listed above, the refreshment strategy has also to take into account the particular constraints related to the coding techniques and thus the coding syntax being used. In this chapter, the MPEG-4 Visual Core Profile [MPEG4-V] will be used. This profile allows for rectangular as well as arbitrarily shaped video objects and thus both shape and texture refreshment need to be considered. While MPEG-4 texture coding relies on the well-known block-based hybrid coding scheme, shape coding brings new challenges to intra refreshment.

At the time the MPEG-4 video coding scheme was developed, a lot of effort was invested to achieve a very efficient shape coding algorithm. This resulted in a natural reduction of its error resilience, which the encoder needs to compensate for by means of intra refreshment. The two major MPEG-4 shape coding properties that have a special impact in terms of error resilience are:

- **Inter coding of shape modes** – As opposed to what happens with texture coding, the shape coding mode of a block can be inter coded, which means it does not stand on its own but depends on previous modes (texture coding modes are always intra coded).
- **Decodability dependency on correctness of past information** – As opposed to what happens with texture coding, the prediction error for an inter coded shape block cannot be decoded without errors if the shape block used as prediction was not reconstructed without errors (for texture, the fact that the prediction is corrupted has no impact on the decodability of the prediction error).

These two MPEG-4 shape coding properties will strongly influence the intra refreshment strategy to be proposed and therefore deserve a more detailed explanation.

4.3.1 Inter Coding of Shape Modes

Inter coding of shape modes was introduced to provide an even higher level of shape coding efficiency. This shape coding tool is always used, except when the whole VOP shape is intra coded, in which case the shape coding modes will also be intra coded. This means that if the shape coding mode of the past shape block used as a predictor is corrupted, the coding mode of the current shape block will be erroneously decoded. And, therefore, so will the shape data itself.

Inter coding of shape modes significantly complicates the development of a shape intra refreshment solution also because shape information is very critical. Since the objective of intra refreshment is to fully eliminate the coding dependency on the past to avoid error propagation, one might be tempted to encode some of the critical shape blocks in intra mode, but this will not eliminate the dependency because the shape coding modes will still be inter coded. The only way to truly eliminate the shape coding dependency on the past is by encoding the complete VOP shape in intra mode, because this is the only way to get intra coded shape coding modes. This implies that true shape intra refreshment can only be made for the complete VOP shape all at once and not piece by piece, along time.

On the other hand, texture information does not have this problem and can still be intra refreshed by small fractions (groups of macroblocks) of the video object.

4.3.2 Decodability Dependency on Correctness of Past Information

The dependency of data decodability on the correctness of past information is related to the fact that context-based arithmetic encoding (CAE) is used to encode the shape data itself, which is represented by means of a binary alpha plane for the object bounding box². In MPEG-4 shape coding, for each shapel³ in a shape macroblock (also known as Binary Alpha Block or BAB) a context is computed based on a set of (spatially and temporally) surrounding shapels. The shapels used to compute the context are illustrated in Figure 4.1 (a) for shape intra coding and in Figure 4.1 (b) for shape inter coding.

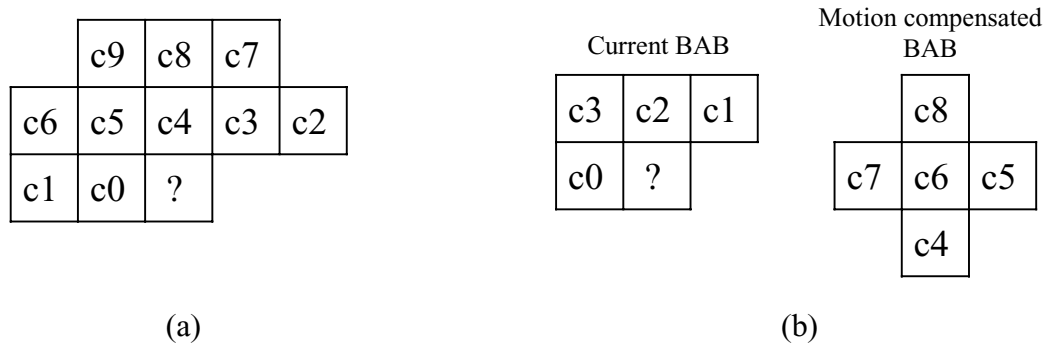


Figure 4.1 – Shapels used for context computation in the CAE algorithm (the question mark represents the shapel for which the context is being computed): (a) Intra mode; (b) Inter mode (the question mark is co-located with c6)

² The bounding box is the smallest matrix with an integer number of 16×16 blocks that can completely hold the video object.

³ A shapel is an element of the alpha plane; its name results from the contraction of the words *shape* and *element*. In a binary alpha plane, the shapels can be either opaque if they are inside the object or transparent if they are outside.

4. Object-based Refreshment Need Metrics

The context value, C , for a given shapel is computed using the following expression:

$$C = \sum_k c_k \cdot 2^k \quad (4.1)$$

where c_k is 0 for transparent shapels and 1 for opaque shapels. After computing the context, its value is used to index a probability table and retrieve a probability value, which is then used to drive the arithmetic encoder and generate the output binary arithmetic code (see [MPEG4-V]). If the inter coding mode is used to encode the shape, the context is computed using shapels from the current and also from the past time instants. This means that, if some of the past shape data was erroneously decoded, the context will also be wrong and the current arithmetic code will be erroneously decoded. In addition, since these arithmetic codes do not have a fixed length, synchronism will probably be lost and the following information up to the next synchronization marker will be wrong or lost.

This situation is clearly different from what happens with texture data decoding. In that case, it is only necessary to correctly decode the current (intra coded) coding mode and then the decoding of the texture update will be independent of the past. How the texture is reconstructed if the prediction is wrong is a totally different issue. Basically, the problem is that, in the case of the shape, an erroneous past block can make the whole current shape data useless, whereas in the case of the texture an erroneous past block will only propagate to the current blocks that use it as a prediction. This shows that errors in the shape data can propagate much faster than errors in the texture data, and therefore a lot of resources should be assigned to shape refreshment (also because shape is subjectively and functionally very important).

4.4 Proposal of Refreshment Need Metrics

The *refreshment need* for a given visual data entity, such as a shape or a texture macroblock, is a parameter that measures the necessity of refreshing this visual data entity according to some criteria. In order to compute this parameter, which will be used to determine the encoder decisions to refresh or not a given visual data entity, the factors described in Section 4.2 are considered. In the next sections, measures for the shape and texture refreshment needs will be proposed. These measures will then be integrated in the complete intra coding refreshment scheme proposed in Chapter 5.

In addition to the critical factors described in Section 4.2, the MPEG-4 refreshment constraints described in Section 4.3 also have to be taken into account since the target of the proposed refreshment scheme is an MPEG-4 encoder. As a consequence, in the proposed refreshment scheme the shape will be fully refreshed at a given time instant, whereas the texture can be refreshed by small fractions⁴ of the VOPs. This means that once in a while the whole VOP shape has to be intra encoded, while for each VOP only some texture macroblocks will be refreshed. As a consequence, the shape refreshment need measure is computed at the VOP level (for the whole shape) while the texture refreshment need measure is computed at the macroblock level.

⁴ If random access were a requirement, the texture would have to be fully refreshed at the same time as the shape, in order to create an access point to the bitstream, thus leading to a very bursty bitstream. However, as shall be explained later in Chapter 5, only applications that do not have random access requirements will be considered because this corresponds to the most interesting situation in terms of refreshment (more flexibility is allowed instead of an imposed periodic full refreshment solution).

4.4.1 Shape Refreshment Need

The shape refreshment need depends on several factors already described in Section 4.2. The parameter SRN_i measures the *shape refreshment need* for the i -th VOP, which corresponds to the contribution of the VOP in question to the *average shape refreshment need* parameter ($ASRN_i$). Therefore, the $ASRN_i$ parameter is simply defined as:

$$ASRN_i = \frac{1}{i - j + 1} \cdot \sum_{k=j}^i SRN_k \quad (4.2)$$

where i is the index corresponding to the VOP number, and j is the index of the most recent previous VOP with intra coded shape.

As for the SRN_i parameter, it must consider the most important factors impacting the *shape refreshment need* and thus the following definition is proposed:

$$SRN_i = SEV_i \times SCD_i = SEV_i \times f_{SCD}(\overline{SSCD}_i, \overline{TSCD}_i) \quad (4.3)$$

where SEV_i is the Shape Error Vulnerability (SEV), SCD_i is the Shape Concealment Difficulty (SCD), \overline{SSCD}_i is the scaled Spatial Shape Concealment Difficulty ($SSCD$), \overline{TSCD}_i is the scaled Temporal Shape Concealment Difficulty ($TSCD$), all for VOP i . Additionally, f_{SCD} is a function that combines the values of \overline{SSCD}_i and \overline{TSCD}_i (both with values between 0 and 1) into a single SCD value. By using scaled versions to the same range ([0,1]) of $SSCD$ and $TSCD$, it becomes easier to combine them into a final SCD metric, possibly as a weighted sum. If $SSCD$ and $TSCD$ had different ranges, extra constants would have to be introduced in the expression of SCD to compensate for the different ranges of the two concealment difficulty metrics.

SRN_i is defined as a product where the first factor is the shape vulnerability measuring the statistical exposure of the shape data to errors, already considering the used bitstream structure, and the second factor corresponds to the shape concealment difficulty which expresses how hard the shape is to recover using concealment techniques. A product is used because the dependency of the refreshment need on the shape vulnerability and the shape concealment difficulty should be such that when either one of them approaches zero, so should the shape refreshment need. For instance, if the considered shape is extremely easy to conceal (concealment difficulty approaching zero), the refreshment need should also approach zero independently of the shape vulnerability. Or, on the other hand, if the shape is not vulnerable to errors (because almost no bits were spent or the amount of errors is very small), the refreshment need should also approach zero independently of the concealment difficulty.

4.4.1.1 Defining Shape Error Vulnerability

It is proposed here that the shape error vulnerability, the first factor in Equation (4.3), be measured by the fraction of shape bits that will have to be discarded due to errors in VOP i . Therefore, by considering a random variable X^i representing the percentage of shape data bits in VOP i that has to be discarded, SEV_i can be defined as the expected value of this random variable as:

$$SEV_i \equiv E\{X^i\}. \quad (4.4)$$

4. Object-based Refreshment Need Metrics

Since in MPEG-4 the VOP data is divided into several independently decodable video packets (VP), the random variable X^i can be defined as a sum of random variables, each of them being equal to the percentage of the total shape data bits used for VOP i that has to be discarded in a given VP j , as:

$$X^i = \sum_{j=0}^{N_{vp}(i)-1} X_j^i \quad (4.5)$$

where i is the index corresponding to the VOP number, j is the index corresponding to the VP number within VOP i and $N_{vp}(i)$ is the number of VPs in VOP i .

Since the following expression holds even if the various random variables are not statistically independent [Carlson02]:

$$E\{X^i\} = \sum_{j=0}^{N_{vp}(i)-1} E\{X_j^i\} \quad (4.6)$$

it is possible to compute SEV_i by first determining the expected shape data bits percentage that will have to be discarded for each VP. This depends on the type and amount of errors, the number of bits used and how they are arranged in the VP, and the error detection capacity of the used decoder. Therefore, it is necessary to consider the structure of one VP in the used syntax; in the following, the MPEG-4 syntax will be considered since this is the only normative syntax and thus the most widely available.

In MPEG-4, the video information in a VP can be arranged according to two possible strategies, which are further detailed in [MPEG4-V]:

- **Combined mode** – All the information (shape, motion and texture) is multiplexed at the macroblock level, justifying the name of “combined mode”.
- **Data partitioning** – Corresponds to dividing the VP into two separate partitions where the first one holds the shape and motion data multiplexed at the macroblock level and the second one the texture data (also multiplexed at the macroblock level). This second strategy is called “data partitioning” and it is the preferred one when using error-prone channels since more sophisticated concealment solutions may be used, and therefore it will be the only one considered in the remainder of this chapter (and the Thesis).

Here, it will be conservatively assumed that if a given VP has at least one error in the first partition (including the shape and motion data) the whole shape data in the VP will be discarded; otherwise, no shape data bits will be discarded. This corresponds to the simplest possible recovery strategy at the decoder and it is assumed that all decoders can do it. Considering this, the expression for the expected fraction of the total shape data bits (used for VOP i) that will be discarded in VP j becomes:

$$E\{X_j^i\} = sbp(j, i) \cdot P\{\text{the shape data in VP } j \text{ of the VOP } i \text{ is to be discarded}\} \quad (4.7)$$

where i is the index corresponding to the VOP number, j is the index corresponding to the VP number within VOP i , $P\{event\}$ is the probability assigned to *event*, and $sbp(j, i)$ is the percentage of the total shape bits (used for VOP i) that are carried in the first partition of VP j .

Finally, by combining Equations (4.6) and (4.7), the expression for SEV_i becomes:

$$SEV_i = \sum_{j=0}^{N_{vp}(i)-1} sbp(j, i) \cdot P\{\text{the shape data of VP } j \text{ of VOP } i \text{ is to be discarded}\} \quad (4.8)$$

where $i, j, P\{\text{event}\}, sbp(j, i)$, and $N_{vp}(i)$ have the same meaning as before.

In order to proceed and determine the probability of the shape data in one video packet being discarded, it is necessary to consider the possible error patterns. Here, three types of errors are considered.

4.4.1.1.1 Uniform errors

These are uniformly distributed bit inversion errors and are simply characterized by the *bit error rate* (BER). This type of errors corresponds to the case of a fading channel in a circuit-switched wireless network where extensive interleaving is performed by the network terminals and the residual errors after channel decoding are simply being passed on to the video layer by the multiplex layer. Errors in the multiplex packet headers, which would correspond to entire packets being discarded, are not considered here.

In this case, the shape data in a given video packet is considered lost if at least one bit error exists in the first partition. Let P_e be the probability of a given bit being in error, then $1-P_e$ is the probability of this same bit not being in error. Therefore $(1-P_e)^{bs}$ is the probability of all the bits being correct, which means that $1-(1-P_e)^{bs}$ is the probability of at least one bit being in error. Therefore:

$$P\{\text{shape data of one VP is to be discarded}\} = 1 - (1 - P_e)^{bs} \quad (4.9)$$

where P_e is the bit error rate, and bs is the number of bits in the first partition, including the resync marker which indicates the beginning of the VP, the VP header, the shape data, the motion data and the marker which indicates the end of the first partition.

In the definition of bs , the resync marker is included because if it is corrupted, the decoder will not be able to identify the beginning of the VP, thus losing all the information in it (including the shape data). As for the other marker, it is also included because if it is corrupted by an error the decoder will not be able to correctly identify the end of the first partition and will go on assuming that it is still decoding it. This will cause the decoder to detect errors in this partition and throw the shape data away.

Therefore, Equation (4.8) becomes the following for uniform errors:

$$SEV_i = \sum_{j=0}^{N_{vp}(i)-1} sbp(j, i) \cdot [1 - (1 - P_e)^{bs}]. \quad (4.10)$$

4.4.1.1.2 Burst errors

These errors appear as bursts of bit inversions and are characterized by the *average BER*, the BER within the burst (*burst BER*) and the *average burst length*. This type of errors corresponds to a fading channel in a circuit-switched wireless network where no interleaving is used and the residual errors after channel decoding are simply being passed on to the video layer by the multiplex layer. As in the previous case, errors in the multiplex packet headers, which would correspond to entire packets being discarded, are not considered here.

These errors can be modeled as a discrete-time two-state Markov chain [Papoulis91] [Rabiner89], also known as Gilbert model [Gilbert60], which is basically a Markov process \mathbf{x}_n (where n represents the discrete time). The \mathbf{x}_n process is used to determine, at each time instant n , if there should be a bit inversion (i.e., an error in the bitstream at bit n) or not; an \mathbf{x}_n value of 1 corresponds to a bit inversion, while a value of 0 corresponds to no bit inversion.

4. Object-based Refreshment Need Metrics

These bit inversions are determined by considering that the Markov process has the two following states a_k defined as:

State a_1 : outside the burst

State a_2 : inside the burst

While the Markov process is in state a_1 (outside the burst), no errors can occur and therefore \mathbf{x}_n will always be 0. However, in state a_2 (inside the burst), uniformly distributed errors will occur with a given probability and therefore \mathbf{x}_n can either be 0 or 1.

This Markov chain is specified in terms of two different sets of probabilities: the *state probabilities* and the *transition probabilities*. While the state probabilities

$$p_k[n] = P\{\mathbf{x}_n = a_k\} \quad k = 1, 2 \quad (4.11)$$

represent the probability of the Markov process \mathbf{x}_n being in a given state a_k at discrete time instant n , the transition probabilities

$$\pi_{kl}[n_1, n_2] = P\{\mathbf{x}_{n_2} = a_l \mid \mathbf{x}_{n_1} = a_k\} \quad (4.12)$$

represent the probability of the Markov process \mathbf{x}_n being in state a_l at time instant n_2 , given that it was in state a_k at time instant n_1 .

Since the Markov process considered in the Gilbert model is homogeneous [Papoulis91], the transition probabilities depend only on the difference $m = n_2 - n_1$ and, to determine them, it is sufficient to know the one-step transition probabilities (i.e. the transition probabilities for two consecutive time instants)

$$\pi_{kl}[1] = P\{\mathbf{x}_{n+1} = a_l \mid \mathbf{x}_n = a_k\} \quad (4.13)$$

which are the elements of the one-step transition matrix

$$\Pi = \Pi[1] = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix}. \quad (4.14)$$

In addition, since the considered Markov process is also stationary, the state probabilities do not change over time and thus

$$p_k = p_k[n] \quad k = 1, 2. \quad (4.15)$$

In the Gilbert model [Gilbert60], the one-step transition probabilities are defined as:

$$\pi_{11} = 1 - \pi_{12}, \quad \pi_{12} = \frac{1}{\text{error free length}}, \quad \pi_{21} = \frac{1}{\text{burst length}} \quad \text{and} \quad \pi_{22} = 1 - \pi_{21} \quad (4.16)$$

where *error free length* is the average number of consecutive bits in state a_1 and *burst length* is the average number of consecutive bits in state a_2 .

As for the state probabilities, these are defined as:

$$p_1 = 1 - p_2 \quad (4.17)$$

and

$$p_2 = \frac{\pi_{12}}{\pi_{12} + \pi_{21}}. \quad (4.18)$$

Routine manipulation of Equations (4.16) through (4.18) leads to the following definitions of p_1 and p_2 :

$$p_1 = \frac{\text{error free length}}{\text{error free length} + \text{burst length}} \quad (4.19)$$

and

$$p_2 = \frac{\text{burst length}}{\text{error free length} + \text{burst length}}, \quad (4.20)$$

a rather obvious result since p_1 corresponds to the fraction of time spent in state a_1 and p_2 corresponds to the fraction of time spent in state a_2 .

Since errors only occur in state a_2 , and then just with probability *burst BER*, the *average BER* becomes:

$$\text{average BER} = \text{burst BER} \cdot p_2. \quad (4.21)$$

By replacing p_2 with its expression given in Equation (4.20), Equation (4.21) becomes:

$$\text{average BER} = \text{burst BER} \cdot \frac{\text{burst length}}{\text{error free length} + \text{burst length}}, \quad (4.22)$$

from which the error free length can be computed as:

$$\text{error free length} = \frac{\text{burst length} \cdot (\text{burst BER} - \text{average BER})}{\text{average BER}}, \quad (4.23)$$

which depends only on parameters that characterize the burst errors.

In this case, the shape data of the video packet is considered lost if at least one error burst starts within the first partition. Let P_t be the probability of a burst starting at a given bit, then $1-P_t$ is the probability of no burst starting at that bit. Therefore $(1-P_t)^{bs}$ is the probability of no bursts starting within the first partition of the VP, which means that $1-(1-P_t)^{bs}$ is the probability of at least one burst starting within the first partition of the VP. The probability P_t of a burst starting at a given bit can be easily computed because it is equal to the transition probability π_{12} determined above (hence the t subscript in P_t), which is the probability of the state changing from 1 to 2. Therefore, the probability of at least one error burst starting within the first partition of the VP is:

$$P\{\text{the shape data of the VP is to be discarded}\} = 1 - (1 - P_t)^{bs} \quad (4.24)$$

where P_t is the transition probability from state a_1 (outside the burst) to state a_2 (inside the burst), and bs is the number of bits in the first partition, including the resync marker at the beginning of the VP, the VP header, the shape data, the motion data and the marker which indicates the end of the first partition, for the reasons already explained for uniform errors.

Therefore, Equation (4.8) becomes the following for burst errors:

$$SEV_i = \sum_{j=0}^{N_p(i)-1} sbp(j,i) \cdot [1 - (1 - P_t)^{bs}], \quad (4.25)$$

which can be obtained from Equation (4.10) by replacing P_e with P_t .

4. Object-based Refreshment Need Metrics

4.4.1.1.3

Packet losses

These errors appear as deleted bits corresponding to the payload size of the used multiplex packets (not necessarily related to the VPs described above) and can be characterized by the packet loss rate and the payload size of the packets. This type of errors corresponds to two possible situations: i) a fading channel in a circuit-switched wireless network where the multiplex layer is unable to decode the corrupted packet headers, and hence the whole data in these multiplex packets is lost (i.e. the data is not passed on to the video layer), and ii) a packet-switched network where, due to network congestion, certain multiplex packets may not arrive when they are needed and therefore are considered discarded by the video layer; this might occur in an Internet application.

By assuming that the multiplex packets that can be lost have a fixed payload length S (this restriction will be relaxed later) and that the start of the first multiplex packet (starting inside the first partition) coincides with the first bit in the first partition of the VP, then the probability of at least one lost multiplex packet starting inside the first partition of the VP is:

$$P\{\text{shape data of one VP is to be discarded}\} = 1 - (1 - PLR)^{N_p(0)} \quad (4.26)$$

where PLR is the packet loss rate and $N_p(0)$ is the number of multiplex packets starting within the first partition of the VP assuming that the start of the first multiplex packet coincides with the first bit (bit 0) in the first partition.

However, the start of the first multiplex packet (starting inside the first partition) does not necessarily coincide with the first bit in the first partition of the considered VP; it can coincide with any of the first S bits. The number of bits spent in the previous VPs, which is variable, will determine this. If the start of the first multiplex packet (starting inside the partition) coincides with the second bit (bit 1) in the partition, then the same expression as before can be used by replacing $N_p(0)$ with $N_p(1)$. This reasoning can be repeated for the first S bits in the first partition. Assuming that the probability of the first multiplex packet (starting inside the partition) coinciding with any one of the first S bits is equal, then the probability of at least one lost packet starting inside the first partition of the VP is:

$$P\{\text{the shape data of the VP is to be discarded}\} = \frac{1}{S} \sum_{n=0}^{S-1} P_A(n) \quad (4.27)$$

with

$$P_A(n) = 1 - (1 - PLR)^{N_p(n)} \quad (4.28)$$

where PLR is the packet loss rate, S is the payload size of the multiplex packets, $P_A(n)$ is the probability of at least one lost multiplex packet starting within the first partition of the VP assuming that the start of the first multiplex packet coincides with bit n (with n varying between 0 and S), and $N_p(n)$ is the number of multiplex packets starting within the first partition of the VP assuming that the start of the first multiplex packet (starting inside the partition) coincides with bit n in the first partition. The resync marker at the beginning of the VP, the VP header, the shape data, the motion data and the marker which indicates the end of the first partition are all included in the first partition, for the reasons already explained for uniform errors.

If the payload size of the used packets is not fixed, then instead of S , the average packet payload size should be used.

Therefore, Equation (4.8) becomes the following for packet losses:

$$SEV_i = \sum_{j=0}^{N_{vp}(i)-1} \left\{ sbp(j, i) \cdot \frac{1}{S} \sum_{n=0}^{S-1} \left[1 - (1 - PLR)^{N_p(n)} \right] \right\}. \quad (4.29)$$

4.4.1.2 Defining Shape Concealment Difficulty

As for the shape concealment difficulty, the second factor in Equation (4.3), it measures the shape concealment difficulty, i.e., the difficulty to recover the shape if errors occur, which depends on the concealment techniques used at the receiver. Since these techniques can be spatial and/or temporal, the shape concealment difficulty in Equation (4.3) is expressed as a function f_{SCD} of \overline{SSCD}_i and \overline{TSCD}_i which has to be defined according to the used concealment solutions. Therefore, in order to have a precise definition of f_{SCD} , the encoder needs to know the type of concealment that is being used at the decoder. If this information is not available, the encoder will simply have to assume that the concealment is the simplest possible, in order to ensure proper working conditions for all decoder types. This way, the two relevant possible concealment solutions are:

- **Static shape concealment** – In this case, the shape concealment scheme used at the decoder is always the same and known beforehand, notably by the encoder; therefore, the function f_{SCD} that expresses the shape concealment difficulty is fixed and defined in advance. As an example, in a simple case where only temporal concealment is used at the decoder, the function f_{SCD} should not depend on \overline{SSCD}_i and should be simply equal to the \overline{TSCD}_i parameter.
- **Dynamic shape concealment** – In this case, the shape concealment scheme used at the decoder is adaptively chosen in order to get the best concealment results. Therefore, it is impossible for the encoder to know exactly what type of concealment scheme (spatial, temporal or a combination of both) is being used at a given time instant (since the decoder selection will also depend on the channel errors). A possible solution is to assume that the decoder is “intelligent” and always decides to use the concealment scheme that will yield the best results, which appears to be a sensible assumption. If the decoder were not “intelligent” and always decided to use the wrong concealment technique, it would be better not to use an adaptive approach and rather use a fixed concealment scheme. This way, the encoder can obtain a good estimate of the shape concealment difficulty by considering that the function f_{SCD} corresponds to the minimum of the spatial, temporal and spatio-temporal shape concealment difficulty. While the first two concealment difficulties are given by the \overline{SSCD}_i and \overline{TSCD}_i parameters, respectively, the spatio-temporal concealment difficulty, which is the concealment difficulty when a combination of both spatial and temporal techniques are used, will have to be estimated. This can be done by considering the values of the \overline{SSCD}_i and \overline{TSCD}_i parameters and how much improvement can be obtained by combining both types of techniques. Another solution is to assume that the decoder only implements a rather simple error concealment technique (e.g., only temporal concealment without motion compensation). Whenever nothing is known about the decoder concealment capabilities, this simple solution represents a conservative approach since it provides an intra refreshment solution without risks in the sense that sufficient refreshment will always be present; this assurance may have to be paid for with some decrease in terms of compression efficiency, sometimes not really necessary because the decoder is more “intelligent” than expected.

4. Object-based Refreshment Need Metrics

Since the *SSCD* and *TSCD* metrics are computed at the encoder to have an idea of how difficult the concealment will be at the decoder, it is important that the metric be applied to the locally reconstructed shape (encoded and locally decoded), instead of to the original, since this is the information that will be available at the decoder. For instance, if the metrics were applied to the original shape and *lossy* shape coding were used, the *SSCD* and *TSCD* values at the encoder would not precisely reflect the concealment difficulty at the decoder because the reconstructed shape at the decoder would be different from the original one used at the encoder.

4.4.1.2.1 Spatial Shape Concealment Difficulty

The \overline{SSCD}_i term in Equation (4.3) expresses how difficult it is to recover the shape in question by only using spatial information. In order to define an efficient *SSCD* measure, it is important to first identify which are the factors that influence the spatial shape concealment difficulty. The two major factors are:

- **Intrinsic shape complexity** - This factor expresses the complexity of the shape associated to its spatial variation. For instance, if a given fraction of the shape has been lost, it is much easier to conceal if the shape to recover is smooth rather than edgy with very fast changes along its contour. This can be seen in Figure 4.2, where two shapes with different intrinsic complexities are shown.
- **Shape grid fitting** - This factor relates to the way the shape fits in the coding grid imposed by the video codec. Although this factor is closely related to the shape size, the shape size itself is not a good difficulty measure; in fact, if a shape with half the resolution in both directions (and thus with a quarter of the size) is encoded using a grid of 8×8 blocks, the concealment difficulty will be about the same as with the original shape being encoded with a 16×16 grid, which shows that the size is not the key factor. This, of course, is true if the detail loss due to the resolution reduction of the contour is not taken into account, which appears to be a good approximation for shapes that are not extremely complex (i.e. less complex than the Cyclamen object shown in Figure 4.2). For these extremely complex shapes, when the resolution decreases so does their complexity because a lot of the detail is removed and the shapes become smoother, which means that they are also easier to conceal. For shapes that are not very complex, however, the resolution decrease does not influence much the shape complexity.



Figure 4.2 – Shapes with different intrinsic shape complexity: (a) low complexity; (b) high complexity

Taking into account the two factors above, it is proposed that the $SSCD_i$ parameter (i.e., $SSCD$ for the i -th VOP, before scaling) be defined as:

$$SSCD_i = ISC_i \times SGF_i \quad (4.30)$$

where, for VOP i , ISC_i is the Intrinsic Shape Complexity (ISC), which is an edginess measure of the contour, and SGF_i is the Spatial Grid Factor (SGF), which takes into account the used spatial resolution and the coding grid block dimension.

$SSCD_i$ is defined as a product of two factors where the first one represents the intrinsic shape complexity, and the second one is a scale factor (ranging between 0 and 1) used to express (the inverse of) the adjustment of the shape to the shape grid being used for coding (i.e. an SGF_i value of 0 corresponds to the best possible adjustment). This way, when SGF_i approaches zero, so will $SSCD_i$ and when SGF_i approaches 1, the $SSCD_i$ will tend to the ISC_i value; on the other hand, when ISC_i approaches zero, $SSCD_i$ will also approach zero, independently of the SGF_i value.

A) Measuring ISC

As for the ISC computation, which is intended to measure the intrinsic shape complexity, it could be based on any of several existing shape descriptors. However, since a standard-based solution is always preferred, it is proposed here to base it on the contour-based shape descriptor adopted by the MPEG-7 standard [Bober99][MPEG7-SW][MPEG7-V]; MPEG-7 is the new MPEG standard dealing with the description of audiovisual content to allow the quick and efficient searching, processing and filtering of various types of multimedia material [Manjunath02]. This contour-based shape descriptor is defined using the Curvature Scale Space (CSS) representation of the contour [Mokhtarian96][Mokhtarian96a][Mokhtarian99]. This type of use shows that the description tools provided by the MPEG-7 multimedia content description standard can also be useful while encoding video data, and not only for multimedia retrieval and filtering purposes.

The idea behind the CSS representation is that the contour can be represented by the set of points where the contour curvature changes and the curvature values between them; the curvature value is defined as the tangent angle variation per contour length. For each point in the contour, it is possible to compute the curvature of the contour at that point, based on the neighboring points [Haralick92]; a point whose two closest neighbors have different curvature values is considered a curvature change. In fact, not all curvature changes are needed to compute the CSS representation, but only those changes where the curvature goes from a positive to a negative value or vice-versa. When this happens, the curvature values have necessarily to go through zero and therefore these changes are called zero-crossings of the curvature, as illustrated in Figure 4.3. As for the average curvature between two of these zero-crossings, it basically corresponds to the angle difference between the tangents to the contour at these two points divided by the arc length joining these two points, which can be expressed as:

$$C_{av} = \frac{\Delta\varphi}{\Delta u} \quad (4.31)$$

where C_{av} is the average curvature value between the two points in question, $\Delta\varphi$ is the angle difference between the tangents to the contour at the points, and Δu is the arc length between the two points.

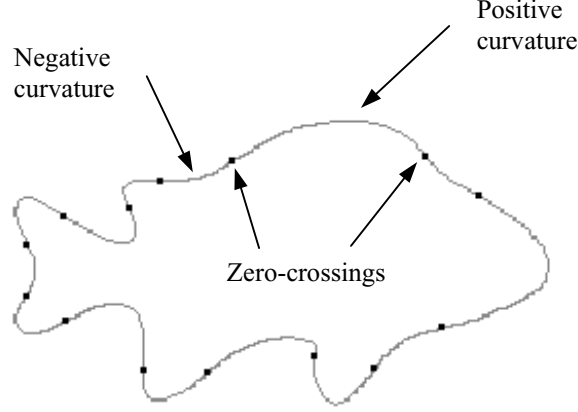


Figure 4.3 – Zero-crossings of the curvature

Since the number of zero-crossings of the curvature and the average curvature values between them have shown, in some preliminary experiments, to be a good indication of the spatial concealment difficulty, it is proposed here to define an intrinsic shape complexity measure, to be used as part of the shape concealment difficulty measure, based on the MPEG-7 CSS descriptor. One region-based shape descriptor using Zernike moments [Kim99][Kim00], which was initially selected to be included in the MPEG-7 standard but was later superseded by the Angular Radial Transform (ART) [Kim99a][MPEG7-V] due to its improved performance, was also tried as a complexity measure. However, the results obtained showed a very poor discrimination capability between shapes with very different complexities.

In order to generate the CSS representation of a given contour, the considered contour has to be a closed planar curve (i.e. a non-self-intersecting contour), with the following parametric representation:

$$\Gamma = \{(x(u), y(u)) | u \in [0, 1]\} \quad (4.32)$$

where u is the normalized arc length parameter, varying between 0 and 1, and $x(u)$ and $y(u)$ are the parametric coordinate functions.

The idea is to compute the convolution of the parametric coordinate functions of the curve Γ with a 1D Gaussian kernel with a progressively larger width σ , which is equivalent to low-pass filtering the original contour with a filter with a progressively lower bandwidth. In the case of a discrete contour, with $x(u)$ and $y(u)$ sampled at equidistant values of u , this can be implemented by repetitive application of the (normative) low-pass filter with kernel (0.25, 0.5, 0.25) as in [MPEG7-SW]. In order to guarantee that $x(u)$ and $y(u)$ are sampled at equidistant values of u , the original contour may have to be resampled at different values of u . After each pass of the low-pass filter, all the curvature zero-crossings are located. This is simply done by computing the curvature for all the points in the contour and determining where the contour curvature goes from a positive value to a negative one, and vice-versa. This means that in a curve segment between two zero-crossings, the curvature will be either positive or negative for all the points. The curve Γ is successively low-pass filtered until it becomes completely convex (i.e. there are no curvature zero-crossings). Finally, the CSS representation of the contour (CSS image) basically corresponds to a plot where the arc length parameter value (relatively to an arbitrary starting point) is the x-coordinate and the number of low-pass filter passes (or iterations) is the y-coordinate.

This CSS generation process is illustrated in Figure 4.4, which shows a contour at two different stages of the smoothing process (after 20 and 80 passes of the low-pass filter). Next to the contours is the CSS image obtained from the contour evolution, where the contour curvature zero-crossings (A through H) and the corresponding points on the CSS image are marked.

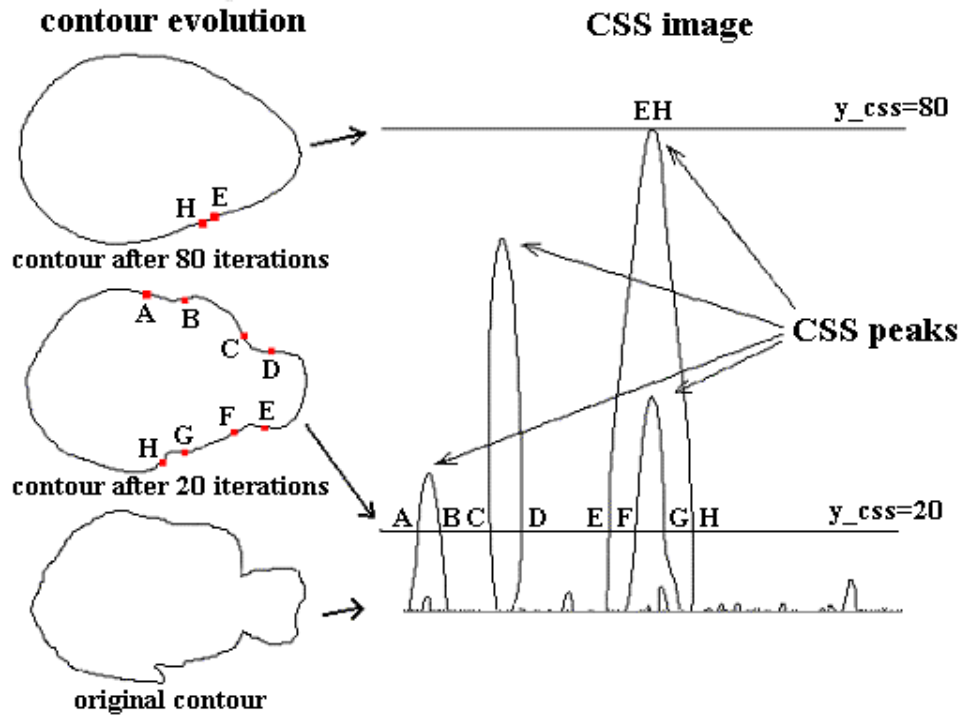
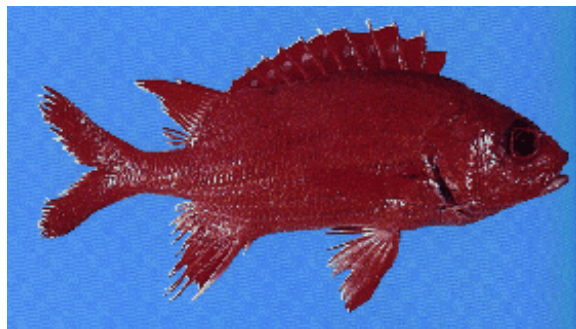


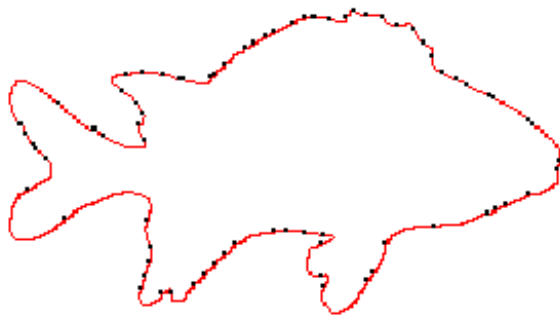
Figure 4.4 – CSS image formation [MPEG7-V]

In order to better clarify the explanation given above, a detailed example is given in the following. For this, consider the image in Figure 4.5 (a), which basically corresponds to a fish object; the rest of Figure 4.5 shows the intermediate steps that are taken to generate the CSS representation of the contour for this fish object. In Figure 4.5 (b), (d), (f), (h) and (j), the contour of the fish is shown with progressive levels of smoothing (using the filter described above) where the zero-crossings have been represented by black dots, while Figure 4.5 (c), (e), (g) and (i) correspond to the progressive formation of the final CSS image, shown in Figure 4.5 (k). The more interested reader should refer to [SQUID] for an animated demonstration of the CSS image formation process.



(a)

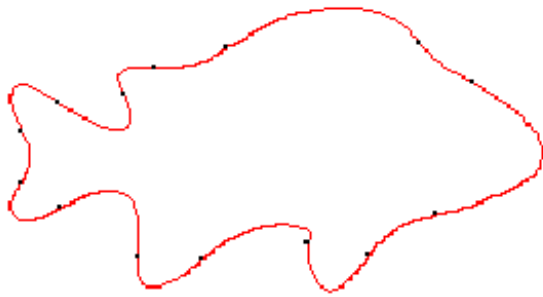
4. Object-based Refreshment Need Metrics



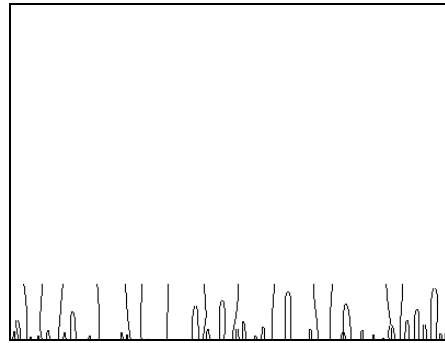
(b)



(c)



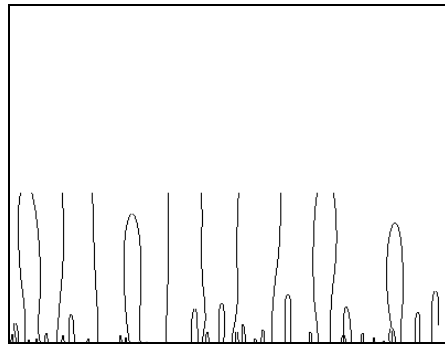
(d)



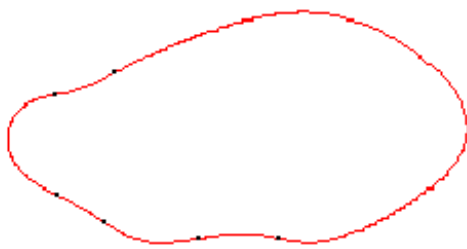
(e)



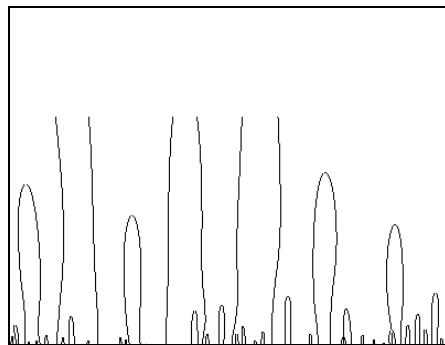
(f)



(g)



(h)



(i)

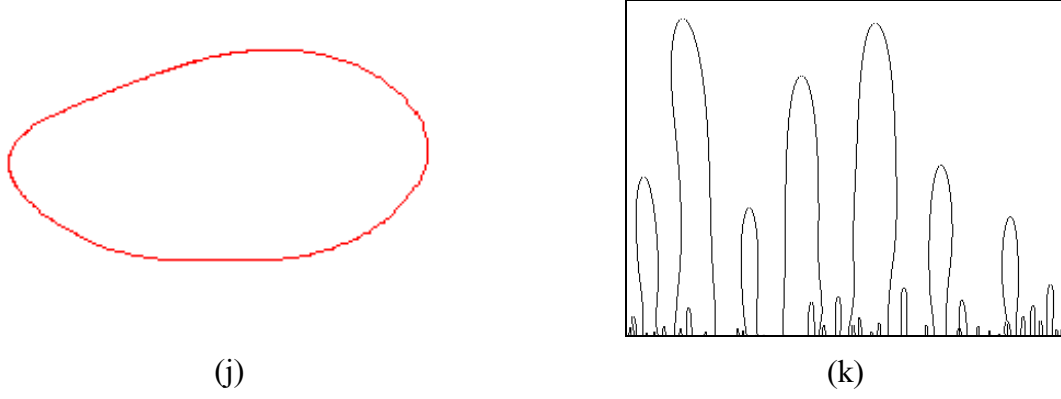


Figure 4.5 – Fish object: (a) original image; (b), (d), (f), (h) and (j) contours with progressive amounts of low-pass filtering (after 3, 13, 30, 45 and 60 passes of the low-pass filter, respectively); (c), (e), (g), (i) and (k) corresponding progressive formation of the CSS representation [SQUID]

It should be noted that as low-pass filtering smoothes the contour, the zero-crossings will group two by two, approaching each other until they merge and finally disappear, forming a CSS peak. Each zero-crossing does not necessarily group with an adjacent zero-crossing, which means that at the end smaller peaks can exist inside larger ones. These smaller peaks, which visually correspond to small ripples in the contour, are due to contour sections delimited by two zero-crossing that are close together and therefore, even if the curvature value between them is large, the small ripple in the contour disappears after the first few passes of the smoothing filter. Adding noise to a relatively smooth contour will have this same effect, as shown in [Mokhtarian96]. On the other hand, the highest peaks correspond to sections of the contour delimited by two contour zero-crossings further apart than in the previous case and with a higher average curvature value between them.

Since the number of these contour sections and their curvature values is related to the shape complexity, it is proposed that the ISC_i parameter be computed by:

$$ISC_i = \sum_{j=1}^N Peak(i, j) \quad (4.33)$$

where i is the index corresponding to the VOP number, j is the index corresponding to the peak number of the CSS representation, and $Peak(i, j)$ is the amplitude of the j -th peak of the CSS representation of the shape of the i -th VOP.

As in the MPEG-7 standard [MPEG7-V], the amplitude of a given peak is determined based on the CSS image, as follows:

$$Peak(i, j) = 3.8 \cdot \left(\frac{y_{css}(i, j)}{num_samples^2(i)} \right)^{0.6} \quad (4.34)$$

where $y_{css}(i, j)$ is the number of iterations that are necessary to create the j -th peak in the CSS image of the i -th VOP, and $num_samples(i)$ is the number of equidistant samples that are used to represent the contour of the i -th VOP; in this case, the number of samples used to represent the contour is the same as the number of points in the original contour, which ensures an accurate representation of the contour.

Additionally, in the MPEG-7 Reference Software [MPEG7-SW] all the peaks that do not amount to, at least, 5% of the highest peak in the CSS image of the considered contour are discarded. This makes sense for retrieval applications because the smaller peaks correspond

4. Object-based Refreshment Need Metrics

to very fine detail in the shape of an object, which might make more difficult a search for similar objects in a database. Here, however, the very fine detail is being (losslessly) encoded and, therefore, it makes sense to consider even the smaller peaks.

The above explanation was given for objects that have a single contour, i.e. with only one connected component and no holes. However, it is possible for objects to have more than one contour⁵. This happens for objects whose alpha planes have non-connected regions and/or holes, such as the Cyclamen object in Figure 4.6. As can be seen, when the contour is extracted from the alpha plane, multiple contours (in this case, 7) appear due to the 6 holes. Therefore, the CSS representation has to be computed for all the individual contours; in that case, the total *ISC* is obtained by adding the peak amplitudes for all the contours in question.



Figure 4.6 – (a) Video object with (6) holes and (b) its multiple contours (7)

B) Measuring SGF

In order to define *SGF*, it is important to acknowledge that the use of a block-based coding grid implies the existence of three types of shape blocks in the alpha plane: transparent blocks (all the shapels are transparent), opaque blocks (all the shapels are opaque) and border blocks (block with both opaque and transparent shapels). These different types of shape blocks are illustrated in Figure 4.7.

⁵ A contour is here understood as a closed planar curve that defines the limits of one connected component or region of an object. If an object has any holes or more than one connected component, then several such contours are needed to define the boundaries of the object.

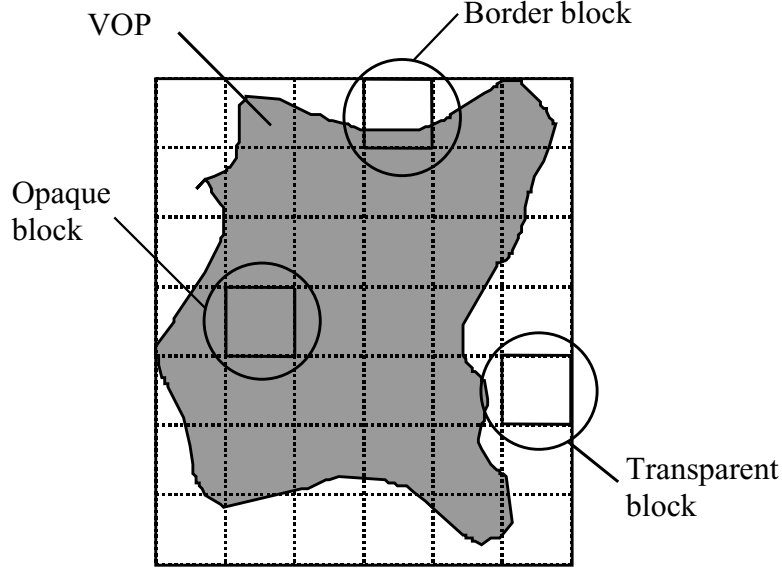


Figure 4.7 – Types of shape blocks in a VOP

Of these three cases, the border blocks correspond to the hard case in terms of shape concealment since an arbitrary opaque-transparent frontier has to be defined. By comparison, the concealment of the two other cases is rather trivial. Therefore, when considering two shapes with the same *ISC* values, the one with a larger percentage of border blocks is considered harder to conceal. Similarly, if two shapes with the same *ISC* values and the same percentage of border blocks are considered, the one with the higher ratio of opaque shapels inside border blocks versus the total amount of opaque shapels should be considered harder to conceal. This is so, because if border blocks in the object that has a larger percentage of opaque shapels in border blocks are lost, a larger percentage of the whole shape will be affected. Therefore, the proposed definition for the parameter SGF_i is:

$$SGF_i = \sqrt{(A \cdot B)_i} \quad (4.35)$$

with

$$A = \frac{\text{number of border blocks}}{\text{total number of border and opaque blocks}} \quad (4.36)$$

and

$$B = \frac{\text{number of opaque shapels in border blocks}}{\text{total number of opaque shapels}}. \quad (4.37)$$

The two factors A and B express the effects described above in terms of spatial shape concealment. Since these two factors vary between 0 and 1, so will their product. However, these factors are not completely independent and therefore their simple multiplication would yield an artificially small value. To avoid this undesirable effect, the square root is used. In fact, it should be noticed that by applying the square root to the product, SGF_i becomes the geometric average of the two factors described above.

4. Object-based Refreshment Need Metrics

C) SSCD Performance

In order to illustrate the performance of the proposed *SSCD* measure, some results will be given for the reconstructed shapes (encoded and decoded) with different complexities shown in Figure 4.8. In order to create a worst-case situation in terms of spatial detail, the lossless mode was used in the encoding (i.e. the reconstructed shape is exactly the same as the original). The shapes in Figure 4.8 correspond to the first VOP of the object sequences Cyclamen, Tennis Player, Weather Girl and Logo (from the News sequence).

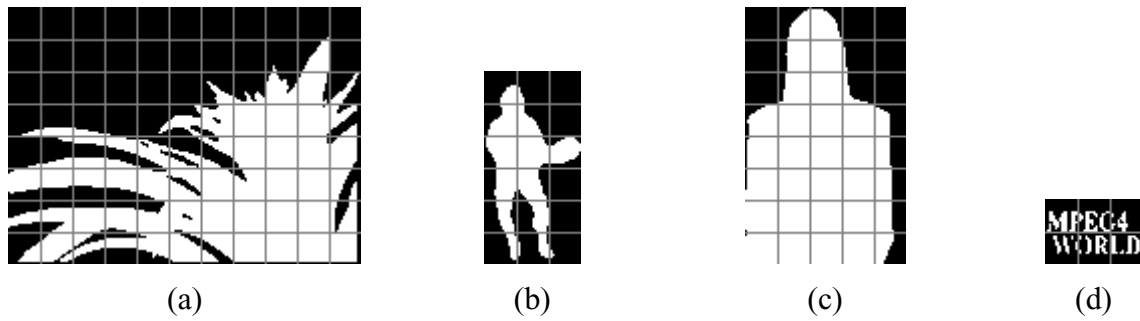


Figure 4.8 – Shapes with different spatial concealment difficulty in their bounding boxes (QCIF) with the coding grid overlaid (16x16 shapels): (a) Cyclamen; (b) Tennis Player; (c) Weather Girl; (d) Logo

Based on the knowledge that the shape encoding scheme is block-based (like the MPEG-4 shape coding algorithm) and that the decoder will use spatial shape concealment techniques, any error concealment expert can order these shapes (displayed in the same QCIF resolution) according to their spatial concealment difficulty, starting with the Logo object as the hardest to conceal followed by Cyclamen, Tennis Player and, finally, Weather Girl. This order should hopefully be reflected by the *SSCD* values corresponding to the proposed *SSCD* metric, shown in Table 4.1, indicating that the proposed objective metric reaches the target to automatically express how difficult it is to conceal a given object with spatial concealment techniques. In addition, the *SSCD* values should also increase as the resolution goes down because the objects will become increasingly harder to conceal, notably because the percentage of border blocks will increase.

Table 4.1 – *SSCD* measures for various shapes at QCIF (176x144) and CIF (352x288) resolutions

Video object	Resolution	ISC	SGF	SSCD
Cyclamen	QCIF	7.045	0.816	5.748
	CIF	8.981	0.624	5.607
Tennis Player	QCIF	1.690	1.000	1.690
	CIF	1.683	0.730	1.229
Weather Girl	QCIF	0.996	0.461	0.459
	CIF	1.157	0.246	0.285
Logo	QCIF	10.844	1.000	10.844
	CIF	13.504	1.000	13.504

The analysis of the results in Table 4.1 leads to the conclusion that the proposed *SSCD* measure behaves rather well in terms of expressing the spatial concealment difficulty. For instance, the results correctly reflect the different concealment difficulties of the various objects, by classifying the Logo object as the hardest to conceal with spatial concealment techniques, followed by the Cyclamen, Tennis Player and, finally, Weather Girl objects. In addition, the *SSCD* values tend to increase as the resolution goes down; this was expected because more direction changes in the contour exist per area unit for the lower resolutions. The only exception here is the Logo object, whose *SSCD* value increases with the resolution. This can be explained by considering two facts. First of all, the *SGF* is 1 for both QCIF and CIF resolutions due to the fact that the object is so complex that no opaque blocks exist, only border blocks. And second, since this object is extremely complex, when the resolution decreases a lot of detail is lost, meaning that the object actually becomes smoother and, therefore, easier to conceal. For the Cyclamen object, the *ISC* value also increases with the resolution, but since the *SGF* value decreases more, the *SSCD* will also decrease when the resolution increases. As for the remaining objects, they are much smoother and therefore the detail loss due to resolution reduction is not as strong as for the Logo and Cyclamen objects and the *ISC* values are practically constant for both resolutions. In this case, the variation of the *SSCD* values is mainly dictated by the *SGF* variation.

Since the *SSCD* values in Table 4.1 are not in the $[0,1]$ range and Equation (4.3) requires them to be between 0 and 1, a scaling operation has to be performed. To scale the *SSCD* values, a possibility would be to divide them by their theoretical maximum. However, since this maximum is typically never reached for real situations, only a small fraction of the $[0,1]$ range would end up being used. Therefore, an experimentally determined value should be used instead for the scaling process. Thus, the results for the Logo in the News sequence, which is very likely the hardest object to conceal with spatial techniques in the whole MPEG-4 test set, will be used as reference for the scaling process. In terms of spatial concealment difficulty, the worst-case usually corresponds to encoding the object with the lossless mode, using the lowest resolution (typically QCIF). However, for the Logo object, due to its high complexity, the highest *SSCD* value occurs for the CIF resolution. Since the *SSCD* value for this object, in these conditions, is 13.504, the *SSCD* values will be truncated at 14 and scaled to be in the $[0,1]$ range, assuming that this is a high enough *SSCD* value. Therefore, the scaled *SSCD* (\overline{SSCD}) shall be defined as:

$$\overline{SSCD} = \begin{cases} \frac{SSCD}{14}, & 0 \leq SSCD < 14 \\ 1, & SSCD \geq 14 \end{cases}. \quad (4.38)$$

4.4.1.2.2

Temporal Shape Concealment Difficulty

As for the \overline{TSCD}_i term in Equation (4.3), it expresses how difficult it is to conceal the shape in question by using only temporal information. In order to define the *TSCD* measure, it is important to first identify which are the factors that influence the temporal shape concealment difficulty. The two major factors are:

- **Temporal shape stability** – This factor expresses the amount of shape variation from one time instant to the next. For instance, if a given fraction of the shape has been lost, it is much easier to conceal if the shape to recover has changed very little with respect to the previous time instant.

4. Object-based Refreshment Need Metrics

- **Shape grid fitting** – This factor relates to the way the shape fits in the coding grid imposed by the video codec. The same comments that were made in the previous section dedicated to the *SSCD* measure also apply here.

Taking into account the two factors above, it is proposed that the $TSCD_i$ parameter (i.e., $TSCD$ for the i -th VOP, before scaling) be defined as:

$$TSCD_i = TSS_i \times TGF_i \quad (4.39)$$

where, for VOP i , TSS_i is the Temporal Shape Stability (TSS), which basically measures the amount of shape change from one time instant to the next, and TGF_i is the Temporal Grid Factor (TGF), taking into account the used spatial resolution and the coding grid block dimension.

$TSCD_i$ is defined as a product of two factors where the first one represents the temporal shape stability, and the second one is a scale factor (ranging between 0 and 1) used to adjust the temporal shape stability to the shape grid being used for coding. This way, when TGF_i approaches zero, so will $TSCD_i$ and when TGF_i approaches 1, the $TSCD_i$ will tend to the TSS_i value; on the other hand, when TSS_i approaches zero, $TSCD_i$ will also approach zero, independently of the TGF_i value.

A) Measuring TSS

The TSS metric basically expresses the amount of change in the shape from one time instant to the next. Therefore, a good metric seemed to be the number of different shapels in the current and previous alpha planes for a certain object. However, this metric measures only the absolute change and the concealment difficulty is also related to the relative amount of change. For instance, if two different objects have the same number of different shapels from the previous time instant to the current one, then the smaller object should be considered harder to conceal because a larger percentage of the whole shape has changed and will be more difficult to conceal in case of error. Therefore, the number of changed shapels relative to the size of the object in question appears to be a good metric, resulting for TSS_i :

$$TSS_i = \left(\frac{\text{number of different shapels between current and previous VOP}}{\text{number of opaque shapels in current VOP}} \right)_i. \quad (4.40)$$

B) Measuring TGF

The TGF metric expresses the effect of the used spatial coding grid on the temporal shape concealment difficulty, which is not taken into account by the TSS metric. Therefore, in order to define TGF , it is important to acknowledge that the use of a block-based coding grid implies the existence of two types of shape blocks in terms of temporal stability: stable blocks (blocks whose shape does not change) and unstable blocks (blocks whose shape changes). Of these two cases, the unstable blocks correspond to the hard case in terms of shape concealment because the co-located block in the previous time instant cannot be simply copied. By comparison, the concealment of the stable blocks is rather trivial. Therefore, when considering two shapes with the same TSS values, the one with a larger percentage of unstable blocks is considered harder to conceal. Similarly, if two shapes with the same TSS values and the same percentage of unstable blocks are considered, the one with the higher ratio of opaque shapels inside unstable blocks versus the total amount of opaque shapels should be considered harder to conceal. This is so, because if unstable blocks in the object that has a larger percentage of opaque shapels in unstable blocks are lost, a larger percentage

of the whole shape will be affected. Therefore, the proposed definition for the parameter TGF_i is:

$$TGF_i = \sqrt{(C \cdot D)_i} \quad (4.41)$$

with

$$C = \frac{\text{number of unstable blocks}}{\text{total number of border and opaque blocks}} \quad (4.42)$$

and

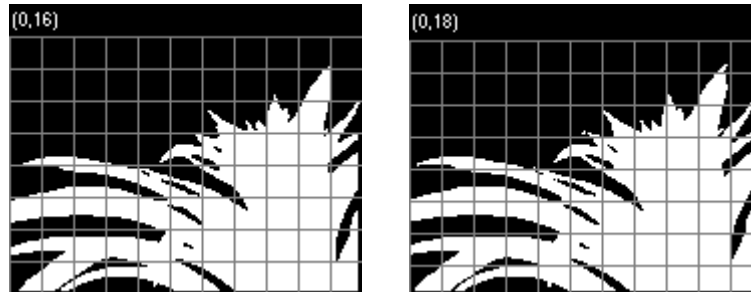
$$D = \frac{\text{number of opaque shapels in unstable blocks}}{\text{total number of opaque shapels}}. \quad (4.43)$$

The two factors C and D express the effect described above in terms of temporal shape concealment. As in the SGF_i definition, the square root is used to avoid that the TGF_i parameter becomes artificially small, as already explained.

C) TSCD Performance

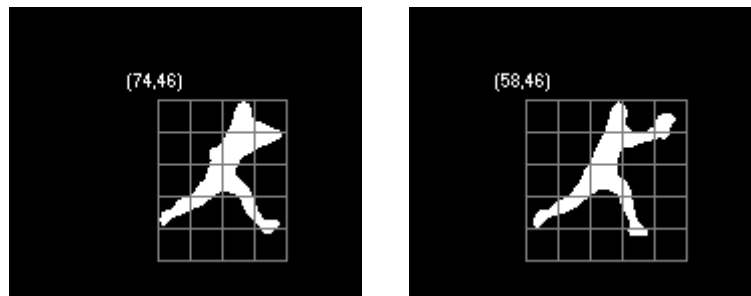
In order to show the performance of the $TSCD$ metric, results are given below for six objects from five different sequences with different spatial resolution (shown in Figure 4.9 for QCIF). In order to create a worst-case situation in terms of spatial detail, the lossless mode was used in the encoding (i.e. the reconstructed shape is exactly the same as the original). The first four object sequences are the same that were already used to illustrate the $SSCD$ performance.

The Cyclamen sequence corresponds to a long pan to the right of a plant, combined with a tilt down first and then a tilt up. The Stefan sequence is the fastest sequence in the MPEG-4 test set and focuses on a tennis player moving in all directions. The sequence Weather is a very slow sequence with a girl presenting the weather forecast. Finally, the Logo object in the News sequence is a still text object. As for the additional two objects, they correspond to the two boats in the Coastguard sequence. In this sequence, the camera is following a small boat moving to the left until a larger one comes into the scene. At this point, there is a fast tilt up and the camera starts following the large boat moving to the right. For more information on these sequences, the reader may refer to Annex A of this Thesis.

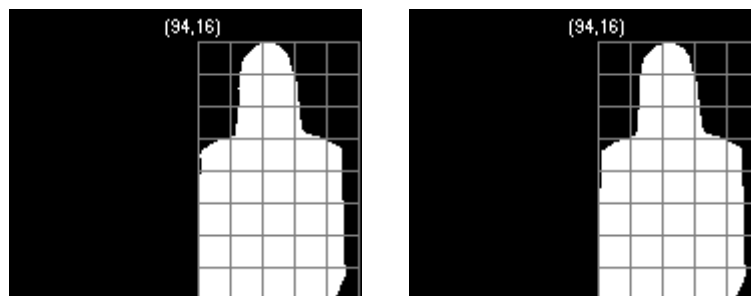


(a)

4. Object-based Refreshment Need Metrics



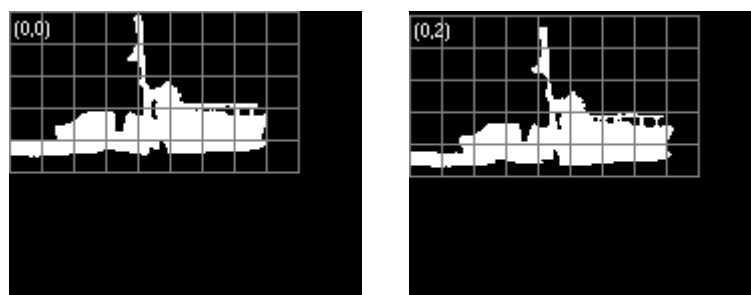
(b)



(c)



(d)



(e)

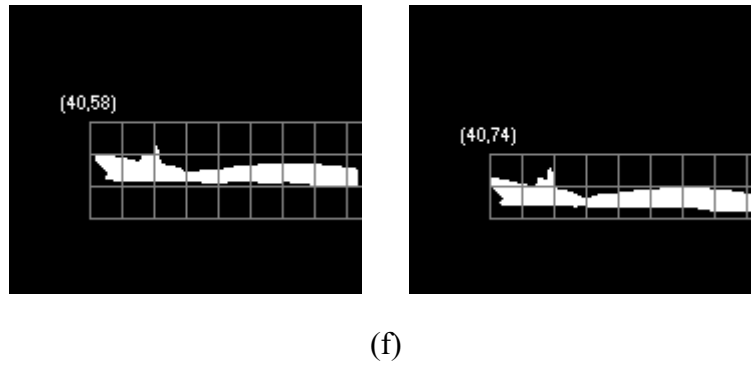


Figure 4.9 – Shapes with different temporal concealment difficulties in their bounding boxes (QCIF) with the coding grid overlaid (16x16 shapes): (a) frames 0 and 3 of the Cyclamen object; (b) frames 180 and 182 of the Tennis Player object; (c) frames 0 and 3 of the Weather Girl object; (d) frames 0 and 3 of the Logo object; (e) frames 66 and 69 of the Large Boat object (Coastguard sequence); (f) frames 69 and 72 of the Small Boat object (Coastguard sequence)

Similarly to what was said for the SSCD case, assuming that temporal concealment techniques are being used, any error concealment expert can rank the objects according to their temporal shape concealment difficulty, starting with the Small Boat object as the hardest to conceal followed by Tennis Player, Large Boat, Cyclamen, Weather Girl and, finally, Logo. This ranking should hopefully be reflected by the TSCD values, shown in Table 4.2, indicating that the proposed objective metric reaches the target to automatically express how difficult it is to conceal a given object shape with temporal concealment techniques. In addition, the TSCD values should also increase as the resolution goes down because the objects will become increasingly harder to conceal, notably because the percentage of unstable blocks will increase.

Table 4.2 –TSCD measures for the various shapes shown in Figure 4.9

Video object	Resolution	TSS	TGF	TSCD
Cyclamen	QCIF	0.1435	0.7335	0.1053
	CIF	0.1448	0.5985	0.0867
Tennis Player	QCIF	1.3761	1.0000	1.3761
	CIF	1.3580	1.0000	1.3580
Weather Girl	QCIF	0.0059	0.2862	0.0017
	CIF	0.0064	0.2060	0.0013
Logo	QCIF	0.0000	0.0000	0.0000
	CIF	0.0000	0.0000	0.0000
Large Boat	QCIF	0.6798	1.0000	0.6798
	CIF	0.6615	0.9184	0.6075
Small Boat	QCIF	1.9262	1.0000	1.9262
	CIF	1.9368	1.0000	1.9368

4. Object-based Refreshment Need Metrics

The analysis of the results in Table 4.2 leads to the conclusion that the proposed *TSCD* measure behaves rather well in terms of expressing the temporal shape concealment difficulty. For instance, the Logo object which is still along the whole sequence and therefore very easy to conceal with temporal techniques by just copying it from the previous time instant has a *TSCD* value of zero. The Weather Girl object, which is also relatively easy to conceal with temporal techniques because it hardly moves, comes next with slightly higher *TSCD* values. As for the Cyclamen object, it is significantly harder to conceal than the two previous ones because significant motion exists, even if not very fast due to the size of the object; this explains the *TSCD* values, which are much higher than for the two previous objects. In this case, good results can still be achieved if a more advanced temporal concealment technique, e.g. relying on motion information, is used because the motion is coherent and not very fast. And, finally, come the Large Boat, Tennis Player and Small Boat, which are even harder to conceal, with the Small Boat object being the hardest because of the very fast motion, which is correctly expressed by the highest *TSCD* value.

As for the dependency of the results with the spatial resolution, the expected behavior is also confirmed, since the *TSCD* values tend to decrease as the resolution increases. However, for the Tennis Player and Small Boat objects the *TSCD* values remain practically unchanged. This happens because these objects are such that when the resolution changes from QCIF to CIF the *TGF* parameter remains equal to 1 (all the blocks are unstable). For the Logo object, the *TGF* parameter remains equal to 0 when the resolution changes from QCIF to CIF (all the blocks are stable), but even if it did not, the *TSCD* value would remain fixed at zero because this object does not change in time and therefore the *TSS* parameter is zero. For the remaining objects (Cyclamen, Weather Girl and Large Boat), the *TGF* parameter decreases when the resolution is increased from QCIF to CIF, which explains the decrease in the *TSCD* values.

Considering that the Stefan sequence is the fastest changing sequence among the MPEG-4 test sequences and that in terms of temporal concealment, the worst-case corresponds to encoding the Tennis Player object with the lossless mode, using the lowest acceptable frame rate for viewing (which is about 15 fps for this sequence), the *TSCD* values for this object in QCIF and CIF resolutions have been plotted in Figure 4.10.

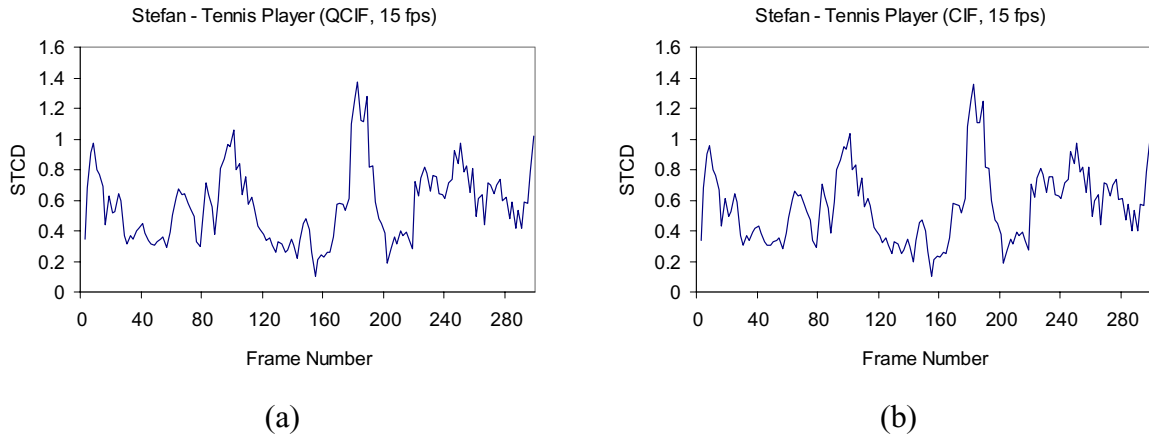


Figure 4.10 – *TSCD* for the Tennis Player object: (a) QCIF at 15 fps; (b) CIF at 15 fps

As expected, the obtained *TSCD* values are globally higher than for the other test sequences, throughout time (i.e. close and sometimes even above 1). However, by considering the Small Boat object also encoded with the lossless mode and using the lowest acceptable frame rate for viewing this sequence (about 10 fps since it is globally slower than Stefan), it can be seen

in Figure 4.11 that even higher $TSCD$ values can be achieved even if only for a short period of time (when the camera tilts up very fast, just before it starts following the Large Boat).

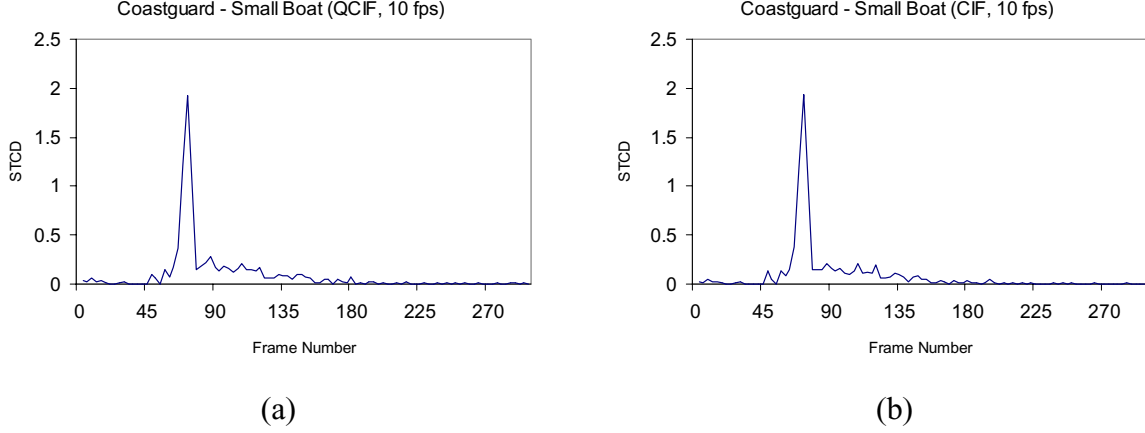


Figure 4.11 – $TSCD$ for the Small Boat object: (a) QCIF at 10 fps; (b) CIF at 10 fps

As expected, both resolutions have the same type of $TSCD$ temporal evolution. For instance, the maximum value for the QCIF resolution is 1.3761 for Tennis Player (for frame 182) and 1.9262 for Small Boat (for frame 72), whereas for the CIF resolution it is 1.3580 for Tennis Player (also for frame 182) and 1.9368 for Small Boat (also for frame 72). Considering that these are representative $TSCD$ values of a very critical object, in terms of temporal concealment, they will be used as reference for the scaling of the $TSCD$ measure. Therefore, the scaled $TSCD$ (\overline{TSCD}) will be defined as:

$$\overline{TSCD} = \begin{cases} \frac{TSCD}{2}, & 0 \leq TSCD < 2 \\ 1, & TSCD \geq 2 \end{cases}. \quad (4.44)$$

4.4.1.3 Evaluating the Shape Refreshment Need Metric Performance

To show the performance of the shape refreshment need metric proposed in this paper, SRN_i of Equation (4.3), results are given in Table 4.3 for the object shown in Figure 4.9 (b) and the corresponding Background, which were extracted from the Stefan sequence (QCIF at 192 kbps) and in Table 4.4 for the object shown in Figure 4.9 (c) and the corresponding background (i.e., Weather Map), which were extracted from the Weather sequence (QCIF at 56 kbps). These results were obtained for 2 and 4 VPs per VOP and for two different burst error patterns: one with an average BER of 10^{-2} (Burst 1) and the other with an average BER of 10^{-3} (Burst 2). In both cases, the BER inside the burst is 0.5 and the burst length is 10 ms. In addition, the encoder will consider that temporal concealment techniques are used at the decoder, which means that the f_{SCD} function is simply equal to \overline{TSCD} .

4. Object-based Refreshment Need Metrics

Table 4.3 – Shape refreshment need (SRN) values for the Stefan sequence objects, in addition to their shape error vulnerability (SEV) and scaled temporal shape concealment difficulty (\overline{TSCD})

Video object	Error pattern	Video packets per VOP	SEV	\overline{TSCD}	SRN
Tennis Player	Burst 1	2	0.01330	0.68805	0.009151
Tennis Player	Burst 1	4	0.00860	0.68805	0.005919
Tennis Player	Burst 2	2	0.00131	0.68805	0.000904
Tennis Player	Burst 2	4	0.00085	0.68805	0.000585
Background	Burst 1	2	0.04132	0.00657	0.000271
Background	Burst 1	4	0.02142	0.00657	0.000141
Background	Burst 2	2	0.00414	0.00657	0.000027
Background	Burst 2	4	0.00212	0.00657	0.000014

Table 4.4 – Shape refreshment need (SRN) values for the Weather sequence objects, in addition to their shape error vulnerability (SEV) and scaled temporal shape concealment difficulty (\overline{TSCD})

Video object	Error pattern	Video packets per VOP	SEV	\overline{TSCD}	SRN
Weather Girl	Burst 1	2	0.02248	0.00084	0.000019
Weather Girl	Burst 1	4	0.01341	0.00084	0.000011
Weather Girl	Burst 2	2	0.00223	0.00084	0.000002
Weather Girl	Burst 2	4	0.00132	0.00084	0.000001
Weather Map	Burst 1	2	0.03323	0.00014	0.000005
Weather Map	Burst 1	4	0.02198	0.00014	0.000003
Weather Map	Burst 2	2	0.00331	0.00014	0.000001
Weather Map	Burst 2	4	0.00218	0.00014	0.000000

The results in Table 4.3 and Table 4.4 show that the proposed SRN metric behaves rather well in terms of expressing the shape refreshment need for the conditions tested. In fact, for the Stefan sequence, the Background object has a much lower SRN than Tennis Player, meaning that the encoder should refresh the latter much more often. For the Weather sequence, however, the two objects have much closer SRN values, meaning that the encoder should refresh the Weather Girl object only slightly more often than the Weather Map object. Additionally, when the amount of errors increases, the SEV , as well as the SRN , increases, implying that the objects need to be refreshed more. When less VPs are used per VOP, the objects also need to be refreshed more (SRN increases), which is understandable because VPs become larger and, thus, more vulnerable to errors (e.g., less resynchronization available).

4.4.2 Texture Refreshment Need

As already stated, the texture refreshment need depends on several factors described earlier in Section 4.2. The parameter TRN_i^k proposed here measures, for macroblock k in VOP i , the *texture refreshment need*, which corresponds to the contribution of this macroblock to the *accumulated texture refreshment need* parameter ($ATRN_i^k$). In addition to TRN_i^k , the $ATRN_i^k$ parameter also depends on the texture refreshment need of past texture macroblocks, co-located or not with macroblock k . Since in hybrid coding schemes with motion compensation, texture macroblocks in VOP i are predicted from texture macroblocks in the previous VOP by using motion compensation, the $ATRN_i^k$ parameter depends on the accumulated refreshment need of the macroblock used as a prediction instead of the co-located macroblock in the previous VOP. Therefore, by taking motion-compensation into account, $ATRN_i^k$ is simply defined as:

$$ATRN_i^k = MCATRN_i^k + TRN_i^k \quad (4.45)$$

where i is the index corresponding to the VOP number, k is the index corresponding to the macroblock number within VOP i , where macroblocks are counted in raster scan starting at the top-left corner of the bounding box and ending at the bottom-right corner, and $MCATRN_i^k$ is the motion-compensated accumulated texture refreshment need value for VOP i and macroblock k .

In order to determine $MCATRN_i^k$, the $ATRN_{i-1}$ matrix must first be considered. This matrix has the same size (in terms of macroblocks) as the bounding box of VOP $i-1$ and is used to store the accumulated texture refreshment need values of all the macroblocks ($ATRN_{i-1}^k$) in this VOP. As for $MCATRN_i$, which is a matrix with the same size (in terms of macroblocks) as the bounding box of VOP i , it is obtained by motion-compensating $ATRN_{i-1}$, with the texture motion vectors determined for VOP i , as illustrated in Figure 4.12. It is important to notice that, since the shape of video objects changes in time, the bounding boxes of two consecutive VOPs do not necessarily have the same number of macroblocks and of course the same number of opaque, transparent and boundary macroblocks.

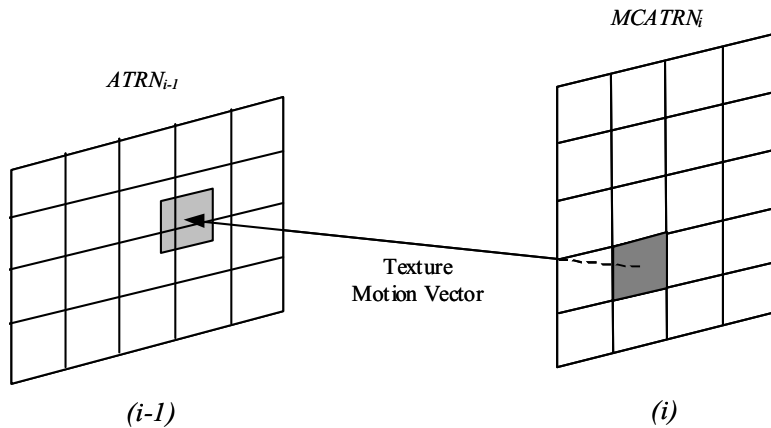


Figure 4.12 – Determination of $MCATRN_i$ by motion-compensating $ATRN_{i-1}$ with the texture motion vectors determined for VOP i

4. Object-based Refreshment Need Metrics

Motion-compensation is used to compute the accumulated texture refreshment need because it is the mechanism that propagates errors in the texture information from past time instants. Therefore, when computing the accumulated texture refreshment need for a given macroblock, the accumulated texture refreshment need of the macroblock(s) used as prediction has to be used. Since, as is shown in Figure 4.12, each texture motion vector can point to a square area of $ATRN_{i-1}$ that includes up to four different $ATRN$ values (corresponding to four texture macroblocks), the maximum of these values is copied into the appropriate position in $MCATRN_i$. Finally, $MCATRN_i^k$ is the value in $MCATRN_i$ corresponding to macroblock k . If macroblock k is either transparent or opaque with intra-coded texture, which means that it does not depend on the past for any texture prediction and no errors can be propagated from the past, $MCATRN_i^k$ is set to zero.

As for the TRN_i^k parameter, it must consider the most important factors impacting the *texture refreshment need* and thus the following definition is proposed:

$$TRN_i^k = TEV_i^k \times TCD_i^k = TEV_i^k \times f_{TCD}(\overline{STCD_i^k}, \overline{TTCD_i^k}) \quad (4.46)$$

where TEV_i^k is the Texture Error Vulnerability (TEV), TCD_i^k is the Texture Concealment Difficulty (TCD), $\overline{STCD_i^k}$ is the scaled Spatial Texture Concealment Difficulty ($STCD$), $\overline{TTCD_i^k}$ is the scaled Temporal Texture Concealment Difficulty ($TTCD$), all for macroblock k in VOP i . Additionally, f_{TCD} is a function that combines the values of $\overline{STCD_i^k}$ and $\overline{TTCD_i^k}$ (both with values between 0 and 1 for the reasons explained in Section 4.4.1) into a single TCD value.

TRN_i^k is defined as a product where the first factor is the texture vulnerability measuring the statistical exposure of the texture data to errors already considering the used bitstream structure, and the second factor corresponds to the texture concealment difficulty measuring how hard the texture is to recover using concealment techniques. A product is used because the dependency of the texture refreshment need on the texture error vulnerability and the texture concealment difficulty is similar to the dependency of the shape refreshment need on the shape error vulnerability and the shape concealment difficulty, which was explained in Section 4.4.1.

4.4.2.1 Defining Texture Error Vulnerability

It is proposed here that the texture error vulnerability, the first factor in Equation (4.46), be measured by the probability of the texture bits of macroblock k having to be discarded due to errors in VOP i . Therefore, TEV_i^k can be defined as:

$$TEV_i^k \equiv P\{\text{the texture data of macroblock } k \text{ of VOP } i \text{ is to be discarded}\} \quad (4.47)$$

where i is the index corresponding to the VOP number, and k is the index corresponding to the macroblock number within VOP i .

This means that TEV_i^k , such as SEV_i , depends on the type and amount of errors, the number of bits used and how they are arranged in the Video Packet (VP), and the error detection capacity of the used decoder. Therefore, it is necessary to consider the structure of one VP in MPEG-4, which was already detailed in Section 4.4.1; at that time, it was explained that the data partitioning mode is the appropriate mode to be used.

Here, similarly to what was done for shape, a conservative recovery strategy is considered, assuming that all decoders can perform it. This recovery strategy implies that if a given VP has at least one error in the first partition (including the shape and motion data), the whole data in the VP will be discarded; moreover, if a given VP has at least one error in the second partition (conveying the texture data), the whole texture data in the VP will be discarded. Since both cases correspond to the discarding of the complete texture data in the VP, one can say that if a given VP has at least one error, the whole texture data in the VP will be discarded; otherwise, no texture data bits will be discarded.

In order to proceed and determine the probability of the texture data in one VP being discarded, it is necessary to consider the possible error patterns, which are the same that were considered in Section 4.4.1.

4.4.2.1.1 Uniform errors

In this case, the texture data in the video packet is considered lost if at least one bit error exists in the VP. Let P_e be the probability of a given bit being in error, then $1-P_e$ is the probability of this bit not being in error. Therefore $(1-P_e)^{bvp}$ is the probability of all the bits being correct in the VP, which means that $1-(1-P_e)^{bvp}$ is the probability of at least one bit being in error. Therefore:

$$P\{\text{the texture data of one VP is to be discarded}\} = 1 - (1 - P_e)^{bvp} \quad (4.48)$$

where P_e is the bit error rate, bvp is the VP size, which includes the resync marker indicating the beginning of the VP, the VP header, the shape data, the motion data, the marker which indicates the end of the first partition and the texture data.

Therefore, Equation (4.47) becomes the following for uniform errors:

$$TEV_i^k = 1 - (1 - P_e)^{bvp}. \quad (4.49)$$

4.4.2.1.2 Burst errors

In this case, the texture data of the video packet is considered lost if at least one error burst starts within the VP. Let P_t be the probability of a burst starting at a given bit, then $1-P_t$ is the probability of no burst starting at that bit. Therefore $(1-P_t)^{bvp}$ is the probability of no bursts starting within the VP, which means that $1-(1-P_t)^{bvp}$ is the probability of at least one burst starting within the VP. Since the probability P_t of a burst starting at a given bit is equal to the transition probability π_{12} , which is the probability of the state changing from 1 to 2 as defined in Section 4.4.1, the probability of at least one error burst starting within the first partition of the VP is:

$$P\{\text{the texture data of the VP is to be discarded}\} = 1 - (1 - P_t)^{bvp}$$

where P_t is the transition probability from state a_1 (outside the burst) to state a_2 (inside the burst) and bvp is the VP size, which includes the resync marker, the VP header, the shape data, the motion data, the marker which indicates the end of the first partition and the texture data.

Therefore, Equation (4.47) becomes the following for burst errors:

$$TEV_i^k = 1 - (1 - P_t)^{bvp}. \quad (4.50)$$

4. Object-based Refreshment Need Metrics

4.4.2.1.3 Packet losses

Assuming that the packets that can be lost have a fixed payload length S (similarly to what was done in Section 4.4.1, this restriction will be relaxed later) and that the start of the first multiplex packet (starting inside the VP) coincides with the first bit in the VP, then the probability of at least one lost multiplex packet starting inside the first partition of the VP is:

$$P\{\text{the texture data of one VP is to be discarded}\} = 1 - (1 - PLR)^{N_{vp}(0)} \quad (4.51)$$

where PLR is the packet loss rate and $N_{vp}(0)$ is the number of multiplex packets starting within the VP assuming that the start of the first packet coincides with the first bit (bit 0) in the VP.

If, on the other hand, the start of the first packet (starting inside the VP) coincides with the second bit (bit 1) in the VP, then the same expression as before can be used by replacing $N_{vp}(0)$ with $N_{vp}(1)$. This reasoning can be repeated for the first S bits in the VP. Assuming that the probability of the first multiplex packet (starting inside the VP) coinciding with any one of the first S bits is equal, then the probability of at least one lost multiplex packet starting inside the VP is:

$$P\{\text{the texture data of the VP is to be discarded}\} = \frac{1}{S} \sum_{n=0}^{S-1} P_B(n) \quad (4.52)$$

with

$$P_B(n) = 1 - (1 - PLR)^{N_{vp}(n)}$$

where PLR is the packet loss rate, S is the payload size of the multiplex packets, $P_B(n)$ is the probability of at least one lost multiplex packet starting in the VP assuming that the start of the first packet coincides with bit n (with n varying between 0 and S), and $N_{vp}(n)$ is the number of multiplex packets starting within the VP, assuming that the start of the first multiplex packet starting inside the VP coincides with bit n in the VP.

If the payload size of the multiplex packets is not fixed, then instead of S , the average packet payload size should be used.

Therefore, Equation (4.47) becomes the following for packet losses:

$$TEV_i^k = \frac{1}{S} \sum_{n=0}^{S-1} \left[1 - (1 - PLR)^{N_{vp}(n)} \right]. \quad (4.53)$$

4.4.2.2 Defining Texture Concealment Difficulty

As for the texture concealment difficulty, the second factor in Equation (4.46), it corresponds to the texture concealment difficulty, i.e. the difficulty to recover the texture if errors occur, which depends on the concealment techniques used at the receiver. Since these techniques can be spatial and/or temporal, the texture concealment difficulty in Equation (4.46) is expressed as a function f_{TCD} of \overline{STCD}_i^k and \overline{TTCD}_i^k that has to be defined according to the used error concealment solutions. Similarly to what was said for the shape concealment difficulty in Section 4.4.1.2, the possible concealment solutions are either static or dynamic.

Since the $STCD$ and $TTCD$ metrics are computed at the encoder to have an idea of how difficult the concealment will be at the decoder, it is important that the metric be applied to the locally reconstructed texture (encoded and locally decoded) instead of the original texture

since this is the information that will be available at the decoder. For instance, if the metrics were applied to the original texture and a large quantization step were used for the texture coding, the $STCD$ and $TTCD$ values at the encoder would not precisely reflect the concealment difficulty at the decoder because the reconstructed texture at the decoder would be different and, very likely, less complex than the original texture.

4.4.2.2.1 Spatial Texture Concealment Difficulty

The \overline{STCD}_i^k term in Equation (4.46) expresses how difficult it is to recover the texture of the macroblock in question by only using spatial information. It is proposed here that the definition of the $STCD$ measure be based on the Spatial perceptual Information (SI) metric defined in ITU-T Recommendation P.910 [P.910], which is computed as:

$$SI = \max_{time} \{std_{space}[Sobel(Y_i)]\} \quad (4.54)$$

where Y_i is the luminance component of frame i , $Sobel()$ is the Sobel filter operator, std_{space} is the standard deviation over the pixels of one frame and max_{time} is the maximum value over time.

The SI metric was developed by ITU-T to evaluate the amount of spatial information on test scenes and it corresponds to an edginess measure. This measure is computed by first determining an edginess value for each frame in the sequence (the standard deviation of the Sobel-filtered image) and then choosing its maximum value to represent the whole sequence.

The major difference between $STCD$ and SI is that $STCD$ has to be an instantaneous value for a given macroblock and not the maximum over time as SI . Therefore, the definition of $STCD_i^k$ becomes:

$$STCD_i^k = std_{space}[Sobel(Y_i^k)] \quad (4.55)$$

where Y_i^k is the luminance component of macroblock k in VOP i . In this case, the luminance space is the set of pixels for each macroblock for which there is a shape support (opaque shapels). Only the luminance component is used here due to its primordial relevance in terms of the Human Visual System (HVS).

In order to illustrate the performance of the proposed $STCD$ measure, results are included below for the texture macroblocks with different complexities shown in Figure 4.13. They correspond to reconstructed macroblocks extracted from the objects Weather Girl, Tennis Player and Background (from the Stefan sequence). These macroblocks were encoded with a quantization step of 2 (the lowest possible in MPEG-4) in order to guarantee that the reconstructed texture is the closest possible to the original one, preserving most of the original spatial detail (i.e. a worst-case situation in terms of spatial detail is created).

4. Object-based Refreshment Need Metrics

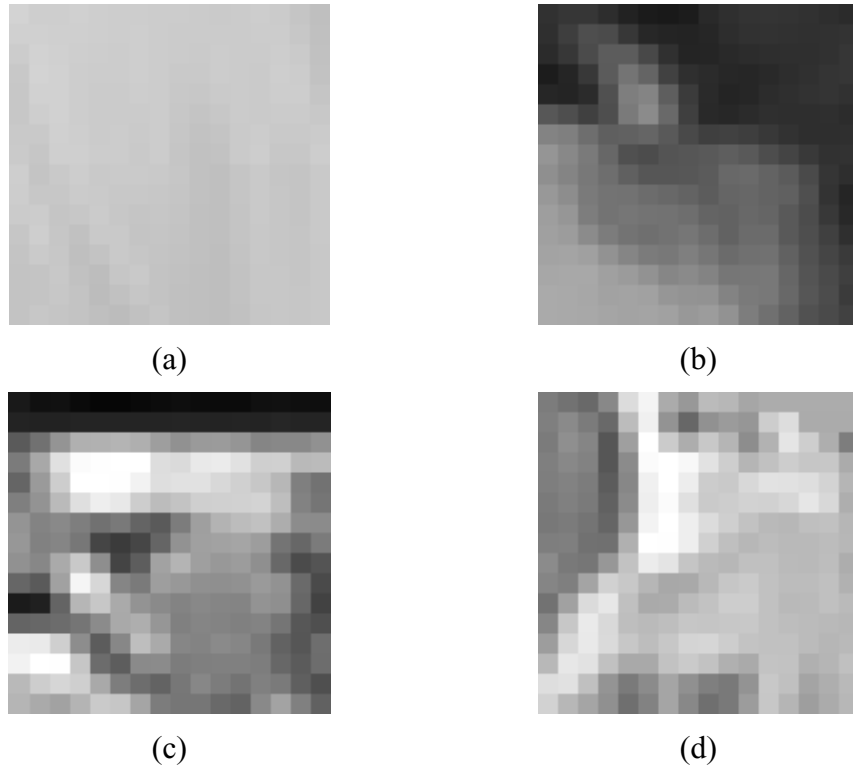






Figure 4.13 – Macroblocks with different complexities: (a) and (b) macroblocks 71 and 15 of *Weather Girl* (frame 0, CIF); (c) macroblock 14 of *Background* (frame 0, CIF); (d) macroblock 14 of *Tennis Player* (frame 0, CIF)

Just by looking at the texture macroblocks depicted in Figure 4.13, any error concealment expert can rank them according to their spatial concealment difficulty, starting with (c) as the most difficult followed by (d), (b) and finally (a). This order should be reflected by the *STCD* values, shown in Table 4.5, indicating that the proposed objective metric reaches the target of automatically expressing how difficult it is to conceal a given texture macroblock with spatial concealment techniques.

Table 4.5 - *STCD* measures for texture macroblocks with different complexities

Video object	Resolution	Macroblock	STCD
Weather Girl	CIF		12.3804
Weather Girl	CIF		65.4812
Background	CIF		78.4404
Tennis Player	CIF		72.0860

The analysis of the results in Table 4.5 leads to the conclusion that the proposed *STCD* measure behaves rather well in terms of expressing the spatial texture concealment difficulty.

Theoretically, the *STCD* measure ranges from 0 (corresponding to the uniform macroblock where all pixels have the same value) to 127.5 (when half the pixels in the outcome of the Sobel filter have 0 value and the other half has 255, independently of how they are distributed in the macroblock). But, although the theoretical maximum of this measure is 127.5, this value can only be reached for *academic* situations. By using two very textured sequences in the MPEG-4 test set, *Stefan* and *Children* (especially the shirt of the child on the right), and

considering that in terms of spatial concealment difficulty the worst-case corresponds to using a quantization step of 2 for the texture encoding (i.e. almost no detail is removed by the quantization process), it can be shown that the typical maximum *STCD* values are much lower than the theoretical *STCD* maximum. In Figure 4.14 and Figure 4.15, the maximum *STCD* values for each VOP (taken at macroblock level) are shown for the Background objects of the Stefan sequence and the Kids object of the Children sequence, respectively, for different resolutions.

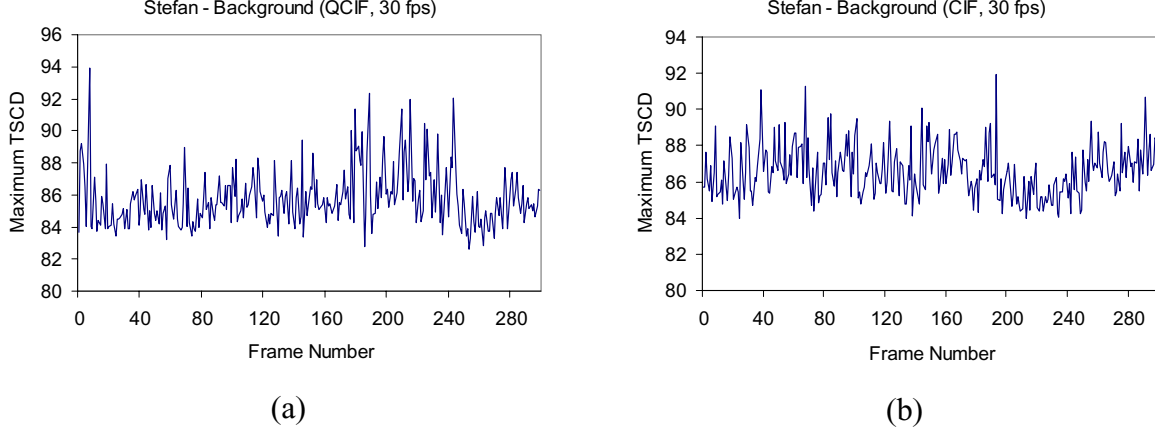


Figure 4.14 – *STCD* for the Background object of the Stefan sequence: (a) QCIF at 30 fps; (b) CIF at 30 fps

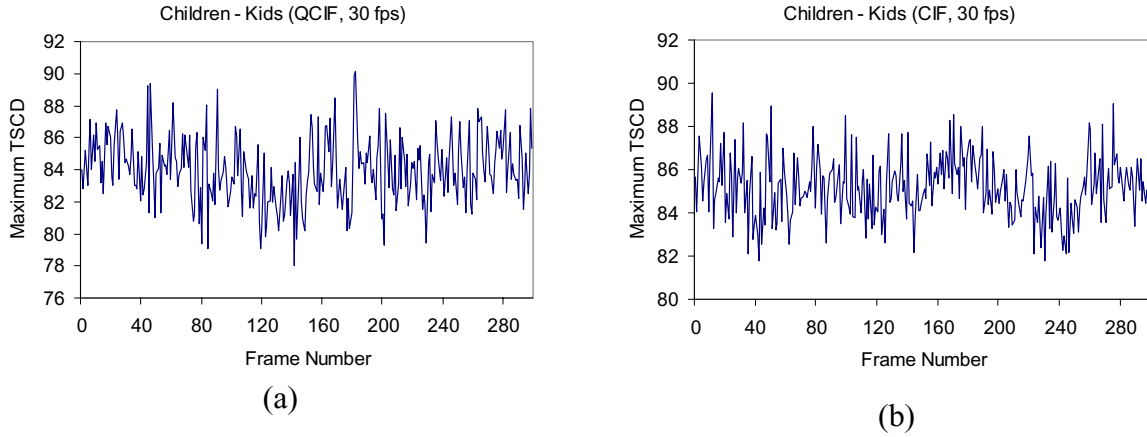


Figure 4.15 – *STCD* for the Kids object: (a) QCIF at 30 fps; (b) CIF at 30 fps

From Figure 4.14 and Figure 4.15, it can be seen that the most textured macroblock (in VOP 7 of the QCIF version of Stefan's Background object) reaches a *STCD* value of 93.9206. This value will be taken as reference for the truncation and scaling of the *STCD* measure. Therefore, the scaled *STCD* (\overline{STCD}) metric shall be defined as follows:

$$\overline{STCD} = \begin{cases} \frac{STCD}{94}, & 0 \leq STCD < 94 \\ 1, & STCD \geq 94 \end{cases} \quad (4.56)$$

4. Object-based Refreshment Need Metrics

4.4.2.2.2

Temporal Texture Concealment Difficulty

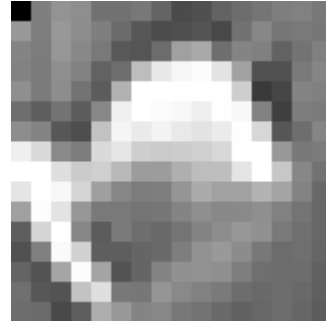
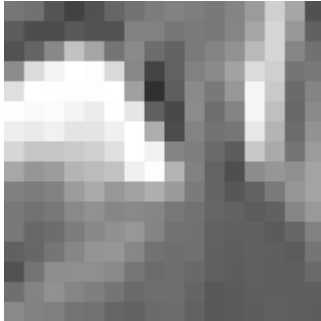
The \overline{TTCD}_i^k term in Equation (4.46) expresses how difficult it is to conceal the texture macroblock in question by using only temporal information. This difficulty can be simply measured by the Sum of the Absolute Differences (*SAD*) between the current and the previous time instants defined as:

$$TTCD_i^k = SAD = \sum_{\substack{\text{opaque} \\ \text{shapels}}} |Y_i^k(x, y) - Y_{i-1}^k(x, y)| \quad (4.57)$$

where i is the index corresponding to the VOP number, k is the index corresponding to the macroblock number within the VOP, $Y_i^k(x, y)$ is the luminance value of the pixel with (x, y) coordinates in the k -th macroblock of the i -th VOP, and $Y_{i-1}^k(x, y)$ is the luminance value of the pixel co-located with $Y_i^k(x, y)$ in the $(i-1)$ -th VOP.

The texture concealment difficulty when temporal concealment techniques are used is basically dictated by the amount of changes in the texture from one time instant to the next. Therefore, a good way to measure this is by means of the differences in luminance values corresponding to opaque shapels in the current and previous alpha planes, which can be easily done with the *SAD* metric. This, of course, does not mean that other metrics are not possible, such as the Mean Square Error (*MSE*). The *SAD* could also be computed after motion compensating the prediction macroblocks from the previous time instant. However, this would mean assuming that the decoder can use motion compensation for temporal concealment, which is not always true. The decoder may not be able to do so because it either lacks the computational power (i.e., it is a very simple decoder) or the motion vectors might not be available (e.g., because they were corrupted by errors).

In order to illustrate the performance of the proposed *TTCD* measure, results are shown below for the texture macroblock pairs in Figure 4.16 with different amounts of temporal change. They correspond to reconstructed macroblocks extracted from the Cyclamen, Weather Girl and Tennis Player objects with a frame rate of 10 fps (the original material is at 30 fps). These macroblocks were encoded with a quantization step of 2 (the lowest possible in MPEG-4) in order to guarantee that the reconstructed texture is the closest possible to the original one, preserving most of the original spatial detail (i.e. a worst-case situation in terms of spatial detail).



(a)

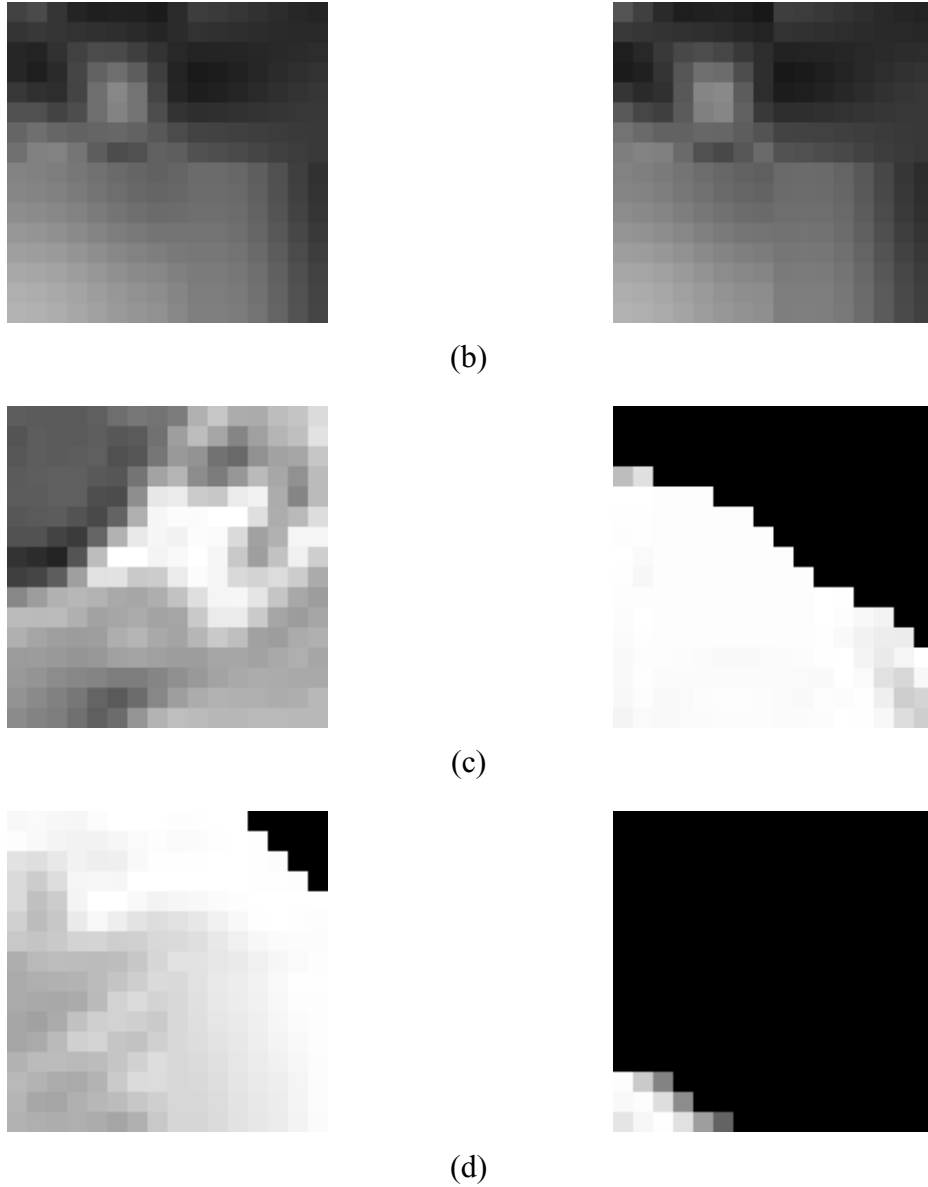










Figure 4.16 – Samples of texture macroblock pairs from various object sequences: (a) *Cyclamen* – macroblock 104 of VOP 3 (left) and co-located one in VOP 0 (right); (b) *Weather Girl* – macroblock 15 of VOP 3 (left) and co-located one in VOP 0 (right); (c) *Tennis Player* – macroblock 19 of VOP 99 (left) and co-located one in VOP 96 (right); (d) *Tennis Player* – macroblock 20 of VOP 99 (left) and co-located one in VOP 96 (right)

By looking at the texture macroblock pairs displayed in Figure 4.16, any error concealment expert can rank them according to their temporal concealment difficulty, starting with (d) as the hardest to conceal with temporal techniques, followed by (c), (a) and, finally, (b). This order should be clearly reflected by the *TTCD* values, shown in Table 4.6, indicating that the proposed objective metric reaches the target to automatically express how difficult it is to conceal a given texture macroblock with temporal concealment techniques.

4. Object-based Refreshment Need Metrics

Table 4.6 – *TTCD measures for different texture macroblocks*

Video object	Resolution	Macroblock	Co-located macroblock in previous time instant	TTCD
Cyclamen	CIF			14096
Weather Girl	CIF			892
Tennis Player	CIF			31841
Tennis Player	CIF			53409

The analysis of the results in Table 4.6 leads to the conclusion that the proposed *TTCD* measure behaves rather well in terms of expressing the temporal texture concealment difficulty.

Theoretically, the *TTCD* measure ranges from 0 (when the absolute difference is 0 for all the 256 pixels of a macroblock) to 65280 (when the absolute difference is 255 for all the 256 pixels in a macroblock). However, this theoretical maximum can only be reached for *academic* situations. By using the Stefan sequence, which is clearly the most textured and fastest sequence within the MPEG-4 test set and considering that, in terms of temporal concealment difficulty, the worst-case corresponds to using a quantization step of 2 for the texture encoding and a frame rate of 15 fps (which is the lowest frame rate acceptable for this sequence), it can be shown that the typical maximum *TTCD* values are much lower than the theoretical *TTCD* maximum. In Figure 4.17 and Figure 4.18, the maximum *TTCD* values (computed over all macroblocks for each VOP) are shown for the Tennis Player and Background objects, respectively, for different resolutions.

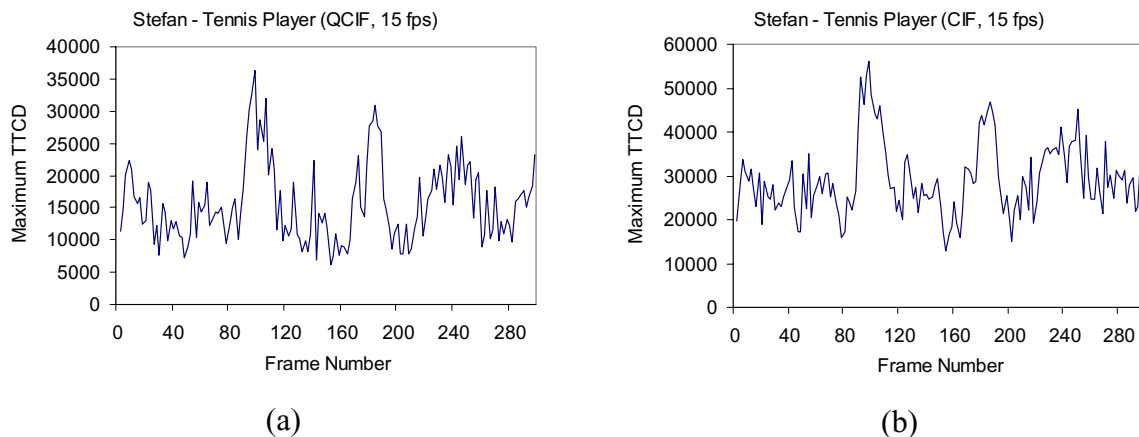


Figure 4.17 – *TTCD for the Tennis Player object: (a) QCIF at 15 fps; (b) CIF at 15 fps*

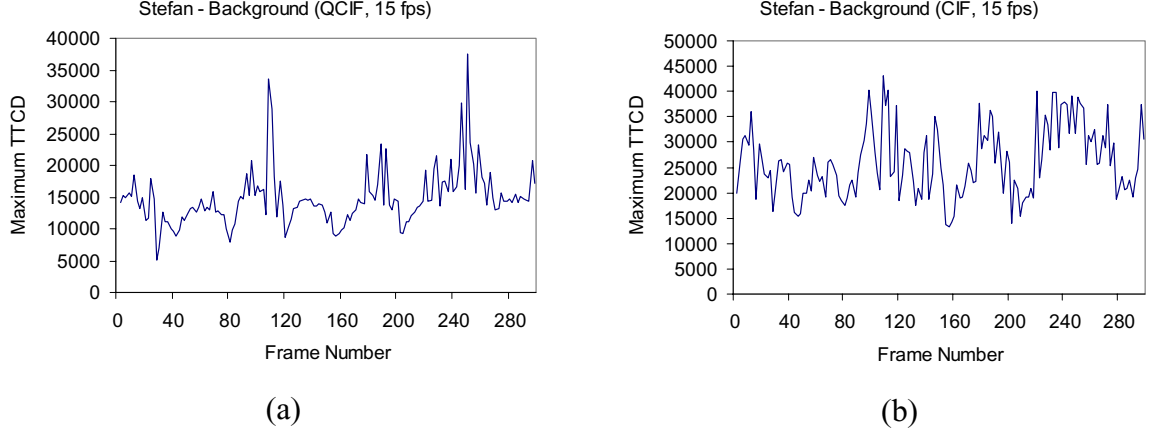


Figure 4.18 – $TTCD$ for the Background object of the Stefan sequence: (a) QCIF at 15 fps; (b) CIF at 15 fps

From Figure 4.17 and Figure 4.18, it can be seen that the most unstable texture macroblock (in frame 98 of the CIF version of Tennis Player) reaches a $TTCD$ value of 56055. This value will be taken as reference for the truncation and scaling of the $TTCD$ measure. Therefore, the scaled $TTCD$ (\overline{TTCD}) measure shall be defined as follows:

$$\overline{TTCD} = \begin{cases} \frac{TTCD}{57000}, & 0 \leq TTCD < 57000 \\ 1, & TTCD \geq 57000 \end{cases} \quad (4.58)$$

4.4.2.3 Evaluating the Texture Refreshment Need Metric Performance

To show the performance of the texture refreshment need metric proposed in this paper, TRN_i^k of Equation (4.46), results are given in Table 4.7 for two different macroblocks from VOP 3 of the Background object from the Stefan sequence (QCIF at 192 kbps), when 2 and 4 VPs are used per VOP and for the same two different burst error patterns used in Section 4.4.1.3. One macroblock corresponds to macroblock 22 in the VOP, which is located in the (very inhomogeneous) audience part (MB1), while the other corresponds to macroblock 71 in the VOP and is located in the (homogeneous) court part (MB2). In Table 4.8, results are given for two different macroblocks from VOP 3 of the Weather Girl object from the Weather sequence (QCIF at 56 kbps), for the same conditions as above. One macroblock corresponds to macroblock 7 (MB3) in the VOP and the other to macroblock 11 (MB4), being both located in the face of the Weather Girl. In both cases, the encoder considers that temporal concealment techniques are used at the decoder, which means that the f_{TCD} function is simply equal to \overline{TSCD} .

4. Object-based Refreshment Need Metrics

Table 4.7 – Texture refreshment need (*TRN*) values for two different macroblocks in the Stefan sequence, in addition to their texture error vulnerability (*TEV*) and scaled temporal texture concealment difficulty (\overline{TTCD})

Macroblock	Error pattern	Video packets per VOP	TEV	\overline{TTCD}	TRN
MB1	Burst 1	2	0.18386	0.28698	0.05276
MB1	Burst 1	4	0.09762	0.28698	0.02802
MB1	Burst 2	2	0.01975	0.28698	0.00567
MB1	Burst 2	4	0.01004	0.28698	0.00288
MB2	Burst 1	2	0.17085	0.00589	0.00101
MB2	Burst 1	4	0.08808	0.00589	0.00052
MB2	Burst 2	2	0.01823	0.00589	0.00011
MB2	Burst 2	4	0.00901	0.00589	0.00005

Table 4.8 – Texture refreshment need (*TRN*) values for two different macroblocks in the Weather sequence, in addition to their texture error vulnerability (*TEV*) and scaled temporal texture concealment difficulty (\overline{TTCD})

Macroblock	Error pattern	Video packets per VOP	TEV	\overline{TTCD}	TRN
MB3	Burst 1	2	0.08121	0.04056	0.003294
MB3	Burst 1	4	0.04610	0.04056	0.001870
MB3	Burst 2	2	0.00828	0.04056	0.000336
MB3	Burst 2	4	0.00462	0.04056	0.000187
MB4	Burst 1	2	0.04567	0.03293	0.001504
MB4	Burst 1	4	0.03895	0.03293	0.001283
MB4	Burst 2	2	0.00458	0.03293	0.000151
MB4	Burst 2	4	0.00389	0.03293	0.000128

The results in Table 4.7 and Table 4.8 show that the proposed *TRN* metric is able to adequately express the texture refreshment need for the conditions here tested. In fact, MB2 has a much lower *TRN* than MB1, meaning that the encoder should refresh the latter much more often, as could be expected. As for MB3 and MB4, their *TRN* values are much closer, meaning that the encoder should refresh only slightly more often the former. Additionally, when the amount of errors increases, the *TEV*, as well as the *TRN*, increases, implying that the macroblocks need to be more refreshed. When less VPs are used per VOP, the macroblocks also need to be more refreshed (*TRN* increases), which is understandable because VPs become larger and, thus, more vulnerable to errors (there is less resynchronization and more bits are lost each time there is an error).

4.5 Final Remarks

In this chapter, both shape and texture refreshment need metrics were proposed; these metrics express the necessity of refreshing the shape and the texture data while coding a given video object. While the VOP-based shape refreshment need metric, *SRN*, is defined as a product of the shape error vulnerability and the shape concealment difficulty, the macroblock-based texture refreshment need metric, *TRN*, is similarly defined as a product of the texture error

vulnerability and the texture concealment difficulty. In both cases, the error vulnerability measures the statistical exposure of the shape/texture data to channel errors and the concealment difficulty expresses how difficult the corrupted shape/texture data is to recover when spatial and/or temporal error concealment techniques are considered.

Since the results have shown that the proposed metrics are able to correctly express the necessity of refreshment, it is anticipated that they can be extremely useful in allowing the encoder to efficiently decide which video objects should have their shapes and texture macroblocks intra coded to improve the subjective impact at the decoder side. For instance, in terms of shape, the encoder can decide to increase or decrease the shape refreshment periods of the various video objects in a scene, depending on their *SRN* values. Similarly, in terms of texture, the encoder can distribute the available texture refreshment resources among the various video objects in a scene, depending on the average *TRN* values of the macroblocks in each video object, and then distribute the assigned texture refreshment resources within each VOP based on the individual *TRN* values of the macroblocks. This objective is extremely important, especially for emerging mobile applications, where error conditions are challenging and bandwidth resources are scarce and thus maximizing the quality is a must.

In the following chapter, an adaptive object-based video coding refreshment scheme, dealing with both shape and texture data, will be proposed; in this scheme, the encoder will rely on the shape and texture refreshment need metrics proposed here to decide which data should be refreshed and when.

Chapter 5

Adaptive Object-based Video Coding

Refreshment Scheme

5.1 Introduction

In Chapter 4, two refreshment need metrics were proposed: the *shape refreshment need* and the *texture refreshment need*. As explained then, these metrics have been defined because, in the context of object-based video coding systems, it would be helpful for the encoder to have a method to determine which components of the video data (shape and texture) of which objects should be refreshed and when, especially if based on an appropriate set of *refreshment need* metrics. This way, based on the refreshment need metrics previously defined, it is now possible to propose an adaptive object-based video coding refreshment scheme that is able to deal with both shape and texture data.

With respect to the type of refreshment to be used (intra versus inter), only intra refreshment solutions will be considered in the following. This choice mainly results from the objective of developing a refreshment scheme to be used in MPEG-4 compliant terminals; since leaky-difference techniques cannot be used in a compatible way with the MPEG-4 standard¹ (or would require changing the standard), intra refreshment is the natural solution since it does not impact standard compatibility.

As said in Chapter 3, the refreshment technique depends on some characteristics of the application, notably if encoding is done in real-time or not and if information about the channel errors is available or not. The solution proposed in this section will take into account

¹ As well as with all the other available standards, such as H.261, H.263, MPEG-1 and MPEG-2.

the information available about the channel errors, if this information is available. If this information is not available, the refreshment scheme will still work although its performance may be less than optimal.

In addition, only applications that do not have random access requirements will be considered because this corresponds to the most interesting situation in terms of refreshment (more flexibility is allowed instead of an imposed periodic full refreshment solution). Examples of applications that fall into this category are mobile video telephony and point-to-point video streaming on the Internet. The former application is a representative of a burst errors scenario while the latter is a representative of a packet-loss scenario; these are the major error/application scenarios considered in this Thesis since they are the most relevant in practice and involve different types of errors and different types of content.

This chapter, dealing with the coding refreshment problem in the context of object-based video coding architectures is organized as follows. In Section 5.2, the object-based coding refreshment scheme targeting the MPEG-4 standard is proposed. Then, in Section 5.3, the performance of the proposed solution is evaluated and, finally, Section 5.4 concludes the chapter.

5.2 Combined Shape-Texture Intra Refreshment Procedure

Having defined the refreshment needs for both the shape and texture data in Chapter 4, it is now possible to describe the combined shape-texture intra refreshment procedure proposed in this chapter. For the sake of clarity, a high level description of the algorithm is given first, which is followed by a more detailed explanation of the various elementary modules. Finally, the algorithm is summarized as a set of successive steps.

5.2.1 High Level Description of the Combined Intra Refreshment Procedure

The proposed intra refreshment algorithm considers a multi-object video scene, where all the video objects are encoded at the same VOP rate², and has the target to efficiently distribute the available video refreshment resources among the various video objects, depending on their refreshment needs. It is considered that some refreshment resources (i.e., some bit rate which is allocated with the specific purpose of increasing error resilience by means of intra refreshment) are initially allocated; the amount of these resources depends on the efficiency-resilience bit rate trade-off adopted. This trade-off can always be moved in one direction or the other, meaning that it is always possible to increase the amount of refreshment resources and thus the resilience at the cost of coding efficiency, and vice-versa (considering that the total available bit rate is fixed). The amount of refreshment resources and thus the chosen efficiency-resilience trade-off are defined by means of two parameters specified by the user, at the beginning of encoding or later on, if an update is necessary and possible:

- **Target Average Shape Refreshment Period ($ASRP_{target}$)** – This parameter is the target average of the individual Shape Refreshment Periods (SRP) of the various video objects in the scene, at a given time instant. The values of the individual $SRPs$ are updated at every time instant and, for each object, they are always relative to the last VOP with intra

² Since all the video objects have the same VOP rate, at a given coding time instant all the VOPs have the same number i . Therefore, in all of the following, when referring to the time instant when all the VOPs in the scene have number i , the phrase “time instant i ” will be used.

5. Adaptive Object-based Video Coding Refreshment Scheme

coded shape. The proposed intra refreshment algorithm has the target, at each time instant, to assign adequate *SRP* values to each video object, taking into account their refreshment needs, while maintaining the target average shape refreshment period, $ASRP_{target}$, specified by the user.

- **Number of Texture Macroblocks to be Refreshed (NTMR) at each time instant** – This parameter indicates the total number of texture macroblocks (considering all the video objects) to be refreshed at a given time instant. This means that, at a given time instant, the proposed algorithm has to decide how many texture refreshment macroblocks shall be assigned to each VOP and which macroblocks should be refreshed within each VOP. These assignments are done while taking into account the texture refreshment needs of the macroblocks in each VOP and guaranteeing that the total number of refreshed macroblocks at that time instant is equal to the value specified by the user, *NTMR*.

These parameters establish the amount of shape and texture intra refreshment resources to be used, implying that a certain amount of the bit rate will have to be used for these refreshments and not to increase the error-free video quality. After these two refreshment parameters have been specified, the encoder has to distribute the corresponding refreshment resources, as efficiently as possible, among the various video objects to be refreshed. By doing this, it is expected that, for a given bit rate, the overall video quality will be improved when compared to the cases where less sophisticated refreshment schemes are used (e.g. a fixed refreshment period is used for all the objects). This expectation is based on the principle that entities with low refreshment need can be refreshed less often without a significant reduction in their quality, thus saving refreshment resources, and that the quality of entities with high refreshment need will greatly improve if they are refreshed more often, using the refreshment resources saved from the objects with low refreshment need.

The proposed algorithm performs an efficient allocation of the available refreshment resources as follows:

- **Shape** – At each time instant, the encoder has to rank the existing video objects according to their average shape refreshment need (following the metric proposed in Section 4.4.1). Those with higher refreshment needs will have their shape refreshment periods decreased, and vice-versa. The average of the shape refreshment periods is maintained constant, for all time instants, and equal to the $ASRP_{target}$. Otherwise, the amount of shape refreshment resources used would be higher than established, decreasing the error-free quality (in this case, the error-free quality of the texture because lossless shape coding is used).
- **Texture** – At each time instant, the encoder has to rank the existing video objects according to (the average of) the accumulated texture refreshment need of their macroblocks (following the metric proposed in Section 4.4.2). Those with higher refreshment needs will have higher intra refreshment rates assigned, and vice-versa. The number of refreshed texture macroblocks per time instant is maintained constant, for all time instants, and equal to *NTMR*. Otherwise, the amount of texture refreshment resources used would be higher than established, decreasing the error-free texture quality.

The encoder architecture adopted to implement the proposed intra refreshment scheme targeting the best allocation of the initially defined refreshment resources is illustrated in Figure 5.1.

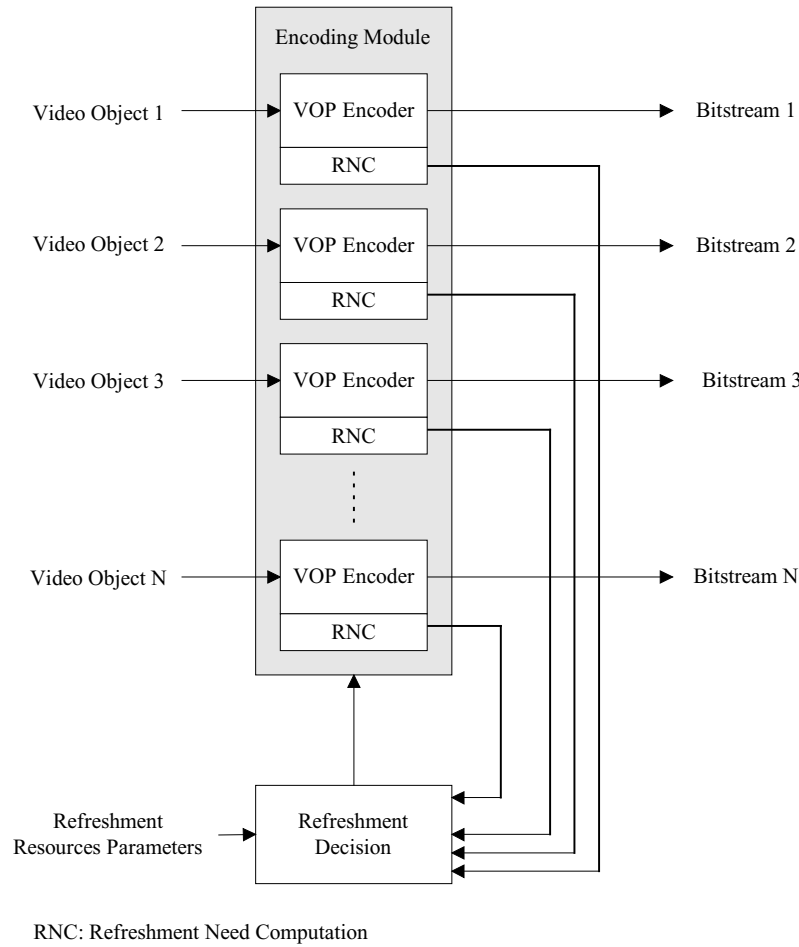


Figure 5.1 – Multi-object intra refreshment encoder architecture

The encoder architecture includes two major modules:

- **Encoding module** – This module is responsible for the encoding of the VOPs making up the scene, with or without intra refreshment, as well as for computing all the necessary refreshment need parameters.
- **Refreshment decision module** – This module is responsible for deciding which VOPs will be refreshed in terms of shape and which macroblocks, within each VOP, will be refreshed in terms of texture. These decisions are taken based on the refreshment need parameters provided by the VOP encoder module and the refreshment resources parameters initially determined by the user.

5.2.2 Encoding Module

The encoding module consists of as many VOP encoders as video objects in the scene being encoded. Each one of these VOP encoders is responsible for encoding the VOPs corresponding to a single video object and computing its refreshment need values.

As for the encoding itself, no further explanations will be given since this is not really the scope of this chapter and it is not necessary for the understanding of the technique, which can be applied to any off-the-shelf MPEG-4 object-based video encoder. Anyway, the basics on

5. Adaptive Object-based Video Coding Refreshment Scheme

MPEG-4 video encoding can be found in [Pereira02]. However, some explanations have to be given in terms of the computation of the refreshment need parameters.

- **Shape** – The shape refreshment need is computed for each object as the average shape refreshment need since the last (and including it) shape intra refreshment ($ASRN_i$ from Section 4.4.1). At each time instant i , $ASRN_i$ is computed for all the N video objects making up the scene by using Equation (4.2). Therefore, at each time instant i , the encoding module, after the encoding itself has taken place, outputs the following vector with the $ASRN_i$ values for all the N video objects:

$$\mathbf{ASRN}_i^T = [ASRN_i^1 \quad ASRN_i^2 \quad \dots \quad ASRN_i^N]. \quad (5.1)$$

This vector is given to the refreshment decision module, which will use it to take the necessary shape intra refreshment decisions regarding the next time instant.

- **Texture** – The texture refreshment need is computed for each macroblock within each VOP to be coded as the accumulated texture refreshment need since the last texture intra refreshment for that macroblock. At each time instant i , $ATRN_i^k$ is computed for all the macroblocks of all N video objects making up the scene by using Equation (4.45). Therefore, at each time instant i , the encoding module, after the encoding itself has taken place, outputs one vector for each video object in the scene (adding to a total of N) with the $ATRN_i^k$ values for all the macroblocks in the corresponding VOP:

$$\mathbf{ATRN}_i^T = [ATRN_i^1 \quad ATRN_i^2 \quad \dots \quad ATRN_i^M] \quad (5.2)$$

where M is the number of macroblocks in the video object at time instant i .

These vectors are given to the refreshment decision module, which will use them to make the necessary texture intra refreshment decisions regarding the next time instant.

5.2.3 Refreshment Decision Module

The refreshment decision module corresponds to the core of the proposed intra refreshment scheme and decides which VOPs have their shape refreshed and which texture macroblocks are refreshed for each VOP, at each time instant.

5.2.3.1 Shape Intra Refreshment Decision

In terms of shape, the refreshment decision module receives as input the vector \mathbf{ASRN}_{i-1} from the encoding module and the target average shape refreshment period ($ASRP_{target}$) parameter set by the user. Based on these values, the refreshment decision module has to decide which VOPs will have their shape refreshed at the next time instant i . The decision is made based on the principle that not all the video objects in the scene have equal shape refreshment needs and, therefore, should not have the same shape refreshment period. Since the selected $ASRP_{target}$ value sets the overall efficiency-resilience bit rate trade-off, video objects with higher refreshment needs should have a lower refreshment period than $ASRP_{target}$, while video objects with lower refreshment needs should have a higher refreshment period, in order to maximize the final subjective impact in the presence of errors.

In order to make these decisions, the refreshment decision module goes through the following two steps:

1. **Shape refreshment period computation** - The shape refreshment period for each video object is computed, while guaranteeing that their average over all the objects is kept equal to the $ASRP_{target}$ value specified by the user. At time instant i , new shape refreshment periods, which are defined in terms of number of VOPs instead of a time interval, are computed for all the N video objects in the scene based on the $ASRN_{i-1}$ values. This does not guarantee that the new SRP for an object will be greater than or equal to the number of elapsed VOPs since the last intra coding VOP, but this is not a problem as will be seen in the following.
2. **Individual shape refreshment decision** - The shape refreshment decision module decides which shapes will be refreshed, at time instant i . This is done by considering the new shape refreshment periods computed in the previous step and the last VOPs that had their shapes refreshed. This way, if the number of elapsed VOPs since the last intra refreshment (including it) is greater than or equal to the shape refreshment period for a given video object, then it should have its shape refreshed at time instant i .

Each of these two steps is explained, in detail, below.

5.2.3.1.1 Shape Refreshment Period Computation

This step is responsible for updating the shape refreshment period parameter for each video object by taking also into account the refreshment needs of the VOPs coded after the last intra coded VOP (including this one); thus, an \mathbf{SRP}_i vector defined as follows is computed:

$$\mathbf{SRP}_i^T = [SRP_i^1 \quad SRP_i^2 \quad \dots \quad SRP_i^N], \quad (5.3)$$

where N is the number of objects in the scene.

Since the updated shape refreshment period parameters depend on the average shape refreshment needs of the existing video objects, computed for the previous time instant, and on the target average shape refreshment period introduced by the user, the \mathbf{SRP}_i vector can be described as a function g of the input vector \mathbf{ASRN}_{i-1} and $ASRP_{target}$:

$$\mathbf{SRP}_i = g(\mathbf{ASRN}_{i-1}, ASRP_{target}). \quad (5.4)$$

At time instant i , the function g assigns shorter or longer $SRPs$ to the individual video objects according to their $ASRN_{i-1}$ values, while guaranteeing that the average shape refreshment period ($ASRP_{avg}$) is maintained equal to $ASRP_{target}$. However, since several sets of SRP values, with different variation ranges, guarantee the same $ASRP_{avg}$, equal to the $ASRP_{target}$ value specified by the user, it is necessary to choose the most adequate SRP variation range in terms of the achieved decoded video quality. Therefore, in order to determine the relationship between $ASRN$ and SRP , it is necessary to know the maximum admissible range for both these parameters.

A) SRP Range

For the individual SRP values, the determination of the lower bound is straightforward since the minimum shape refreshment period possible is 1 (only intra coding is used). As for the upper bound, the maximum shape refreshment period (SRP_{max}) is automatically limited since the number of video objects and $ASRP_{target}$ are known. Therefore, the maximum shape refreshment period happens when all but one video object have the minimum shape refreshment period, i.e.:

5. Adaptive Object-based Video Coding Refreshment Scheme

$$SRP_{\max} = N \cdot (ASRP_{\text{target}} - 1) + 1, \quad (5.5)$$

which defines the range that can be used for the individual shape refreshment period values as $[1, SRP_{\max}]$.

B) ASRN Range

As for the individual *ASRN* values, their variation range is $[0,1]$ as designed in Section 4.4.1. However, for a given type of channel errors and a given type of error concealment, the actual *ASRN* values fall within a smaller sub-interval of the full range. This happens because the *SRN* is defined as a product of two terms, each one varying between 0 and 1; this means that when a specific set of conditions is considered (channel errors and error concealment) the values of the first and the second terms also fall within a sub-interval of the range $[0,1]$, implying that the full *SRN* range will also not be covered. Therefore, if the full *ASRN* range were considered, for a given type of channel and error concealment technique, the differences between the individual *ASRN* values for all the video objects in the scene would be very small when compared to the full range. This would probably mean that no discrimination capability would exist for the various video objects and thus all of them would be assigned the same individual intra refreshment period (and equal to $ASRP_{\text{target}}$), rendering the proposed dynamic refreshment technique useless. Therefore, only a smaller sub-interval of the range $[0,1]$, more adequate to the specific *ASRN* variation characteristics of the specific channel and error concealment conditions, has to be used. This effective sub-interval of the range has to be determined, for each combination of channel error and error concealment conditions, by running tests with a representative set of video object sequences. This sub-interval will be taken as the new *ASRN* range for that specific set of conditions. If the operating conditions change over time, the encoder can easily adjust by updating the *ASRN* range that is used. In fact, the more information is available about the operating conditions – channel errors and decoder concealment solution – the more adequate will be the *ASRN* range used at coding time. If, on the other hand, no information is known about the operating conditions, the encoder can only guess what *ASRN* range to use. For instance, the encoder can choose a conservative approach and consider a worst-case scenario where the channel errors are very high and the decoder concealment solution is the simplest. Alternatively, it can consider a moderate-case scenario and adjust the used *ASRN* range accordingly.

To do this, for each type of relevant channel error characteristics, sequences whose video objects have different shape refreshment needs are chosen and encoded in conditions that correspond to the highest refreshment needs possible. This corresponds to encoding the sequences with the lowest acceptable frame rate (e.g. 15 fps for Stefan and 10 fps for Weather), in QCIF format and with lossless shape coding. In addition, the number of video packets per VOP is varied from 1 to 5, which are the most common values (e.g. 4 video packets per VOP were used in the experiments carried out by the MPEG Error Resilience Ad Hoc Group [N1586] as well as during the MPEG-4 video error resilience tests performed in June 1998 [N2165]).

As for the bit rates used to encode each video object, they have to be carefully chosen because in some cases they can influence the *ASRN* values. This happens when burst errors are considered, but not when uniform errors and packet losses are considered. In all cases, the shape is encoded in lossless mode, which means that by varying the used bit rate the number of bits spent on shape data remains unchanged. Therefore, as long as the errors are specified in terms of a bit error rate there is no problem since the same number of errors will affect the shape data. However, in the case of burst errors the burst length is specified in seconds,

which means that the length of a burst in bits will depend on the used bit rate. As an example, consider that to encode the shape of a given VOP, 1000 bits are needed. If uniform errors with a BER of 10^{-3} are introduced, the amount of errors in the considered shape data is independent of the used bit rate (an average of 1 error). However, if burst errors with an average BER of 10^{-3} and a burst length of 1 ms are used, this is no longer the case because the burst length in bits is proportional to the used bit rate, which means that the amount of errors will change. Therefore, the bit rates chosen for the experiments to determine the adequate *ASRN* range will be the same that will be used later in Section 5.3.3, which are considered reasonable values for the considered sequences.

By running these experiments, for each video object in a sequence, one shape refreshment need value is generated per VOP. Therefore, determining the minimum and maximum possible values of the shape refreshment need becomes a simple task. However, some of the obtained values can be artificially large, namely for the first VOP in a video object sequence when temporal concealment is considered. This is due to the fact that *TSCD* is set to 1 in this case (an extremely high value when compared to the *TSCD* values obtained in the following VOPs), because there is no past time instant that can be used for temporal concealment. If one of these artificially large values were chosen as the maximum possible value of the shape refreshment need, the algorithm would not work properly because the shape refreshment need would hardly ever reach such high values and two video objects with different shape refreshment needs would still be refreshed at too similar rates because their *ASRN*s would be rather similar in comparison with the maximum *ASRN* variation. Therefore, in order to eliminate these high values, it was decided to take as the maximum possible value of the shape refreshment need the value that guarantees that, approximately 95%, of the shape refreshment values are below it. And if, for any reason, the shape refreshment need exceeds this value, a saturated *SRN* value is adopted, which is not a problem since a very high *SRN* value results anyway. This experiment is repeated for all the considered combinations of error conditions and error concealment, for which the $[ASRN_{min}, ASRN_{max}]$ range is to be determined.

Since the values of $ASRN_{min}$ and $ASRN_{max}$ have to be determined for the specific combinations of channel error and error concealment conditions that are being considered, this will be done later in Section 5.3.3 after the test conditions have been defined.

C) Transformation of *ASRN* Values into *SRP* Values

After the two necessary ranges, $[1, SRP_{max}]$ and $[ASRN_{min}, ASRN_{max}]$, have been determined, it appears quite simple to map the $ASRN_{i-1}$ values of the individual video objects into the determined *SRP* range. However, this cannot be done by simply determining a linear function v that transforms the *ASRN* interval $[ASRN_{min}, ASRN_{max}]$ into the *SRP* interval $[1, SRP_{max}]$, by assigning SRP_{max} to $ASRN_{min}$ and 1 to $ASRN_{max}$ since this alone would not guarantee that the average of the individual *SRPs* would be equal to $ASRP_{target}$.

For this last condition to be verified, it is necessary that the average of the N individual $ASRN_{i-1}$ values, $ASRN_{avg}$, be also transformed into $ASRP_{target}$ by this same function v . This is equivalent to specifying a linear function v based on the three following conditions:

$$\begin{aligned} v(ASRN_{min}) &= SRP_{max} \\ v(ASRN_{max}) &= 1 \\ v(ASRN_{avg}) &= ASRP_{target} \end{aligned} \quad . \quad (5.6)$$

5. Adaptive Object-based Video Coding Refreshment Scheme

Since this is the same as determining the equation of a straight line based on three points, it becomes easy to understand that it is not always possible; it is only possible when the three points at hand are aligned. Therefore, when the three conditions cannot be satisfied simultaneously, only two of these conditions shall be used to determine the v function. One of the conditions has to be $v(ASRN_{avg}) = ASRP_{target}$, because this is essential to guarantee that the average of the individual $SRPs$ is equal to $ASRP_{target}$. The other is chosen among the remaining two conditions such that the $ASRN$ range is transformed into a sub-interval of the SRP range; otherwise, SRP values greater than SRP_{max} or lower than 1 can be obtained, which is undesirable. This leads to the three situations illustrated in Figure 5.2, which are used according to the explanation below.

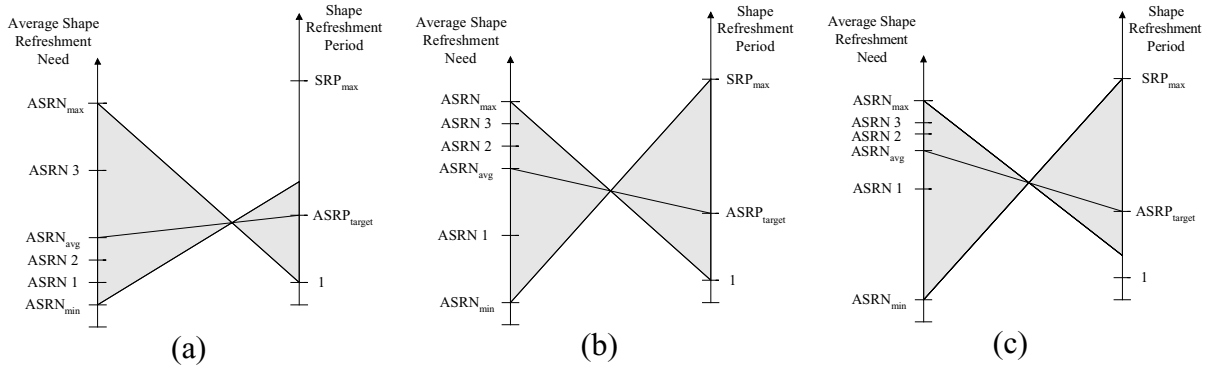


Figure 5.2 – Linear transformation of $ASRN_{i-1}$ values into SRP_i values: (a) Usage of lower SRP range; (b) Usage of full SRP range; (c) Usage of upper SRP range

At each time instant i , the linear transformation to be used to transform the $ASRN$ range into a sub-interval of the SRP range is adaptively chosen from the three situations shown in Figure 5.2, depending on the individual $ASRN_{i-1}$ values as follows:

- **Usage of lower SRP range** – If the individual $ASRN_{i-1}$ values satisfy the following condition:

$$\frac{ASRN_{avg} - ASRN_{min}}{ASRN_{range}} < \frac{SRP_{max} - ASRP_{target}}{SRP_{range}} \quad (5.7)$$

where $ASRN_{range} = ASRN_{max} - ASRN_{min}$ and $SRP_{range} = SRP_{max} - 1$, then the situation corresponds to the one illustrated in Figure 5.2 (a), where the linear transformation assigns 1 to $ASRN_{max}$ and $ASRP_{target}$ to $ASRN_{avg}$. Due to this, only the lower part of the SRP range will be used. In this case, the SRP_i vector can be determined by:

$$\mathbf{SRP}_i = ASRP_{target} \cdot \mathbf{U} + \text{Round} \left[\left(ASRN_{avg} \cdot \mathbf{U} - \mathbf{ASRN}_{i-1} \right) \cdot \frac{ASRP_{target} - 1}{ASRN_{max} - ASRN_{avg}} \right], \quad (5.8)$$

where \mathbf{U} is the unity vector (i.e., a vector where all elements are 1) with dimension N and $\text{Round}()$ is a function that rounds to the nearest integer all the components of the argument vector ($x.5$ is rounded up to $x+1$), which is used to guarantee that all the elements of the \mathbf{SRP}_i vector are integers.

- **Usage of full SRP range** – If the individual $ASRN_{i-1}$ values satisfy the following condition:

$$\frac{ASRN_{avg} - ASRN_{min}}{ASRN_{range}} = \frac{SRP_{max} - ASRP_{target}}{SRP_{range}}. \quad (5.9)$$

where $ASRN_{range}$ and SRP_{range} have the same meaning as above, then the situation is the one illustrated in Figure 5.2 (b), where the linear transformation assigns 1 to $ASRN_{max}$, $ASRP_{target}$ to $ASRN_{avg}$ and SRP_{max} to $ASRN_{min}$. Here, the full SRP range will be used. In this case, the SRP_i vector can be determined by:

$$SRP_i = ASRP_{target} \cdot U + Round \left[(ASRN_{avg} \cdot U - ASRN_{i-1}) \cdot \frac{SRP_{max} - 1}{ASRN_{max} - ASRN_{min}} \right], \quad (5.10)$$

where U and $Round()$ have the same definition as in Equation (5.8).

- **Usage of upper SRP range** – Finally, if the individual $ASRN_{i-1}$ values verify the following condition:

$$\frac{ASRN_{avg} - ASRN_{min}}{ASRN_{range}} > \frac{SRP_{max} - ASRP_{target}}{SRP_{range}}. \quad (5.11)$$

where $ASRN_{range}$ and SRP_{range} have the same meaning as above, then the situation is the one illustrated in Figure 5.2 (c), where the linear transformation assigns SRP_{max} to $ASRN_{min}$ and $ASRP_{target}$ to $ASRN_{avg}$. Due to this, only the upper part of the SRP range will be used. In this case, the SRP_i vector can be determined by:

$$SRP_i = ASRP_{target} \cdot U + Round \left[(ASRN_{avg} \cdot U - ASRN_{i-1}) \cdot \frac{SRP_{max} - ASRP_{target}}{ASRN_{avg} - ASRN_{min}} \right], \quad (5.12)$$

where U and $Round()$ have the same definition as in Equation (5.8).

The consideration of three different cases in terms of $ASRN$ to SRP mapping has to be done in order to guarantee that, at any given time instant, the $ASRN$ range is linearly transformed to the greatest possible sub-interval of the SRP range, while guaranteeing that the average of the individual $SRPs$ is kept equal to $ASRP_{target}$. This ensures that the variation of the SRP values around $ASRP_{target}$ is significant and adequate to the situation in question; otherwise all the video objects would have very similar (or even equal) SRP values, which would mean that no improvements would be visible with the proposed intra refreshment algorithm.

5.2.3.1.2

Individual Shape Refreshment Decision

After the individual SRP_i values have been determined as described above, it is then possible to determine which VOPs will have their shape refreshed at time instant i . This information will then be fed to the encoding module in the form of the Shape Refreshment Decision vector (or SRD_i vector) defined as:

$$SRD_i^T = [SRD_i^1 \quad SRD_i^2 \quad \dots \quad SRD_i^N], \quad (5.13)$$

where N is the number of objects in the scene and

$$SRD_i^j = \begin{cases} 0, & \text{if the shape of video object } j \text{ is not to be refreshed at time instant } i \\ 1, & \text{if the shape of video object } j \text{ is to be refreshed at time instant } i \end{cases}. \quad (5.14)$$

5. Adaptive Object-based Video Coding Refreshment Scheme

In order to determine the \mathbf{SRD}_i vector, a vector storing the index of the last VOP, for each object, which had its shape refreshed (i.e. the Last Refreshed Shape vector, \mathbf{LRS}_i) is used:

$$\mathbf{LRS}_i^T = [LRS_i^1 \quad LRS_i^2 \quad \dots \quad LRS_i^N] \quad (5.15)$$

where N is the number of objects in the scene.

For each object, to determine if a given VOP shall have its shape refreshed or not, a simple test is used: if the number of elapsed VOPs since the last refreshed shape is greater than or equal to the current SRP_i value, then the shape should be refreshed. Therefore, the shape refreshment decision vector, including the refreshment decisions for all the objects in the scene, can be determined as follows:

$$\mathbf{SRD}_i = V((i \cdot \mathbf{U} - \mathbf{LRS}_i) - \mathbf{SRP}_i), \quad (5.16)$$

where \mathbf{U} is the same as defined in Equation (5.8), i is the current VOP number and $V()$ is a function that takes as argument a vector whose components are integer values and is defined as:

$$V([n_1 \quad n_2 \quad \dots \quad n_N]^T) = [s(n_1) \quad s(n_2) \quad \dots \quad s(n_N)]^T \quad (5.17)$$

where $s(n)$ is the discrete unit step function defined as:

$$s(n) = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases} \quad (5.18)$$

If the used rate control algorithm allows for VOP skipping, then the shape refreshment decision may have to be overridden. This can happen if the rate control module decides to skip a given VOP that was supposed to have its shape refreshed. In this case, all the refreshment decision module has to do is reset to zero the appropriate position in the \mathbf{SRD}_i vector; in this way, the shape of the object in question will not be considered refreshed since that VOP is not transmitted and it will be refreshed in the future, very likely at the next time instant, if its instantaneous shape refreshment period does not increase by more than 1.

5.2.3.2 Texture Intra Refreshment Decision

In terms of the texture, the refreshment decision module receives as input as many \mathbf{ATRN}_{i-1} vectors as there are video objects in the scene (N) and the number of texture macroblocks to be refreshed ($NTMR$) at each time instant specified by the user. Based on these parameters, the refreshment decision module has to decide which macroblocks for which VOPs will be refreshed at the next time instant i . This decision is made based on the principle that not all the macroblocks have equal texture refreshment needs and, therefore, should have different intra refreshment rates.

In order to make these decisions, the refreshment decision module goes through the following two steps:

1. **Distribution of refreshment macroblocks among the various VOPs** – The texture refreshment decision module determines the number of texture macroblocks to be refreshed for each VOP in the scene, while guaranteeing that the total number of refreshed macroblocks at that time instant is equal to the $NTMR$ value specified by the user. At time instant i , the number of refreshment macroblocks assigned to each VOP is computed based on the average $ATRN_{i-1}$ values for each VOP.

2. **Distribution of refreshment macroblocks within the VOP** – The texture refreshment decision module decides which macroblocks within each VOP will be refreshed at time instant i . This is done while considering the number of refreshment macroblocks assigned to each VOP in the previous step.

Each of these two steps is explained, in detail, below.

5.2.3.2.1 Distribution of Refreshment Macroblocks among the Various VOPs

In order to distribute the refreshment macroblocks among the various VOPs, at time instant i , the refreshment decision module has first to go through the N input vectors and determine the average ATR_{i-1} value for each video object. Based on these values, it decides how many of the $NTMR$ macroblocks shall be assigned to each VOP. This is simply done by using the following expression:

$$NRM_i^j = Round \left[\frac{AATR_{i-1}^j}{\sum_{k=0}^{N-1} AATR_{i-1}^k} \cdot NTMR \right], \quad (5.19)$$

where j is the index corresponding to the number of the video object in the scene, $AATR_{i-1}^j$ is the average of the ATR_{i-1} values of all the macroblocks in video object j , NRM_i^j is the Number of Refreshment Macroblocks (NRM) assigned to video object j at time i , and $Round()$ is the scalar version of the function used in Equation (5.8).

This, alone, does not guarantee that all the macroblocks in the scene will eventually be refreshed because a video object whose macroblocks have very low texture refreshment needs might always have no refreshment macroblocks assigned to it. Since this situation can potentially lead to indefinitely long error propagations, it should be avoided altogether. After all, even with the low texture refreshment need, it is possible that one or more macroblocks in such a video object be corrupted and errors still remain after concealment. Therefore, an additional rule had to be added to overcome this situation: each video object will have, at least, a refreshment rate of one macroblock per VOP. This way, at each time instant and before the refreshment decision module selects the number of macroblocks to be assigned to each video object, one macroblock is automatically assigned to each video object, independently of the texture refreshment need values, thus guaranteeing that all the macroblocks do eventually get refreshed along time. After this, the remaining macroblocks ($NTMR$ subtracted of one refreshment macroblock per VOP) are distributed among the various video objects by the refreshment decision module. This is equivalent to using the following expression to assign the number of refreshment macroblocks to each VOP, instead of Equation (5.19):

$$NRM_i^j = Round \left[\frac{AATR_{i-1}^j}{\sum_{k=0}^{N-1} AATR_{i-1}^k} \cdot (NTMR - N) \right] + 1, \quad (5.20)$$

where j , $AATR_{i-1}^j$, NRM_i^j and $Round()$ have the same meaning as in Equation (5.19). Of course, the equation is valid under the assumption that $NTMR$ is higher than N .

5. Adaptive Object-based Video Coding Refreshment Scheme

5.2.3.2.2

Distribution of Refreshment Macroblocks within the VOP

After the number of macroblocks to be refreshed has been assigned to each VOP, the refreshment decision module has to distribute those refreshment macroblocks within each VOP, which shall be done in a cyclic manner. This means that, for video object j at time instant i , the intra refreshment procedure resumes where it left off in the previous time instant, by refreshing the next NRM_i^j non-transparent macroblocks, scanning the VOP from the top-left to the bottom-right corner. When the end of the VOP is reached, refreshment is restarted at the top-left corner.

As example, consider a given video object, which had 4 refreshment macroblocks assigned to it at time instant i . If, at time instant $i-1$, the last macroblock to be refreshed was macroblock 23, then, in the current VOP, macroblocks 24, 25, 27 and 28 (assuming that macroblock 26 is transparent) will be refreshed.

The output of this module is N Texture Refreshment Decision vectors, \mathbf{TRD}_i , one per object, specifying the macroblocks that are to be refreshed at time instant i , defined as:

$$\mathbf{TRD}_i^T = [TRD_i^1 \quad TRD_i^2 \quad \dots \quad TRD_i^M], \quad (5.21)$$

where M is the number of macroblocks in the video object at time instant i , and

$$TRD_i^k = \begin{cases} 0, & \text{if macroblock } k \text{ of the video object is not to be refreshed at time instant } i \\ 1, & \text{if macroblock } k \text{ of the video object is to be refreshed at time instant } i \end{cases} \quad (5.22)$$

If the used rate control algorithm allows for VOP or macroblock skipping, then the texture refreshment decision may have to be overridden for the macroblocks of the VOPs that have to be skipped or for the macroblocks skipped within the VOP. If a VOP is skipped, the number of macroblocks to be intra refreshed belonging to the skipped VOP should be redistributed among the remaining video objects, by using the same method described above. All the positions in the \mathbf{TRD}_i vector corresponding to the skipped VOP should be set to zero.

5.2.4 Step-by-Step Algorithm Specification

In conclusion, the intra refreshment algorithm proposed in this chapter includes the following steps:

1. **Initialization** – Before the encoding itself starts, it is necessary to create and initialize the necessary data structures; moreover, values for the $ASRP_{target}$ and $NTMR$ parameters have to be defined.
2. **Intra refreshment decision** – For the first VOP, both shape and texture must be intra coded; therefore go directly to step 3. Otherwise, decide if shape and/or texture intra refreshment should be performed:
 - **Shape** – Decide if any shape of the video objects for time instant i has to be intra refreshed. For this, the individual shape refreshment decision method presented before is used, filling the \mathbf{SRD}_i vector.
 - **Texture** – Decide which texture macroblocks for the various VOPs for instant i are to be refreshed. For this, the texture intra refreshment decision method presented before is used, filling the N \mathbf{TRD}_i vectors.

3. **Encoding** – Encode all the VOPs that have to be encoded for a certain time instant. The VOPs are encoded while taking into account the corresponding decisions related to shape and texture intra refreshment. If the encoder decides that the shape or any texture macroblocks are to be intra coded (e.g. because a smaller number of bits would be generated in a scene cut), independently of the refreshment algorithm, this will still be done. If this happens for the shape data, the LRS_i vector has to be updated, which is done by sending a message to the refreshment decision module.
4. **Computation of refreshment needs** – For each VOP, the new shape refreshment need is computed and is used to update the individual $ASRN$ values. As for the texture data, the individual macroblock refreshment need values are computed and the $ATRN$ value for each macroblock is updated.
5. **Continue or end** – If the last VOP has not been reached, go back to step 2 in order to process the next VOP. Otherwise, stop here.

Finally, after this step-by-step specification of the intra refreshment algorithm proposed in this chapter, it still remains to evaluate its performance, which shall be done in the next section.

5.3 Performance Evaluation

The proposed intra refreshment procedure is especially useful when a multi-object scenario is considered since the idea behind the technique is to adaptively distribute the available refreshment resources among the various objects in the scene, by assigning more resources to the objects with higher refreshment needs and vice-versa (according to the criteria previously defined). Therefore, the best way to evaluate the performance of the proposed intra refreshment technique is by comparing the results obtained in terms of decoded shape and texture quality when several multi-object sequences encoded using the proposed adaptive refreshment technique are sent through an error-prone channel with the results obtained when the same multi-object sequences are encoded with a simpler, less adaptive, intra refreshment scheme, taken here as reference.

5.3.1 Reference Refreshment Technique

As mentioned above, in order to evaluate the proposed adaptive refreshment technique a multi-object scenario is considered where the proposed technique is compared to a *reference refreshment technique*. Since the proposed refreshment technique is divided into shape and texture refreshment techniques, that can be independently used, the same should happen with the reference refreshment technique, meaning that reference shape and texture refreshment techniques must be defined.

5.3.1.1 Reference Shape Intra Refreshment Technique

The obvious solution for the reference shape intra refreshment technique is to use the same fixed periodic intra refreshment for all the video objects in the scene. In order to make the comparison with the adaptive refreshment technique meaningful, the same amount of refreshment resources should be used in both cases. Therefore, the value used for the fixed periodic intra refreshment shall be equal to the $ASRP_{target}$ value selected by the user for the adaptive intra refreshment technique.

5.3.1.2 Reference Texture Intra Refreshment Technique

In terms of texture, the obvious solution is to refresh a fixed number of macroblocks per time instant, while the distribution of the number of macroblocks to be refreshed among the various VOPs is done proportionally to their size (in terms of opaque shapels). Since this may not give an integer number of macroblocks per VOP, this value is rounded to the nearest integer. To distribute the assigned number of refresh macroblocks (i.e., N_{MB}) within the VOP itself, a cyclic scheme is used. This way the refreshment starts at the top left corner and the first N_{MB} non-transparent macroblocks are refreshed. At the next time instant, the following N_{MB} non-transparent macroblocks are refreshed and so on. It should be noticed that the value of N_{MB} may change from one time instant to the next, since the size of a video object (in terms of opaque shapels) may also change with time.

In order to make the comparison with the adaptive refreshment technique meaningful, the same amount of refreshment resources should be the same in both cases. Therefore, the total number of macroblocks to be refreshed at each time instant ($NTMR$) should be equal in both cases. However, due to the rounding described above, it is possible that the sum of the intra refreshment macroblocks assigned to all the VOPs for a certain time instant is not equal to $NTMR$ (it may be more or less). If it is less, this means that some of the $NTMR$ macroblocks were not assigned to any VOP and, therefore, will be assigned to some VOP at the next time instant. In order to avoid favoring one video object over the others, the distribution of the extra refreshment macroblocks is also cyclic, assigning one of the extra refreshment macroblocks to each VOP in a cyclic manner. For instance, consider that a given scene has four video objects and that, at a given time instant, 2 refreshment macroblocks were not assigned to any VOP. This way, at the following time instant, one of these macroblocks will be assigned to VOP 1 in the scene and the other will be assigned to VOP 2. The next time this happens, the distribution will start with VOP 3. If, on the other hand, the sum of the intra refreshment macroblocks assigned to all the VOPs exceeds $NTMR$, the same procedure will be followed but, instead of adding, intra refreshment macroblocks will be removed.

5.3.2 Evaluation Methodology

In order to compare the performance of the proposed adaptive intra refreshment scheme with the performance of the reference intra refreshment technique, a similar approach to the one used for the experiments carried out in the MPEG Error Resilience Ad Hoc Group is adopted [N1586]. Therefore, for both the technique being tested and the reference technique, the following three steps have to be performed:

1. **Bitstream generation** – For each test condition (i.e. a combination of test sequence, frame rate, spatial resolution, bit rate, channel error condition, etc.), the corresponding bitstreams are generated for the various video objects using the MoMuSys version of the MPEG-4 Reference Software [MPEG4-SW]³.
2. **Bitstream corruption** – Each of these bitstreams is sent through a simulated error-prone channel with certain adopted characteristics *fifty* times, thus corrupting it with *fifty* different error patterns, in order to overcome the problems associated with having errors in rather special locations if too few corrupted bitstreams are used; bitstream corruption is performed with the error generating software provided to MPEG by NTT DoCoMo [ErrorGen].

³ It is worthwhile to notice that it was the author of this Thesis that implemented the MPEG-4 error resilient coding techniques in this MPEG-4 Reference Software.

3. **Resilient decoding** – For each video object, the fifty corrupted bitstreams are decoded with the agreed upon error resilience and concealment techniques, while generating the adopted quality metrics.

By following these three steps for both refreshment techniques, two sets of results are obtained (one for the intra refreshment technique being tested and another for the reference technique), which can then be used to compare them and conclude on their relative benefits.

5.3.2.1 Error Resilience and Concealment Techniques

Since the bitstreams are sent through a simulated error-prone channel, the used decoder has to take some kind of action to deal with the errors; these actions do not impact interoperability and as such are not normalized in MPEG-4. The decoder can act at the three following levels:

- **Error detection** – In terms of error detection, syntactic and semantic inconsistencies, such as illegal variable length codes, more than 64 DCT coefficients per 8×8 block, unexpected resynchronization markers, etc., are checked for in order to detect if errors have occurred.
- **Error localization** – As for error localization, video packet resynchronization, as well as data partitioning [MPEG4-V], are used. Although reversible variable length codes (RVLC) were used for the texture data, the reversible decoding capability was not used to minimize the amount of lost texture data. This was done to avoid unduly counting on too sophisticated error resilience techniques at the decoder.
- **Error concealment** – Finally, in terms of error concealment, a technique taking advantage of the data partitioning has been used: if the partition with the shape and motion data is lost, the entire video packet is discarded and replaced with the corresponding data (shape and texture) from the previous VOP; if the error is found in the texture partition, then the correct shape data and motion vectors from the shape and motion partition are used, namely to motion compensate the texture from the previous VOP.

The used concealment scheme involves only temporal concealment and, therefore, the shape and texture concealment difficulties used to compute the shape and texture refreshment needs depend only on the temporal concealment difficulty. Thus, Equation (4.3) becomes

$$SRN_i = SEV_i \times TSCD_i, \quad (5.23)$$

and Equation (4.46) becomes

$$TRN_i^k = TEV_i^k \times TTCD_i^k. \quad (5.24)$$

By showing results with this simple and commonly used (temporal) error concealment technique (such as in the experiments carried out in the MPEG Error Resilience Ad Hoc Group [N1586]) instead of a more complex spatio-temporal concealment solution, it is intended to show that the proposed technique can be successfully applied to any off-the-self MPEG-4 encoder and does not depend on sophisticated error concealment techniques at the decoder side.

5.3.2.2 Video Object Quality Evaluation

In terms of video quality metrics, since object-based video is considered, metrics are needed to evaluate the quality of both the shape and texture of the decoded video objects. Since the

5. Adaptive Object-based Video Coding Refreshment Scheme

objective of the present work is not to define coding quality metrics for arbitrarily shaped video objects, the same metrics that were used by the MPEG Error Resilience Ad Hoc Group for the error resilience experiments on arbitrarily shaped video objects will be used [Brady97]. Although these objective metrics have limitations in terms of reflecting the subjective impact of the coded video objects, they are undoubtedly simple to compute and make the direct comparison between techniques possible without the need for difficult and very time and resources consuming subjective tests.

Thus, to evaluate the shape quality, the Dn metric is used [N1584]. This metric expresses the fraction of the shape area that is distorted and is defined as:

$$Dn = \frac{\text{number of different shapels between decoded and original VOPs}}{\text{number of opaque shapels in original VOP}}. \quad (5.25)$$

Dn can also be expressed as a percentage,

$$Dn[\%] = 100 \times Dn. \quad (5.26)$$

As for the texture quality, a $PSNR$ metric is used. However, since arbitrarily shaped video objects are used, the definition of $PSNR$ has to be adjusted and thus, instead of being computed over all the pixels in a rectangular frame, it is computed over the pixels that belong to both the decoded VOP being evaluated and the original VOP (the parts where there is no overlapping due to errors are not taken into account). This modified version of $PSNR$ is also used by MPEG [MPEG4-SW] and is defined as:

$$PSNR = 10 \log_{10} \left(\frac{1.5 \times 255^2}{MSE_Y + 0.25 \cdot MSE_{Cr} + 0.25 \cdot MSE_{Cb}} \right) \quad (5.27)$$

where MSE_Y , MSE_{Cr} and MSE_{Cb} are the mean square errors of the three components (Y , Cr and Cb) between the decoded VOP and the original VOP, computed only for the pixels that belong to both. The mean square error for a given component X is given by:

$$MSE_X = \frac{1}{N_{opaque}} \cdot \sum_{\substack{\text{opaque} \\ \text{shapels}}} (X(x, y) - \bar{X}(x, y))^2 \quad (5.28)$$

where N_{opaque} is the number of opaque pixels overlapping in the original and decoded VOPs, $X(x, y)$ is the value of component X in the original VOP with (x, y) coordinates, and $\bar{X}(x, y)$ is the value of component X in the decoded VOP with (x, y) coordinates.

While decoding a bitstream, it is possible for the decoder to lose a complete VOP. For instance, this can happen due to errors in the VOP header data. In this case, the lost VOP is replaced with the previously decoded VOP, which will be used for Dn and $PSNR$ computations. This guarantees that, at the end, the number of original and decoded VOPs is the same and thus a synchronized comparison is performed.

Since each decoded bitstream produces one Dn and one $PSNR$ value for each decoded VOP, it is not easy to compare the decoded video quality achieved with two different corrupted bitstreams because it involves comparing several values. Therefore, in the following, the video quality achieved with one corrupted bitstream will be expressed by the average over all the decoded VOPs of the Dn and $PSNR$ values, given by

$$Dn = \frac{1}{N_{VOP}} \sum_{i=0}^{N_{VOP}-1} Dn_i \quad (5.29)$$

$$PSNR = \frac{1}{N_{VOP}} \sum_{i=0}^{N_{VOP}-1} PSNR_i \quad (5.30)$$

where N_{VOP} is the total number of VOPs in the test sequence (sub-sampled at the considered VOP rate) and i is the VOP number.

Moreover, fifty bitstreams are decoded for each test condition and for each video object, instead of just one, to account for the random nature of the errors. Therefore, a method is necessary to combine the Dn and $PSNR$ results from the fifty decoded bitstreams into a single result that can be used to express the decoded video quality associated with that test condition (later referred to as *individual* Dn and $PSNR$, respectively). The plain average (i.e., mean) is commonly used when testing decoder concealment techniques. In that case, each one of the corrupted bitstreams is decoded with the different concealment techniques being compared. This means that the results obtained for the different decoder techniques are based exactly on the same corrupted bitstreams. This way, if a particular error pattern is especially critical (e.g., errors in the header data), all the techniques being compared will be equally affected and will have to deal with it. This guarantees that results corresponding to a given technique will not be strongly biased when compared to the others. However, the adaptive and reference refreshment techniques used here are encoder techniques, which means that they are going to be compared based on differently corrupted streams (the error corruption engine randomly distributes the errors for each run) obtained from different uncorrupted streams. Therefore, a technique which is generally better than another can perform worse for a specific case just because the errors hit different parts of the bitstreams and sometimes may hit more critical syntactic zones. In particular, one bitstream hit by errors at very critical places can strongly bias the mean. To illustrate this, a given error pattern can be considered. When this error pattern is applied to two different uncorrupted bitstreams, obtained because two different encoder techniques were used, the errors will surely corrupt different parts of the bitstreams, which can be very different in terms of criticality. For instance, it is possible that the bitstream produced with the reference refreshment technique has some easily concealable errors on the texture data, while the other one obtained with the adaptive refreshment technique has errors in very critical header data, such as the VOP size or temporal references. Therefore, it is expected that for this error pattern, the adaptive refreshment technique will lead to very poor results. Even if for all the remaining error patterns, the proposed refreshment technique outperforms the reference technique, the poor results obtained with that only error pattern will strongly bias the global results, apparently showing that the adaptive technique is inferior to the reference technique. To avoid this type of bias, the median will be used here instead of the mean, because it is less vulnerable to *outliers*; thus the decoded video quality associated with a given test condition will be given by

$$Dn = median_k \{Dn_k\} \quad (5.31)$$

$$PSNR = median_k \{PSNR_k\} \quad (5.32)$$

where k is the index corresponding to the bitstream number (within the fifty), and Dn_k and $PSNR_k$ are the Dn and $PSNR$ temporal averages, respectively, for bitstream k .

5.3.2.3 Video Scene Quality Evaluation

In the previous section, the video object quality metrics to be used in the following were defined. However, the proposed adaptive refreshment technique tries to improve the individual quality of certain video objects at the expense of others in the scene. This means that the quality of some video objects will increase, while the quality of others may actually

5. Adaptive Object-based Video Coding Refreshment Scheme

decrease. Therefore, the proposed adaptive intra refreshment technique should be compared to the reference intra refreshment technique based on the overall video scene quality achieved, instead of just comparing the quality of the individual video objects, which can be quite a difficult task since there are positive and negative variations with a balance that it is difficult to make. For this, a method to combine the shape and texture quality metrics of the individual video objects (Dn and $PSNR$) into global shape and texture metrics (one for Dn and another for $PSNR$) is needed.

In [Correia00], a method is developed to evaluate the overall segmentation quality of a scene, defined as a weighted sum of the several individual video object segmentation qualities. The reasoning here is that not all video objects contribute in the same way to the video segmentation quality since they capture the human attention differently due to the different relevance that they have in the scene. Notice that a similar reasoning may clearly apply to the overall shape and texture scene qualities.

In [Correia00], the *absolute contextual relevance* of the video objects in the scene, which expresses the importance that a given video object within a scene has to the user, is defined as a combination of elementary metrics which take into account the HVS characteristics and the factors which influence eye movement and attract human attention. The contextual relevance values (varying between 0 and 1) are computed at each time instant and can then be averaged over time to produce a single value for each video object.

The elementary metrics that are used to compute the absolute contextual relevance can be classified as low-level or high-level, depending on the level of semantic information that is needed to compute them, as follows:

- **Low-level metrics** – To compute these metrics, a semantic understanding of the content and its context is not required. The metrics included in this category are:
 - **Motion activity** – According to the HVS, this is one of the most important features, meaning that moving objects usually attract a lot of human attention.
 - **Position** – The position of the object in the scene is also of the utmost importance, since the fovea is usually directed to the center of the image around 25% of the time.
 - **Contrast to neighbors** – Objects that exhibit a large contrast to their neighbors usually stand out more and, therefore, deserve more attention from the user.
 - **Size** – Larger objects usually attract more attention from the user because they occupy a larger percentage of the visual field.
 - **Shape and orientation** – The HVS tends to be attracted to some particular shapes, such as circular shapes (circularity), long and thin shapes (elongation and compactness) and shapes with horizontal or vertical orientation (orientation).
 - **Brightness and redness** – Bright and colorful objects usually attract more attention from the user, especially if they are red.
 - **Object complexity** – Objects with a more complex/detailed spatial content also tend to attract the attention of the user.
- **High-level metrics** – To compute these metrics, a semantic understanding of the content and its context is needed. The metrics included in this category are:
 - **Background** – A given object can belong either to the background or to the foreground, with those in the foreground deserving more attention from the user.

- **Type of object** – Certain video objects deserve more attention from the user due to their semantic nature; for instance, human bodies and faces usually get more attention than other types of objects.

By considering the metrics above, which are defined in [Correia00], the Absolute Contextual Relevance (ACR) for video object j at a given time instant can be computed from:

$$ACR_j = 0.3 \cdot MA + 0.25 \cdot COMP + 0.13 \cdot HL + 0.1 \cdot SHP + 0.085 \cdot BR + 0.045 \cdot (CONT + POS + SZ) \quad (5.33)$$

where MA , $COMP$, HL , SHP , BR , $CONT$, POS and SZ correspond to the motion activity, complexity, high level, shape and orientation, brightness and redness, contrast, position and size metric values, respectively.

However, the absolute contextual relevance of the video objects in the scene cannot be readily used as weights when computing the overall scene quality because their sum is not equal to 1 (i.e. the scale would be altered). For this, the *relative contextual relevance* of the objects in the scene is defined, which is basically a normalized version of the absolute contextual relevance. It expresses the relative importance to the user of a given video object within a scene and can be computed as:

$$RCR_j = \frac{ACR_j}{\sum_{m=1}^N ACR_m} \quad (5.34)$$

where RCR_j is the relative contextual relevance of video object j , and ACR_j is the absolute contextual relevance of video object j , normalized to the $[0,1]$ range, with value 1 corresponding to the highest relevance, according to Equation (5.33).

By adapting the method developed in [Correia00] to the present work, it becomes possible to evaluate the global shape and texture quality of a scene, as follows:

$$Dn_{global} = \sum_{j=0}^{N-1} ARCR_j \cdot Dn_j \quad (5.35)$$

$$PSNR_{global} = \sum_{j=0}^{N-1} ARCR_j \cdot PSNR_j \quad (5.36)$$

where j is the index corresponding to the video object number within the scene, N is the number of video objects present in the scene, Dn_j is the Dn metric associated with video object j , $PSNR_j$ is the $PSNR$ metric associated with video object j , and $ARCR_j$ is the temporal average of the relative contextual relevance of video object j .

It is important to notice that although the relevance of each object in the scene is here being used to evaluate the overall scene quality, this relevance was not taken into account while distributing the available refreshment resources among the various objects in the scene. This could be done but this approach was not adopted to avoid making the refreshment need computations at the encoder, which have to be performed for all VOPs of all video objects, too complex.

5.3.3 Test Conditions

Since a multi-object scenario is considered, all the video sequences used to generate results must have at least two objects. Therefore, three multi-object sequences with different

5. Adaptive Object-based Video Coding Refreshment Scheme

characteristics in terms of spatial detail and amount of motion have been chosen from the MPEG-4 test set, which are the Weather, Coastguard, and Stefan sequences.

The major characteristics of these sequences are summarized in Table 5.1. In addition, frame samples are provided in Figure 5.3. For further details on the sequences, the reader can refer to Annex A.

Table 5.1 – Selected video test sequences

Sequence	Number of video objects	Number of VOPs	Spatial detail	Amount of motion
Weather	2: Weather Map, Weather Girl	300	Low	Low
Coastguard	4: Water, Large Boat, Small Boat, Shore	300	Medium	Medium
Stefan	2: Background, Tennis Player	300	High	High



Figure 5.3 – Test sequence samples: (a) Weather; (b) Coastguard; (c) Stefan

Results will be shown for the chosen sequences encoded at the lowest acceptable frame rate, which is considered to be 15 fps for Stefan and 10 fps for the remaining sequences, QCIF format and lossless shape coding. These conditions are used because they correspond to the highest shape refreshment needs and, therefore, should lead to noticeable improvements. As for the bit rates used for each sequence, they are summarized in Table 5.2. These bit rates are considered reasonable values and should provide acceptable video quality with these coding conditions. The distribution of the bit rate among the various video objects in the sequence is done using the joint rate control scheme implemented in the MoMuSys version of the MPEG-4 Reference Software [MPEG4-SW]. This joint rate control scheme distributes the available bit rate among the various VOPs using as criteria the size, the amount of motion and the texture variation with respect to the past. In terms of video packet size, it was decided to use an average of 4 VPs per VOP, which is the most commonly used value, namely adopted by the MPEG Error Resilience Ad Hoc Group [N1586]. To do this, the amount of bits assigned to each video object by the joint rate control has to be considered, from which it is easy to compute the average number of bits spent on each VOP; the VP size threshold (see Annex E of [MPEG4-V]) is thus chosen to be one fourth of that value.

Table 5.2 – Bit rate used for each video test sequence

Sequence	Bit rate (bps)
Weather	56000
Coastguard	96000
Stefan	192000

For the QCIF format and the adopted frame rates (10 fps for the Coastguard and Weather sequences and 15 fps for the Stefan sequence), the average relative contextual relevance values for the various video objects, computed using Equation (5.34), are given in Table 5.3. These values will be used to determine the scene shape and texture quality metrics in the following experiments.

Table 5.3 – Average relative contextual relevance for the video objects in the test sequences (QCIF format)

Sequence	Video object	Average relative contextual relevance
Weather	Weather Map	0.366
	Weather Girl	0.634
Coastguard	Water	0.196
	Large Boat	0.305
	Small Boat	0.292
	Shore	0.207
Stefan	Background	0.433
	Tennis Player	0.567

In terms of channel errors, several types and amounts were considered, corresponding to different types of networks. These Error Conditions (EC), summarized in Table 5.4, are the same that were used by the MPEG Error Resilience Ad Hoc Group [N1586] and include: uniformly distributed errors, burst errors with different bit error rates (BER) and burst lengths, and packet losses with different packet loss rates (PLR) and multiplex packet payload sizes.

Table 5.4 – Test error conditions

Error condition
1: Uniform Errors ($\text{BER} = 10^{-3}$)
2: Burst Errors ($\text{BER} = 10^{-2}$, burst length: 1 ms)
3: Burst Errors ($\text{BER} = 10^{-2}$, burst length: 10 ms)
4: Burst Errors ($\text{BER} = 10^{-2}$, burst length: 20 ms)
5: Burst Errors ($\text{BER} = 10^{-3}$, burst length: 1 ms)
6: Burst Errors ($\text{BER} = 10^{-3}$, burst length: 10 ms)
7: Packet Loss ($\text{PLR} = 10^{-2}$, multiplex packet payload sizes: 96, 200, 400)
8: Packet Loss ($\text{PLR} = 3 \cdot 10^{-2}$, multiplex packet payload sizes: 96, 200, 400)

For each combination of the error conditions in Table 5.4 and the error concealment scheme described in Section 5.3.2.1, the ASRN_{\min} and ASRN_{\max} values have to be determined. This was done by proceeding as described in Section 5.2.3. For this, the same set of sequences

5. Adaptive Object-based Video Coding Refreshment Scheme

used for the results – Weather, Coastguard and Stefan – was used; these sequences were encoded under the conditions listed in Table 5.5, using lossless shape coding.

Table 5.5 – Coding conditions used to determine $ASRN_{min}$ and $ASRN_{max}$

Sequence	Video objects	Resolution	Frame rate	Bit rate (bps)
Weather	2	QCIF	10 fps	56000
Coastguard	4	QCIF	10 fps	96000
Stefan	2	QCIF	15 fps	192000

The minimum shape refreshment need values happen for objects that have exactly the same shape for several consecutive time instants, thus corresponding to a $TSCD$ and an $ASRN$ of zero (e.g. this is the case of the Logo object in the News sequence). Therefore, $ASRN_{min}$ is also zero for all error conditions as could be expected. As for the maximum shape refreshment need values, by proceeding as explained in Section 5.2.3 and choosing $ASRN_{max}$ as the value that guarantees that, approximately 95%, of the shape refreshment values are below it, the results given in Table 5.6 were obtained.

Table 5.6 – Maximum shape refreshment need for the various error conditions

Error condition	$ASRN_{max}$	Cumulative percentage (%)
1	0.15	95.13
2	0.022	94.84
3	0.0023	95.20
4	0.0011	94.93
5	0.0022	94.91
6	0.0002	95.71
7	0.0085	95.09
8	0.025	95.02

5.3.4 Simulation Results

Since each part of the proposed intra refreshment technique (shape and texture) can be used independently, in addition to their combined evaluation, they will also be evaluated individually in order to show that they can be beneficial independently of each other. Therefore, the following three sets of results will be analyzed:

- **Adaptive shape refreshment scheme alone** – For each test condition, each multi-object video test sequence is encoded twice: with the proposed adaptive shape intra refreshment scheme and with the reference shape intra refreshment scheme. For both cases, the reference texture intra refreshment technique is used.

- **Adaptive texture refreshment scheme alone** – For each test condition, each multi-object video test sequence is encoded twice: with the proposed adaptive texture intra refreshment scheme and with the reference texture intra refreshment scheme. For both cases, the reference shape intra refreshment technique is used.
- **Adaptive combined refreshment scheme** – For each test condition, each multi-object video test sequence is encoded twice: with the adaptive shape and texture intra refreshment schemes and with the reference shape and texture intra refreshment schemes.

5.3.4.1 Adaptive Shape Refreshment Scheme Results

In order to study the stand-alone performance of the shape refreshment part of the proposed adaptive intra refreshment algorithm, the reference texture refreshment scheme is adopted. In all of the following, a fixed refreshment rate of 12 macroblocks per time instant is used, which is the value that will be used in the following section where the texture refreshment algorithm is studied. This value, however, does not influence the shape quality results because lossless shape coding is used and the computation of the shape quality metrics does not depend on the texture data. Its main objective is to guarantee that the generated VPs are realistic (i.e. with some reasonably coded texture), in order to make the shape refreshment results more meaningful and allow for the study of the influence that shape refreshment has on the texture decoded data.

5.3.4.1.1 Encoder Results

In order to understand better how the decoded shape and texture video qualities within a given sequence are affected by the proposed shape intra refreshment technique, it would be interesting to know the evolution of the instantaneous shape refreshment period (which depends on the shape refreshment need) for the different video objects. This would make it possible to have an idea of the improvement to be expected because the achieved improvement is directly related to the disparity in terms of the shape refreshment period of the different video objects within a given sequence. Therefore, the evolution of the instantaneous shape refreshment periods is plotted in Figure 5.4, Figure 5.5 and Figure 5.6 for the Weather, Coastguard and Stefan sequences, respectively. It is important to remind the reader that these values correspond to a situation where only temporal error concealment is applied and thus the shape refreshment need only includes its temporal component. For these plots and for the remaining tests, $ASRP_{target}$ was set to 5. Instead of using this value, optimal $ASRP_{target}$ values could have been determined for each tested error condition; however, to simplify, a reasonable value acceptable for all error conditions was chosen. This $ASRP_{target}$ value imposes the shape refreshment resources to be used (i.e. the video objects will have their shapes refreshed every 5 VOPs, on average) and it is considered a good compromise in terms of the maximum allowed SRP value (the minimum is always 1). If this value were smaller, the used SRP range would also be smaller, meaning that the SRP values assigned to each video object would be too close and the advantages of the adaptive shape refreshment technique less evident. On the other hand, if the $ASRP_{target}$ value were greater, the used SRP range would be larger, meaning that some video objects would have a very high SRP which could compromise their quality. Anyway, since this value is to be determined before the coding starts, the best choice may be decided depending on the relevant conditions.

5. Adaptive Object-based Video Coding Refreshment Scheme

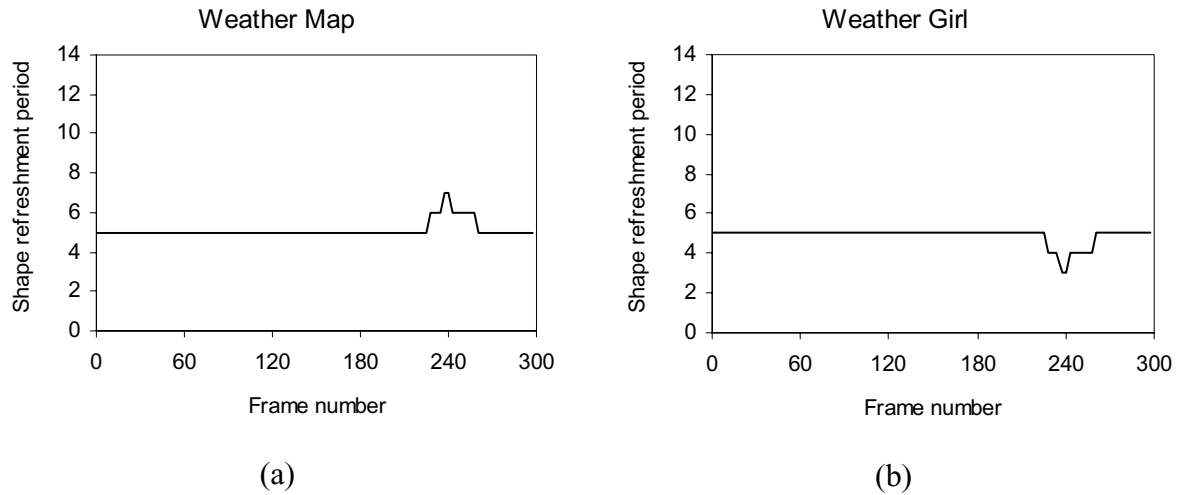


Figure 5.4 - Instantaneous SRP for the Weather sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 10 fps, EC5): (a) Weather Map; (b) Weather Girl

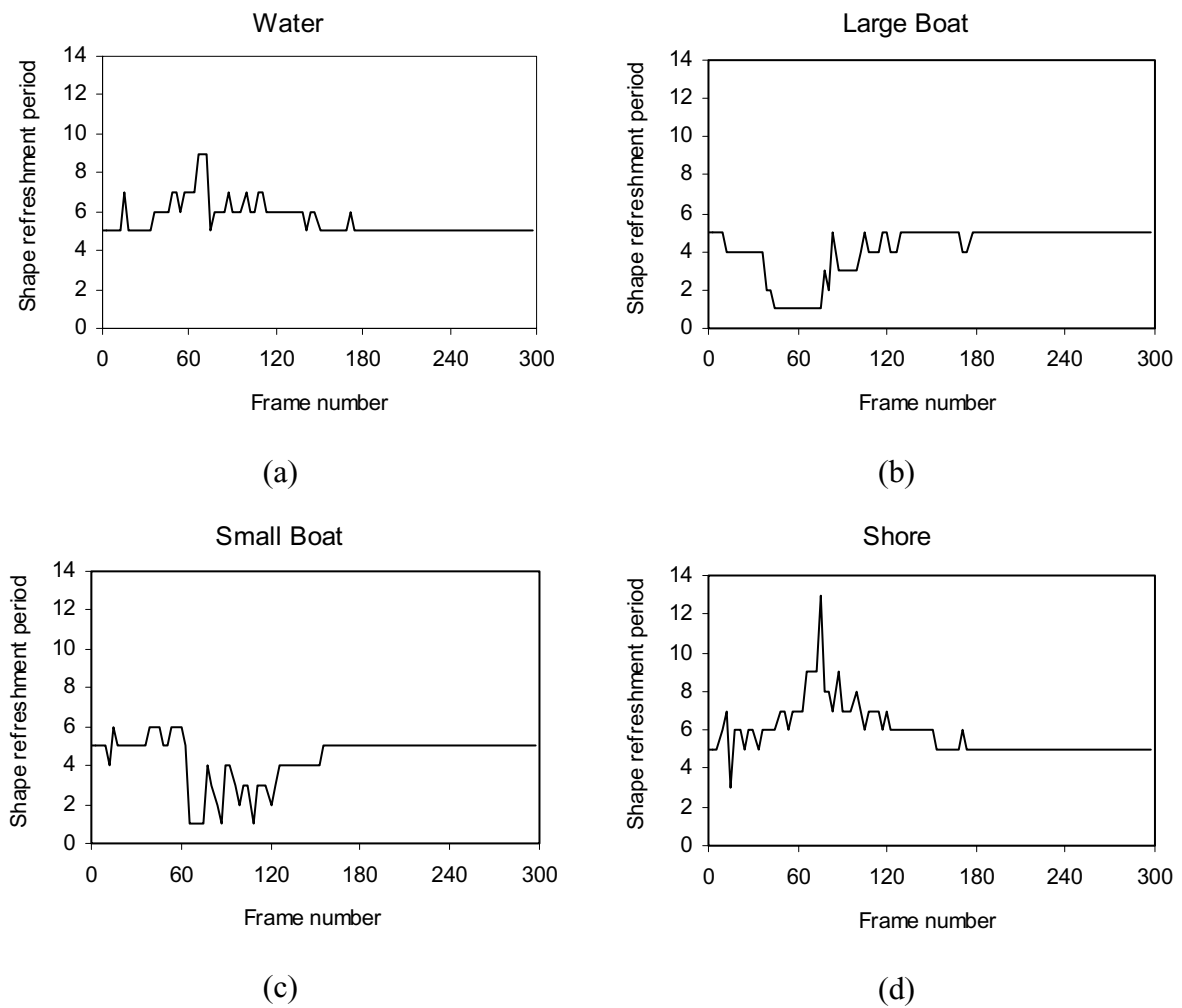


Figure 5.5 – Instantaneous SRP for the Coastguard sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 10 fps, EC5): (a) Water; (b) Large Boat; (c) Small Boat; (d) Shore

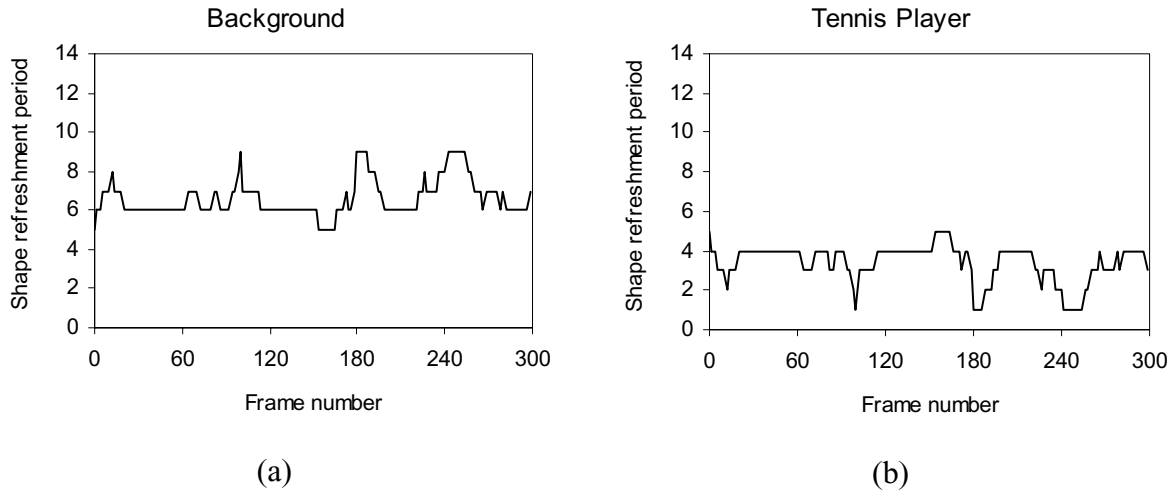


Figure 5.6 - Instantaneous SRP for the Stefan sequence video objects with $ASRP_{target}$ equal to 5 (QCIF, 15 fps, EC5): (a) Background; (b) Tennis Player

From these plots, it is possible to compare the test sequences in terms of the shape refreshment period disparity for the various video objects. This way, it becomes possible to rank the sequences according to the increasing disparity (this disparity is directly related to the expected improvement) in the SRP values of its various video objects:

- **Weather sequence** – The Weather sequence is the one whose video objects are closer in terms of shape refreshment need because the shape refreshment period of its two objects is also almost always constant, except for some time at the end where the Weather Girl object starts being refreshed more frequently. This corresponds to the part of the sequence where the Weather Girl turns sideways and starts pointing at the Weather Map, meaning that there is more motion and thus the temporal concealment adopted should have more difficulties. Therefore, it is expected that the adaptive shape refreshment scheme will perform only slightly better than the reference shape refreshment, improving the quality of the Weather Girl object at the expense of a slight quality decrease for the Weather Map object.
- **Coastguard sequence** – Next in the ranking, comes the Coastguard sequence, for which the video objects largely differ in terms of shape refreshment need and shape refreshment period during, approximately, the first half of the sequence. It is clear from the plot that the Small Boat and the Large Boat objects are more refreshed than the remaining two objects. In the second half of the sequence, however, all the objects have a constant shape refreshment period equal to $ASRP_{target}$. This is easily understandable if the shapes of the different objects are considered. In the second half of the sequence, the Small Boat has left the scene, leaving only its trail behind, and the camera is following the Large Boat. In terms of alpha planes, however, it should be noticed that for all the video objects they hardly change from frame to frame until the end of the sequence, implying a similar shape refreshment need since only temporal error concealment is being used. Therefore, some improvement is expected in the overall decoded video quality of the Small Boat and Large Boat objects when the adaptive shape refreshment is used.

5. Adaptive Object-based Video Coding Refreshment Scheme

- **Stefan sequence** – Finally, comes the Stefan sequence, during all of which the Tennis Player object is more refreshed than the Background object. This was expected, due to the very fast nature of the sequence and the size difference of the two objects (i.e. the shape of the Tennis Player object changes much more from frame to frame than the shape of the Background object). Therefore, a large improvement is expected for the Tennis Player object when the adaptive scheme is used instead of the reference refreshment scheme, very likely at the expense of a slight quality decrease for the Background object.

Although the shown plots correspond to error condition 5 (EC5), the same type of behavior was verified for the other error conditions, using the same test sequences. This happens because the $ASRN$ range is chosen according to the used error condition, through the procedure described in Section 5.2.3. This way, although the individual $ASRN_i$ values fall within very different sub-intervals of the $[0,1]$ range from one error condition to the other, they will be transformed into approximately the same shape refreshment periods, guaranteeing that the behavior of the shape refreshment algorithm is similar for all error conditions.

5.3.4.1.2

Decoder Results

After having analyzed the test sequences and their objects in terms of their shape refreshment period variations (using only temporal error concealment), it is time to study the decoder results showing how the video quality varies for each video object and for the whole scene when the adaptive shape refreshment scheme is used and check if this variation agrees with the preliminary analysis made using the encoder results. Since this section studies the shape refreshment part of the adaptive intra refreshment scheme, one might be tempted to show only the shape quality results (i.e. Dn); however, the texture quality results (i.e. $PSNR$) are also worthwhile to analyze since they depend on the shape quality in three ways:

- **Shape-texture bit rate trade-off** – When the shape refreshment period decreases (increases) for a given video object, the video object encoder has to spend more (less) bits on its (losslessly coded) shape data. Therefore, if the bit rate were fixed for each video object, this would mean that less (more) bits would be available for the texture data of that video object, decreasing (increasing) its $PSNR$ values. In this case, however, a joint rate control scheme is used to distribute the total fixed bit rate among the various video objects. This means that, at each time instant, after the lossless shape data has been encoded, the remaining bits are distributed among the texture of the various video objects, independently of the number of bits spent on the shape data of each video object. When the shape refreshment periods associated with the different video objects are changed (ones are increased while the other ones decreased), it is very likely that a different amount of bits will be spent on shape data at each time instant, implying that a different number of bits (higher or lower) will be available for distribution among the texture data. This can both mean an increase or a decrease in the video texture quality. However, since the number of bits spent on the shape data at a given time instant is not expected to change much when the shape refreshment periods of the various video objects are varied and since they are usually rather small when compared to the number of bits spent on the texture data, only rather small $PSNR$ changes should be attributed to this effect.
- **Decodability of texture dependent on correctness of shape** – Due to the object-based nature of MPEG-4 and its VP syntax, it is not possible to correctly decode the texture in a VP if the shape data has been corrupted. This explains why in the used error concealment scheme, a VP is completely discarded (texture data included) when the shape data is

found to be corrupted by errors. This dependence of the texture quality on the shape quality is further amplified by the effects described in Section 4.3, which may cause the texture in a VP being decoded to be thrown away because its shape cannot be correctly decoded due to a shape error in a past time instant. This way, when more shape data is corrupted with errors, more texture data will also be discarded. And, therefore, it is expected that when the shape quality improves due to the increase in shape refreshment, so will the texture quality. The opposite is also expected.

- **Texture PSNR for overlapping shapes** – The *PSNR* values depend on the shape quality because the *PSNR* is computed for the pixels that belong to both the decoded VOP and the original VOP, as explained in Section 5.3.2.2. If the shape of the decoded VOP is more severely corrupted, this means that a smaller set of pixels will belong to both this VOP and the original one, which means that the *PSNR* will be computed over a smaller set of pixels. This can both increase or decrease the *PSNR* value, depending on the quality of the pixels that remain common to both VOPs. Here, however, each VOP is being coded with constant texture quality (fixed quantization step) within the VOP and, therefore, it is expected that all the pixels in it will have a very similar quality. Thus, in the following results, only very small changes in the *PSNR* values should be attributed to this effect, since the same reference texture refreshment scheme is used in all cases.

For these reasons, both *Dn* and *PSNR* figures are given in the following tables, in order to evaluate the performance of the adaptive shape intra refreshment technique. For the various error conditions defined in Table 5.4, *individual Dn* and *PSNR* figures are given for the various video objects, as well as *global Dn* and *PSNR* figures for the whole scene. The *Dn* and *PSNR* values are given in Table 5.7 and Table 5.8 for the Weather sequence, in Table 5.9 and Table 5.10 for the Coastguard sequence and, finally, in Table 5.11 and Table 5.12 for the Stefan sequence. Recall that *Dn* is an error metric while *PSNR* is a quality metric, meaning that *Dn* values should be as low as possible and *PSNR* values should be as high as possible.

Table 5.7 - *Dn* values for the Weather sequence video objects

Dn (%)	Reference shape refreshment			Adaptive shape refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	4.75	19.38	14.03	4.75	19.38	14.03
EC2	3.40	14.85	10.66	3.46	10.70	8.05
EC3	0.55	1.80	1.34	0.65	1.41	1.13
EC4	0.38	1.20	0.90	0.39	0.86	0.69
EC5	0.43	1.54	1.13	0.52	1.12	0.90
EC6	0.00	0.01	0.01	0.00	0.01	0.01
EC7	0.41	1.63	1.18	0.41	1.63	1.18
EC8	1.31	3.87	2.93	1.31	3.87	2.93

5. Adaptive Object-based Video Coding Refreshment Scheme

Table 5.8 - PSNR values for the Weather sequence video objects

PSNR (dB)	Reference shape refreshment			Adaptive shape refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	21.55	20.07	20.61	21.55	20.07	20.61
EC2	21.92	20.15	20.80	21.83	20.30	20.86
EC3	23.12	24.68	24.11	23.13	24.85	24.22
EC4	23.44	25.84	24.96	23.44	25.84	24.96
EC5	23.31	25.44	24.66	23.29	25.50	24.69
EC6	23.92	27.71	26.32	23.95	27.67	26.31
EC7	23.26	25.27	24.53	23.26	25.27	24.53
EC8	22.48	22.92	22.76	22.48	22.92	22.76

Table 5.9 - Dn values for the Coastguard sequence video objects

Dn (%)	Reference shape refreshment					Adaptive shape refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	11.71	26.41	18.17	4.91	16.67	11.84	24.97	14.95	5.56	15.45
EC2	7.69	25.46	14.26	4.53	14.37	8.72	18.30	10.93	5.31	11.58
EC3	1.53	5.06	2.37	0.61	2.66	2.21	3.55	2.13	0.65	2.27
EC4	0.88	3.18	1.64	0.31	1.69	1.27	2.30	1.38	0.30	1.42
EC5	1.32	4.25	1.37	0.49	2.06	1.90	3.09	1.31	0.54	1.81
EC6	0.03	0.09	0.00	0.02	0.04	0.04	0.04	0.00	0.01	0.02
EC7	1.10	2.45	2.55	0.33	1.78	1.30	2.15	1.77	0.32	1.49
EC8	3.13	7.82	8.13	1.47	5.68	3.56	6.97	5.88	1.97	4.95

Table 5.10 - PSNR values for the Coastguard sequence video objects

PSNR (dB)	Reference shape refreshment					Adaptive shape refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	23.86	17.15	18.90	21.31	19.84	23.80	17.29	19.10	21.23	19.91
EC2	24.34	17.17	18.96	21.27	19.95	24.20	17.33	19.77	21.28	20.21
EC3	29.99	21.84	27.25	25.91	25.86	29.43	22.05	27.50	25.88	25.88
EC4	30.94	23.88	29.30	27.56	27.61	30.91	24.09	29.58	27.17	27.67
EC5	30.24	22.55	29.18	26.54	26.82	29.98	22.97	29.26	26.57	26.93
EC6	32.32	27.32	33.58	29.55	30.59	32.33	27.48	33.58	29.58	30.65
EC7	30.30	23.69	27.68	26.96	26.83	29.98	23.76	28.06	27.01	26.91
EC8	27.03	19.98	23.43	24.15	23.23	27.02	20.08	23.77	24.14	23.36

Table 5.11 - Dn values for the Stefan sequence video objects

Dn (%)	Reference shape refreshment			Adaptive shape refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	7.94	122.38	72.83	8.19	129.35	76.89
EC2	5.99	115.89	68.30	4.66	96.75	56.88
EC3	0.89	11.44	6.87	1.14	6.58	4.22
EC4	0.51	6.64	3.99	0.64	4.90	3.06
EC5	0.50	3.59	2.25	0.68	2.52	1.72
EC6	0.04	0.36	0.22	0.02	0.18	0.11
EC7	1.69	15.27	9.39	2.11	9.11	6.08
EC8	4.87	74.21	44.19	5.25	54.51	33.18

Table 5.12 - $PSNR$ values for the Stefan sequence video objects

$PSNR$ (dB)	Reference shape refreshment			Adaptive shape refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	18.73	13.29	15.65	18.78	12.85	15.42
EC2	19.03	14.25	16.32	19.57	16.00	17.55
EC3	23.79	24.20	24.02	23.66	25.34	24.61
EC4	24.71	26.93	25.97	24.51	27.15	26.01
EC5	24.52	27.28	26.08	24.37	27.54	26.17
EC6	25.93	31.23	28.94	26.01	31.37	29.05
EC7	22.00	22.35	22.20	21.64	23.77	22.85
EC8	19.45	17.90	18.57	19.38	19.03	19.18

In the first part of this section, it was concluded that the adaptive intra shape refreshment technique would be increasingly beneficial for the test sequences in the following order: Weather, Coastguard and Stefan. In order to verify this preview, the obtained decoder results are analyzed by following the same order, as follows:

- **Weather sequence** – For the Weather sequence, Table 5.7 shows that for some error conditions (i.e. EC1, EC6, EC7 and EC8) the performance of the adaptive shape refreshment scheme in terms of global shape quality is the same as that of the reference scheme. However, for the remaining error conditions, the adaptive shape refreshment scheme performs slightly better than the reference scheme. The clear trend is for the shape quality of the Weather Girl object to improve whereas the quality of the Weather Map object decreases. However, it should be noted that the quality increase for the Weather Girl is larger than the quality decrease for the Weather Map, which is translated in the fact that the global scene shape quality improves. For instance, for EC2, the Dn values go from 14.85% to 10.70% for the Weather Girl and from 3.40% to 3.46% for the Weather Map. In terms of scene shape quality, this corresponds to a decrease in the *global* Dn value from 10.66% to 8.05%. In terms of texture quality, the behavior is the same as for the shape quality. The $PSNR$ values shown in Table 5.8 indicate that for some

5. Adaptive Object-based Video Coding Refreshment Scheme

error conditions the performance of the adaptive shape refreshment scheme in terms of global texture quality is the same as that of the reference scheme. However, for the remaining error conditions, the global texture quality is slightly improved when the adaptive scheme is used. Similarly to what was said above, the trend is for the texture quality of the Weather Girl object to improve more than the quality of the Weather Map object decreases.

- **Coastguard sequence** – For the Coastguard sequence, the differences between the performance of the reference and the adaptive shape refreshment schemes are more evident. In fact, Table 5.9 shows that shape quality improvements, due to the adaptive shape refreshment scheme, for the Large Boat and Small Boat objects are quite large, whereas the shape quality degradation for the Water and Shore objects is much smaller. As an example, consider EC3. In this case, the Dn values go from 5.06% to 3.55% for the Large Boat and from 2.37% to 2.13% for the Small Boat, whereas the Dn values go from 1.53% to 2.21% for the Water object and from 0.61% to 0.65% for the Shore object. For the whole scene, this means that the *global Dn* goes from 2.66% to 2.27 %, which is a considerable decrease (approximately 15%) for the same resources. This same kind of behavior occurs for all error conditions, except for EC4 and EC7, where the Dn value associated to the Shore object does not increase (it decreases very slightly). However, this also helps in guaranteeing that the scene shape quality improves. In terms of texture quality, the same behavior is observed: the $PSNR$ values associated with the Large Boat and the Small Boat (the objects where human attention will be concentrated) increase while those associated with the Water and the Shore decrease, when the adaptive shape refreshment scheme is used. This results in the global texture quality being increased for all the error conditions.
- **Stefan sequence** – Finally, the Stefan sequence is where the differences between the reference and the adaptive shape refreshment schemes are more obvious. This was expected because this sequence is also the one where the two objects differ the most in terms of shape refreshment need. For EC1 and EC2, which represent the most severe error conditions, the shape and texture quality values are so low that the results are meaningless (Dn around 100% for Tennis Player and $PSNR$ around 12 to 16 dB also for Tennis Player) and therefore are left out of this analysis. For the other error conditions, the improvements are great in terms of shape quality for the Tennis Player object, but the Dn increase for the Background object is much smaller. This is reflected in the *global Dn* values which decrease considerably for all error conditions. For instance, for EC3, Dn values go from 11.44% to 6.58% for the Tennis Player and from 0.89% to 1.14% for the Background. In terms of the scene shape quality, this corresponds to a decrease in the *global Dn* value from 6.87% to 4.22% (approximately 39%). In terms of texture quality, the same behavior is observed for the $PSNR$ values, with the texture quality of the Tennis Player object increasing more than the texture quality of the Background decreases. Globally, the scene $PSNR$ increases for all the error conditions.

The first remark that can be made regarding the results above is that they agree with what was said about the expected improvements for each one of the tested sequences: the adaptive shape refreshment is increasingly interesting as the test sequences are more unbalanced in terms of the shape refreshment needs for the various video objects since the shape quality of video objects with higher shape refreshment needs is more improved than the shape quality of the other objects is worsened, which leads to an improved scene shape quality. This conclusion can be generalized by saying that the adaptive shape intra refreshment technique:

- Has a similar performance to the reference shape refreshment technique, when the video sequences have video objects with similar shape refreshment needs.
- Outperforms (sometimes very significantly) the reference shape refreshment technique, when the video sequences include video objects with very different shape refreshment needs.

Therefore, one can conclude that the use of the adaptive shape intra refreshment technique is worth considering for any encoder, since it is never outperformed by the reference shape refreshment technique.

Regarding the texture quality, the following two remarks can be made, based on the results shown above:

- When the video sequences include objects with similar shape refreshment needs, the quality of the texture is basically the same for the adaptive and the reference shape refreshment techniques. This is easily understandable because the decoded shapes are approximately the same for both shape refreshment schemes and the texture refreshment technique is the same.
- When the video sequences include objects with very different shape refreshment needs, the quality of the texture increases globally when the adaptive shape refreshment technique is used because the texture quality of the objects being more refreshed improves more than the texture quality of the other objects decreases. This is due to the dependency of the texture quality on the correctness of the shape data, described earlier. In addition, some small variations (increases as well as decreases) can be attributed to the fact that the number of bits spent on shape data at each time instant when the adaptive shape refreshment technique is used can change, thus leaving a different number of bits for the texture, and to the fact that the set of pixels used for the *PSNR* computations is not the same, because the decoded shapes are different.

Therefore, one can further emphasize that the use of the adaptive shape intra refreshment technique is worth considering for any encoder, since it is also never outperformed by the reference shape refreshment technique in terms of texture quality. In addition, if used with an adequate adaptive texture refreshment technique, it should further improve the texture quality of the scene.

5.3.4.2 Adaptive Texture Refreshment Scheme Results

In order to study the performance of the texture refreshment part of the algorithm, the reference shape refreshment scheme is adopted. In all of the following, a fixed shape refreshment rate of 5 is used, which is the value that was already used in the previous section for the shape refreshment study. It is important that this value be realistic, because the texture quality is highly dependent on the shape quality because of the way the *PSNR* is computed. Therefore, if the shape refreshment rate were inadequate, this could render the texture results meaningless. Since fixed shape refreshment rates, ranging from 1 to 6, were tested by Watanabe and Kikuchi in [Watanabe97] with the same error conditions considered here and the decoded video quality was considered acceptable, a shape refreshment rate of 5 was adopted for all conditions.

In terms of the texture refreshment rate, it was decided to adopt a value commonly used. In frame-based coding schemes, it is quite common to have full texture refreshments every 10 frames, which was initially due to random access requirements, but turns out to be also a convenient value for error resilience purposes. Here, however, the refreshment is distributed

5. Adaptive Object-based Video Coding Refreshment Scheme

over time and among the video objects. Since the results are given for the QCIF format, which corresponds to 99 macroblocks per frame, this means that approximately 10 macroblocks should be refreshed, at each time instant, for both techniques (for scenes without overlapping objects). Since texture coding in MPEG-4 is block-based, there will inevitably be some overlapping at the border of the video objects, which means that there are actually more than 99 macroblocks per frame in the QCIF format. The actual number depends on the shape of the video objects. Therefore, in order to take these extra macroblocks into account, an additional 20% was added, which corresponds to refreshing 12 macroblocks instead of 10, at each time instant. In terms of the distribution of the number of macroblocks to be refreshed among the VOPs, this is where the two techniques (adaptive and reference) differ, as was explained in Section 5.3.1.2.

This way, both D_n and $PSNR$ figures are given in the following tables, in order to evaluate the performance of the adaptive texture intra refreshment technique. For the various error conditions defined in Table 5.4, *individual* D_n and $PSNR$ figures are given for the various video objects, as well as *global* D_n and $PSNR$ figures for the whole scene. The D_n and $PSNR$ values are given in Table 5.13 and Table 5.14 for the Weather sequence, in Table 5.15 and Table 5.16 for the Coastguard sequence and, finally, in Table 5.17 and Table 5.18 for the Stefan sequence.

Table 5.13 - D_n values for the Weather sequence video objects

D_n (%)	Reference texture refreshment			Adaptive texture refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	4.75	19.38	14.03	4.11	14.00	10.38
EC2	3.40	14.85	10.66	3.81	10.88	8.29
EC3	0.55	1.80	1.34	0.62	1.96	1.47
EC4	0.38	1.20	0.90	0.38	1.24	0.93
EC5	0.43	1.54	1.13	0.56	1.57	1.20
EC6	0.00	0.01	0.01	0.00	0.01	0.01
EC7	0.41	1.63	1.18	0.49	1.55	1.16
EC8	1.31	3.87	2.93	1.31	3.84	2.91

Table 5.14 - $PSNR$ values for the Weather sequence video objects

$PSNR$ (dB)	Reference texture refreshment			Adaptive texture refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	21.55	20.07	20.61	22.28	20.75	21.31
EC2	21.92	20.15	20.80	22.43	21.26	21.69
EC3	23.12	24.68	24.11	23.01	26.35	25.13
EC4	23.44	25.84	24.96	23.35	27.18	25.78
EC5	23.31	25.44	24.66	23.09	26.87	25.49
EC6	23.92	27.71	26.32	24.11	28.65	26.99
EC7	23.26	25.27	24.53	23.30	26.46	25.30
EC8	22.48	22.92	22.76	22.74	24.03	23.56

Table 5.15 - Dn values for the Coastguard sequence video objects

Dn (%)	Reference texture refreshment					Adaptive texture refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	11.71	26.41	18.17	4.91	16.67	9.91	27.41	16.66	5.85	16.38
EC2	7.69	25.46	14.26	4.53	14.37	8.35	25.18	14.04	5.31	14.52
EC3	1.53	5.06	2.37	0.61	2.66	1.54	4.61	2.85	0.80	2.71
EC4	0.88	3.18	1.64	0.31	1.69	0.74	2.41	1.66	0.35	1.44
EC5	1.32	4.25	1.37	0.49	2.06	1.44	3.02	1.18	0.61	1.67
EC6	0.03	0.09	0.00	0.02	0.04	0.06	0.04	0.00	0.01	0.03
EC7	1.10	2.45	2.55	0.33	1.78	1.03	3.30	2.56	0.42	2.04
EC8	3.13	7.82	8.13	1.47	5.68	2.77	10.18	7.28	1.33	6.05

Table 5.16 - PSNR values for the Coastguard sequence video objects

PSNR (dB)	Reference texture refreshment					Adaptive texture refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	23.86	17.15	18.90	21.31	19.84	23.28	16.53	19.52	21.03	19.66
EC2	24.34	17.17	18.96	21.27	19.95	23.30	17.09	20.43	20.84	20.06
EC3	29.99	21.84	27.25	25.91	25.86	28.45	24.15	28.70	24.88	26.47
EC4	30.94	23.88	29.30	27.56	27.61	30.00	25.79	30.65	26.60	28.20
EC5	30.24	22.55	29.18	26.54	26.82	29.23	25.13	30.42	25.59	27.57
EC6	32.32	27.32	33.58	29.55	30.59	31.56	28.31	33.97	28.78	30.70
EC7	30.30	23.69	27.68	26.96	26.83	29.86	25.09	29.74	26.24	27.62
EC8	27.03	19.98	23.43	24.15	23.23	26.76	21.01	24.87	23.79	23.84

Table 5.17 - Dn values for the Stefan sequence video objects

Dn (%)	Reference texture refreshment			Adaptive texture refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	7.94	122.38	72.83	7.84	123.55	73.45
EC2	5.99	115.89	68.30	5.70	90.40	53.72
EC3	0.89	11.44	6.87	0.85	9.75	5.90
EC4	0.51	6.64	3.99	0.49	5.72	3.46
EC5	0.50	3.59	2.25	0.58	2.87	1.88
EC6	0.04	0.36	0.22	0.04	0.39	0.24
EC7	1.69	15.27	9.39	1.74	15.76	9.69
EC8	4.87	74.21	44.19	3.69	60.21	35.74

Table 5.18 - PSNR values for the Stefan sequence video objects

PSNR (dB)	Reference texture refreshment			Adaptive texture refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	18.73	13.29	15.65	18.71	12.79	15.35
EC2	19.03	14.25	16.32	18.99	17.72	18.27
EC3	23.79	24.20	24.02	23.22	27.99	25.92
EC4	24.71	26.93	25.97	24.09	30.24	27.58
EC5	24.52	27.28	26.08	23.86	30.02	27.35
EC6	25.93	31.23	28.94	25.50	33.03	29.77
EC7	22.00	22.35	22.20	21.50	25.34	23.68
EC8	19.45	17.90	18.57	19.73	20.02	19.89

The Dn and $PSNR$ figures obtained for the studied test sequences, when the reference and the adaptive texture refreshment techniques are used, show that:

- Weather sequence** – For the Weather sequence, in some cases the scene shape quality is improved when the adaptive texture refreshment technique is used (e.g. in EC1 the *global* Dn value goes from 14.03% to 10.38%), while in some other cases it is slightly worsened (e.g. in EC3 the *global* Dn value goes from 1.34% to 1.47%) and still in other cases it remains unchanged (e.g. in EC6 the *global* Dn value remains equal to 0.01%). Individually, the same type of behavior also happens. For instance, by considering the Weather Girl object, in some cases the video object shape quality improves (e.g. in EC1, Dn goes from 19.38% to 14.00%), while in some other cases it slightly decays (e.g. in EC3, Dn goes from 1.80% to 1.96%) and still in other cases it remains unchanged (e.g. in EC6, Dn remains equal to 0.01%). However, the shape quality should have remained unchanged, for all error conditions, for both the reference and the adaptive texture refreshment techniques because in both cases the same lossless shape coding was used and the quality of the shape does not depend on the texture data. The existing differences shown in Table 5.13 are due to the fact that, even if the shape refreshment technique is the same in both cases, the bitstreams used for the evaluation are different due to the different texture coding and error corruption. In terms of texture, Table 5.14 shows a clear improvement in the *global* $PSNR$ values for all error conditions when the adaptive refreshment technique is used. The smallest improvement in texture quality happens for EC6, where the *global* $PSNR$ goes up by 0.67 dB, whereas the largest improvement corresponds to EC3, where the *global* $PSNR$ increases 1.02 dB. In terms of the individual texture quality figures, an interesting aspect should be noticed: the $PSNR$ values of both video objects tend to go up (except for EC3, EC4 and EC5, where only the Weather Girl has its texture quality improved), even though most of the texture macroblocks to be refreshed are assigned to the Weather Girl object. This happens because the Weather Map object is synthetic and still, which makes it very easy to encode and conceal when errors occur, since all the macroblocks, with the exception of the border macroblocks, are identical to the co-located macroblocks in the previous time instant. This way, by shifting most intra refreshment macroblocks from the Weather Map to the Weather Girl object, a lot of bits will be saved (more bits will be saved in the Weather Map object than the extra bits needed for the Weather Girl object), which can be used to improve the quality of the

texture of both objects. The fact that such a small amount of intra refreshment is used in the Weather Map object does not make it more vulnerable because it is still and, therefore, any errors that might occur will be very easy to conceal by simply copying the corresponding texture from a past time instant. As an example, consider EC2. In this case, the texture quality of the Weather Map object increases from 21.92 dB to 22.43 dB and the texture quality of the Weather Girl object also increases from 20.15 dB to 21.26 dB.

- **Coastguard sequence** – For the Coastguard sequence, the same shape quality behavior described for the Weather sequence is shown in Table 5.15. In terms of texture, Table 5.16 shows a clear improvement in the *global PSNR* values for all error conditions when the adaptive refreshment technique is used. The smallest improvement in texture quality happens for EC2 and EC6, where the *global PSNR* goes up by 0.11 dB, whereas the largest improvement corresponds to EC7, where the *global PSNR* increases 0.79 dB. In terms of individual texture quality, however, the *PSNR* figures do not go up for all the video objects as in the Weather sequence. Here, the texture of the Large Boat and the Small Boat objects is improved while the texture quality of the Water and Shore objects decays. This happens because most of the texture macroblocks to be refreshed are assigned to the Large Boat and Small Boat objects at the expense of the other two objects. As an example, consider EC3: in this case, the texture quality of the Water object decreases from 29.99 dB to 28.45 dB and the texture quality of the Shore Boat object decreases from 25.91 dB to 24.88 dB. However, the texture quality of the Large Boat and the Small Boat increase greatly from 21.84 dB to 24.15 dB and from 27.25 dB to 28.70 dB, respectively. Globally, this corresponds to a *global PSNR* increase of 0.61 dB.
- **Stefan sequence** – Similarly to what happened in Section 5.3.4.1, in the most severe error conditions (EC1 and EC2), the shape and texture quality values of the Stefan sequence are so low that the results are meaningless (*Dn* around 100% for Tennis Player and *PSNR* around 12 to 15 dB also for Tennis Player). Therefore, these results are left out of this analysis. In terms of shape quality, and as can be seen in Table 5.17, the same comment that was made about the two previous sequences also applies here: the shape quality is approximately the same when the reference and the adaptive texture refreshment techniques are used. In terms of texture, Table 5.18 shows the same situation that was described for the Weather and the Coastguard sequences but to an even larger extent: the scene texture quality increases for all error conditions. The smallest increase happens for EC6, where the *global PSNR* goes up 0.83 dB, whereas the greatest increase corresponds to EC3, where the *global PSNR* increases 1.90 dB. The individual texture quality figures, however, tend to go up for the Tennis Player object and down for the Background object when the adaptive texture refreshment scheme is used. This happens because most of the macroblocks to be refreshed are assigned to the Tennis Player object at the expense of the texture quality of the Background object. Thus the improvement in the Tennis Player object is more than enough to compensate for the decay in the Background object. As an example, consider EC4: in this case, the texture quality of the Background object decreases from 24.71 dB to 24.09 dB, whereas the texture quality of the Tennis Player object improves greatly from 26.93 dB to 30.24 dB. This corresponds to a scene texture quality improvement of 1.61 dB.

The first remark that can be made about the results above is that the shape quality remains rather unchanged regardless of the texture refreshment technique that is used. This result was expected because the shape refreshment solution is the same in both cases and lossless shape coding is always used. In addition, the quality of the texture does not affect the quality of the shape, whereas the opposite is not true, for the reasons explained in Section 5.3.4.1.

5. Adaptive Object-based Video Coding Refreshment Scheme

In terms of texture, the obtained results show that the use of the adaptive texture refreshment technique always results in an improved scene texture quality, even if it is more effective in some sequences than others. In particular, for the tested sequences, the adaptive texture refreshment technique proved to be increasingly effective for the Coastguard, Weather and Stefan sequences. Therefore, one can conclude that the use of the adaptive texture intra refreshment technique is worth considering for any encoder, since it always outperforms the reference texture refreshment technique.

5.3.4.3 Adaptive Combined Refreshment Scheme Results

In order to study the performance of the combined shape and texture refreshment techniques, the same parameter values that were already used in Section 5.3.4.1 and Section 5.3.4.2 will be used. This means that for the fixed refreshment case, the used shape refreshment period is 5 and the number of macroblocks to be refreshed in each time instant is 12. As for the adaptive refreshment case, an $ASRP_{target}$ value of 5 and an $NTMR$ value of 12 are used.

Both Dn and $PSNR$ figures are presented in the following tables, to evaluate the performance of the adaptive combined intra refreshment technique. For the various error conditions defined in Table 5.4, *individual* Dn and $PSNR$ figures are given for the various video objects, as well as *global* Dn and $PSNR$ figures for the whole scene. The Dn and $PSNR$ values are given in Table 5.19 and Table 5.20 for the Weather sequence, in Table 5.21 and Table 5.22 for the Coastguard sequence and, finally, in Table 5.23 and Table 5.24 for the Stefan sequence.

Table 5.19 - Dn values for the Weather sequence video objects

Dn (%)	Reference shape and texture refreshment			Adaptive shape and texture refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	4.75	19.38	14.03	4.11	14.00	10.38
EC2	3.40	14.85	10.66	3.61	10.17	7.77
EC3	0.55	1.80	1.34	0.95	1.26	1.15
EC4	0.38	1.20	0.90	0.47	0.77	0.66
EC5	0.43	1.54	1.13	0.85	0.95	0.91
EC6	0.00	0.01	0.01	0.00	0.00	0.00
EC7	0.41	1.63	1.18	0.49	1.55	1.16
EC8	1.31	3.87	2.93	1.31	3.84	2.91

Table 5.20 - PSNR values for the Weather sequence video objects

PSNR (dB)	Reference shape and texture refreshment			Adaptive shape and texture refreshment		
	Weather Map	Weather Girl	Global	Weather Map	Weather Girl	Global
EC1	21.55	20.07	20.61	22.28	20.75	21.31
EC2	21.92	20.15	20.80	22.53	21.30	21.75
EC3	23.12	24.68	24.11	23.04	26.61	25.30
EC4	23.44	25.84	24.96	23.33	27.43	25.93
EC5	23.31	25.44	24.66	23.11	27.11	25.65
EC6	23.92	27.71	26.32	24.24	28.49	26.93
EC7	23.26	25.27	24.53	23.30	26.46	25.30
EC8	22.48	22.92	22.76	22.74	24.03	23.56

Table 5.21 - Dn values for the Coastguard sequence video objects

Dn (%)	Reference shape and texture refreshment					Adaptive shape and texture refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	11.71	26.41	18.17	4.91	16.67	10.27	25.38	14.33	7.19	15.43
EC2	7.69	25.46	14.26	4.53	14.37	8.39	17.98	9.89	5.37	11.13
EC3	1.53	5.06	2.37	0.61	2.66	1.70	3.30	2.26	0.80	2.17
EC4	0.88	3.18	1.64	0.31	1.69	0.91	2.07	1.00	0.56	1.22
EC5	1.32	4.25	1.37	0.49	2.06	1.32	1.89	0.96	0.72	1.26
EC6	0.03	0.09	0.00	0.02	0.04	0.03	0.06	0.00	0.04	0.03
EC7	1.10	2.45	2.55	0.33	1.78	1.28	2.37	2.20	0.48	1.72
EC8	3.13	7.82	8.13	1.47	5.68	3.23	7.73	6.94	1.81	5.39

Table 5.22 - PSNR values for the Coastguard sequence video objects

PSNR (dB)	Reference shape and texture refreshment					Adaptive shape and texture refreshment				
	Water	Large Boat	Small Boat	Shore	Global	Water	Large Boat	Small Boat	Shore	Global
EC1	23.86	17.15	18.90	21.31	19.84	24.00	17.16	19.57	20.94	19.99
EC2	24.34	17.17	18.96	21.27	19.95	24.22	17.32	21.07	20.80	20.49
EC3	29.99	21.84	27.25	25.91	25.86	29.01	24.55	29.51	24.89	26.94
EC4	30.94	23.88	29.30	27.56	27.61	30.13	26.03	31.17	26.52	28.44
EC5	30.24	22.55	29.18	26.54	26.82	29.35	25.23	30.88	25.38	27.72
EC6	32.32	27.32	33.58	29.55	30.59	31.56	28.21	34.25	28.76	30.74
EC7	30.30	23.69	27.68	26.96	26.83	29.43	25.45	29.83	26.53	27.73
EC8	27.03	19.98	23.43	24.15	23.23	27.17	21.34	24.90	23.89	24.05

5. Adaptive Object-based Video Coding Refreshment Scheme

Table 5.23 - D_n values for the Stefan sequence video objects

D_n (%)	Reference shape and texture refreshment			Adaptive shape and texture refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	7.94	122.38	72.83	8.55	126.54	75.45
EC2	5.99	115.89	68.30	5.20	100.84	59.43
EC3	0.89	11.44	6.87	1.20	7.95	5.03
EC4	0.51	6.64	3.99	0.68	4.61	2.91
EC5	0.50	3.59	2.25	0.68	2.50	1.71
EC6	0.04	0.36	0.22	0.03	0.21	0.13
EC7	1.69	15.27	9.39	2.12	8.82	5.92
EC8	4.87	74.21	44.19	4.94	35.45	22.24

Table 5.24 - PSNR values for the Stefan sequence video objects

PSNR (dB)	Reference shape and texture refreshment			Adaptive shape and texture refreshment		
	Background	Tennis Player	Global	Background	Tennis Player	Global
EC1	18.73	13.29	15.65	18.73	12.92	15.44
EC2	19.03	14.25	16.32	19.25	16.17	17.50
EC3	23.79	24.20	24.02	23.04	28.34	26.05
EC4	24.71	26.93	25.97	23.93	30.37	27.58
EC5	24.52	27.28	26.08	23.62	30.14	27.32
EC6	25.93	31.23	28.94	25.48	33.28	29.90
EC7	22.00	22.35	22.20	21.29	26.48	24.23
EC8	19.45	17.90	18.57	19.30	20.63	20.05

The D_n and PSNR figures obtained for the studied test sequences when the reference and the adaptive combined shape and texture refreshment techniques are used show that:

- **Weather sequence** – For the Weather sequence, the behavior in terms of shape quality is very similar to the behavior obtained in Section 5.3.4.1, as can be seen by comparing Table 5.7 and Table 5.19. The behavior in terms of texture quality is very similar to the behavior obtained in Section 5.3.4.2, as can be seen by comparing Table 5.14 and Table 5.20. In fact, it can be seen that the *global* D_n values decrease slightly when the adaptive refreshment techniques are used, which corresponds to a slight increase in shape quality. Additionally, there is a significant improvement (between 0.61 dB and 1.19 dB) in the *global* PSNR values when the adaptive refreshment techniques are used.
- **Coastguard sequence** – For the Coastguard sequence, the behavior in terms of shape quality is very similar to the behavior seen in Section 5.3.4.1, as can be confirmed by comparing Table 5.9 with Table 5.21: there is a considerable increase (larger than for the Weather sequence) in the shape quality when the adaptive refreshment techniques are used. Additionally, by comparing Table 5.16 with Table 5.22, it can be seen that the behavior obtained here in terms of texture quality is very similar to the behavior seen in Section 5.3.4.2, which corresponds to a noticeable, but smaller than for the Weather

sequence, improvement (between 0.15 dB and 1.08 dB) of the scene texture quality when the adaptive refreshment techniques are used.

- **Stefan sequence** – Finally, for the Stefan sequence, similarly to what happens for the previous two sequences, the behavior in terms of shape quality is very similar to the behavior shown in Section 5.3.4.1, as can be seen by comparing Table 5.11 with Table 5.23; the behavior in terms of texture quality is very similar to the behavior obtained in Section 5.3.4.2, as can be seen by comparing Table 5.18 with Table 5.24. In fact, it can be seen that the *global Dn* values decrease considerably (even more than for the Coastguard sequence) when the adaptive refreshment techniques are used, which corresponds to a considerable increase in the shape quality. In terms of texture, there is a great improvement (between 0.96 dB and 2.03 dB) in the *global PSNR* values when the adaptive refreshment techniques are used.

The first remark that can be made about these results is that by combining the shape and the texture adaptive refreshment techniques, it is possible to obtain the advantages of both techniques. In terms of shape, the quality has been increasingly improved for the test sequences Weather, Coastguard and, finally, Stefan, as in Section 5.3.4.1. In terms of texture, the quality has improved for all the test sequences, as in Section 5.3.4.2. Therefore, one can conclude that the use of both shape and texture adaptive intra refreshment techniques should be considered in any encoder, since there are evident quality benefits at no cost in terms of bit rate resources.

5.4 Final Remarks

In this chapter, an adaptive object-based intra refreshment scheme has been proposed, with the target to efficiently distribute the available shape and texture refreshment resources among the various video objects, depending on their intra refreshment needs. This refreshment scheme includes shape and texture refreshment solutions, which can be applied independently or simultaneously. This adaptive algorithm is based on the shape and texture refreshment need metrics proposed in Chapter 4, which measure the necessity of intra coding shape and texture data to decrease the negative impact of channel errors, based on some relevant criteria.

Additionally, since no other object-based intra refreshment techniques are available in the literature, a simpler object-based intra refreshment scheme was also proposed, to be used as a reference. The reference intra refreshment scheme also includes shape and texture refreshment techniques, which as the adaptive intra refreshment techniques can be used independently or simultaneously.

In order to compare the two proposed intra refreshment schemes, performance results were obtained for several relevant situations. These results have shown that the performance of the adaptive shape refreshment technique was always equal or better than the reference shape refreshment one and that the performance of the adaptive texture refreshment technique was also always better than the reference texture refreshment one. This conclusion applies both for the shape and texture techniques used independently and combined. Based on these results, it was concluded that the use of the proposed shape and texture adaptive intra refreshment techniques is worth considering in any object-based video encoder.

This concludes the contribution from this Thesis on error resilience techniques to be used at the encoder side of the communication chain. However, much more can be done at the decoder side as will be shown in the following chapter.

Chapter 6

Shape Error Concealment for Object-based Video Coding

6.1 Introduction

In Chapter 5, an intra refreshment scheme was developed in order to decide in an intelligent way when the shape of a given video object should be refreshed and which texture macroblocks should be refreshed at each time instant. With this scheme, considerable improvements were obtained in terms of the decoded video quality. However, further improvements should be possible by using more sophisticated error concealment techniques at the decoder, in terms of both shape and texture. As far as texture error concealment is concerned, many techniques have already been developed for frame-based video coding systems, which can certainly be extended to work with object-based video coding systems, namely those described in Chapter 3. As for shape error concealment, however, this is not the case and there is a lot of room for further developments and improvements.

In this chapter, the specific issue of shape data error concealment in the context of object-based video coding is addressed, including proposals for two original shape concealment techniques. Both techniques assume that the shape of the corrupted object at hand is in the form of a binary alpha plane and that some of the shape data is missing due to channel errors. From this alpha plane, a contour corresponding to the border of the object can be extracted. Due to errors, some of the contour segments will be missing and, therefore, the contour will be broken. The idea behind the first proposed technique, which is a spatial error concealment technique, is to interpolate the missing contours based on the available surrounding contours. After the border has been fully interpolated and recovered, the concealed alpha plane can be

easily reconstructed and used instead of the erroneous one. As for the second proposed technique, which is a temporal error concealment technique, it basically consists of taking the previously correctly decoded (or already concealed) alpha plane and global motion compensate it. Then, the missing shape data of the corrupted alpha plane can be simply copied from the previous motion compensated alpha plane. Additionally, since some of the corrupted areas of the alpha plane may have some local motion associated to them, this is followed by a local motion refinement. Both techniques can be directly applied to any video object where shape can be represented by means of a binary alpha plane, such as in the MPEG-4 Visual standard [MPEG4-V], leading to an improved decoded video quality. To obtain even better results, the two previous techniques can be combined, and a solution for that will also be proposed.

In Section 6.2, the developed spatial shape concealment technique based on contour interpolation will be described, including results showing its performance under relevant conditions. Afterwards, the developed temporal shape concealment technique based on alpha plane motion compensation will be described in Section 6.3, which also includes results showing its performance under relevant conditions. Then, in Section 6.4, the combination of both proposed techniques into an even more effective scheme is proposed. Finally, some closing remarks about the techniques proposed in this chapter will be made in Section 6.5.

6.2 Spatial Shape Error Concealment

The first error concealment technique proposed in this chapter is a spatial object-level shape error concealment technique. This means that it is designed for object-level concealment, as opposed to scene-level concealment (typically, the output of the object-level concealment stage is used as the starting point for the scene-level concealment stage). Moreover, since it is a spatial (or intra) technique, it relies in no way on information from other temporal instants: it only uses the available (spatial, not temporal) information for the video object being concealed at the time instant in question. In addition to this, the proposed technique relies on information from a previous error detection stage, which means that it has no error detection capabilities and only focuses on the error concealment itself. This approach is adopted just to concentrate on the concealment aspect since error detection can be easily performed by checking the bitstream for syntactic and semantic inconsistencies, as was done in Chapter 5.

The proposed shape error concealment technique can be applied to any corrupted still image or video object whose shape can be represented by means of a binary alpha plane or to the shape support of image or video objects with gray scale alpha plane, such as MPEG-4 still images and MPEG-4 video with binary or gray scale shapes [MPEG4-V]. Notice however that although the proposed technique is very adequate for usage in MPEG-4 Visual decoders, it is by no means limited or anyhow tuned to this type of object-based decoders and thus can be generally used.

6.2.1 Proposal for a Spatial Shape Error Concealment Technique

In this chapter, the idea of interpolating the sketch elements in an image as was done in [Atzori98] (see Chapter 3) is extended to binary alpha planes, which can be seen as a binary texture image, since shapels can have only two values: transparent (0) or opaque (1 or 255). The edges that can be extracted from a natural grayscale image, as in [Atzori98], are however very different from those that can be extracted from an alpha plane. In a natural image, many edges may appear, more or less important. While some of these edges have a semantic

meaning, others do not, but even for those edges that do, losses are never too critical. In binary alpha planes, however, the only edges that are available correspond to the exact object contour, which is extremely important in terms of (object and scene) subjective impact. Moreover, the object contour is also typically needed for the decoding of motion and texture information (such as in MPEG-4 video). Thus, the role, importance, and impact of (missing) edges in sketch-based texture coding and (missing) contours in shape coding are very different. Figure 6.1 and Figure 6.2 allow to better understand these differences; while in Figure 6.1 a multitude of more or less important edges can be extracted from the natural image, in Figure 6.2 only the (exact) contour of the object is extracted from the alpha plane. It is important to remind that all the contours extracted from shape alpha planes are closed and, therefore, never terminate abruptly, which can happen to the edges in natural images.

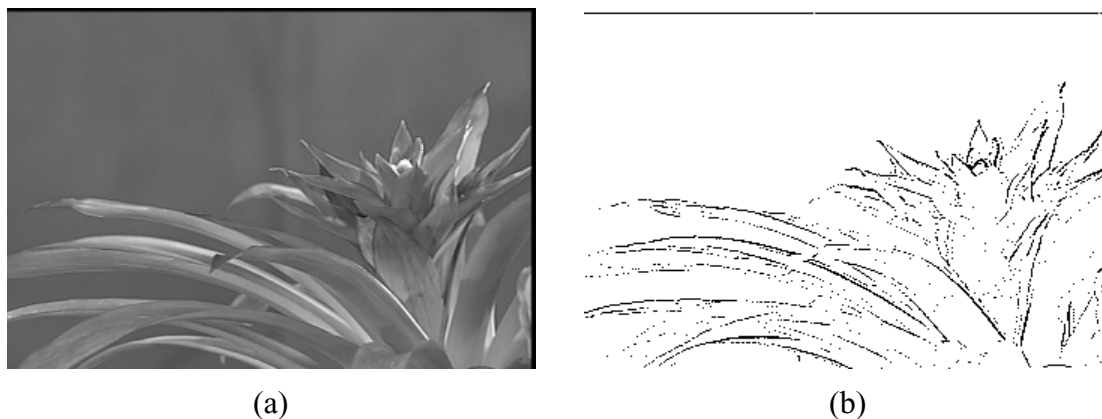


Figure 6.1 – Cyclamen object (only luminance) – (a) Original; (b) Extracted edges

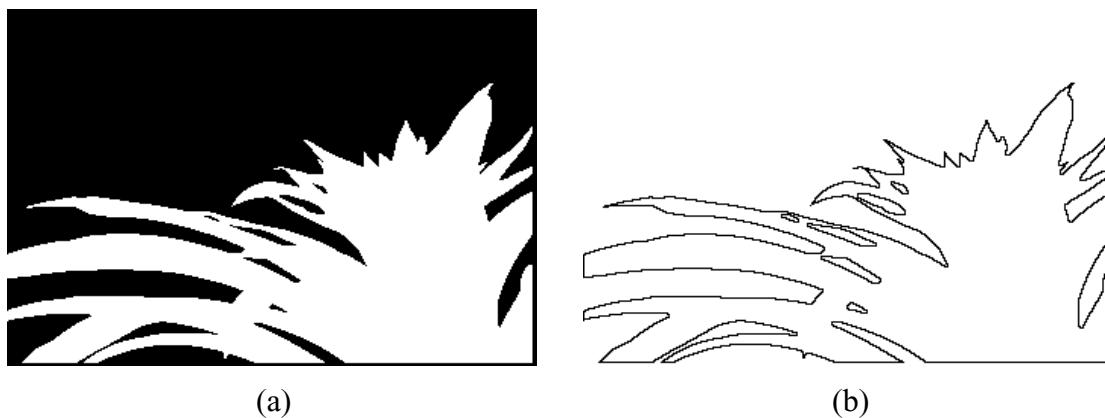


Figure 6.2 – Cyclamen alpha plane – (a) Original; (b) Extracted contour

In alpha planes, the high-frequency content (i.e., contours) is very important, as opposed to the low-frequency content, because in order to recover the complete alpha plane it is sufficient to have the contours and the value of a single shapel (transparent or opaque for one side of the contour). In addition, the contour is also perceptually very important while an edge is never that critical. In fact, a contour with missing parts is topologically invalid and thus not acceptable for further processing, while missing parts in the edge content of an image just result in less quality without stopping any further processing.

The first main contribution of the present chapter is precisely a concealment scheme that can be used for object contours, this means to deal with the loss of alpha plane information which

corresponds to broken contours that have to be interpolated. Although it is fairly easy for a human being to interpolate a missing contour based on the surrounding contours, the same does not apply for a machine. The automatic processing of the broken contours is much more complex. By interpolating the broken contours, it is possible to recover the complete shape with a good accuracy since most contours are fairly well behaved in natural video scenes, showing a slow direction variation. Naturally the quality of the concealment will strongly depend on the validity of this assumption. Therefore, in cases such as geometric shapes where the contours have sharp direction changes (e.g., rectangles or triangles) it is anticipated that the concealment technique may perform less effectively; however it is also rather unexpected that geometric shapes are coded as natural video in the context of object-based visual representation systems. After the broken contours have been recovered, it should be fairly easy to recover the values of the missing shapes from the neighboring ones by using an adequate continuity criterion, and then filling in the shape.

In this chapter, it is assumed that the alpha planes have been encoded with some kind of block-based technique before being delivered. This situation corresponds to the most relevant case in terms of object-based video coding, which is represented by the MPEG-4 Visual standard [MPEG4-V]. In this coding standard, alpha planes are encoded with a block-based technique called Content-based Arithmetic Encoding (CAE); a detailed description can be found in [Brady99][MPEG4-V]. Since in MPEG-4 several consecutive blocks (called macroblocks) are carried in a Video Packet (VP), it is very likely that bitstream errors will manifest themselves in the form of bursts of consecutive erroneous blocks, which have to be detected by an error detection stage before applying the concealment itself. The shape concealment stage is important, not only because motion and texture decoding and concealment highly depend on it, but also because if the concealed shape of the object is very different from the original, then the final subjective impact will be poor. This makes the shape data the most important data transmitted for each object, except maybe for the header data.

The proposed spatial shape concealment technique is based on contour interpolation, using Bézier curves to do the interpolations, instead of the cubic splines suggested in [Atzori98]. Although Bézier curves have one less degree of parametric continuity¹ at join points (C^1 instead of C^2) than natural cubic splines, they are easier to manipulate and much faster to compute [Foley94], which can be a very important feature in (real-time) video applications; this will be detailed in Section 6.2.1.3. The block diagram for the proposed shape concealment technique is presented in Figure 6.3; the input is a corrupted alpha plane with several lost blocks, typically arranged in bursts (see Figure 6.4). The correctly decoded blocks in the same alpha plane will be used to extract useful information for the concealment process, whose output is a fully concealed alpha plane. The four main steps in Figure 6.3 are conceptually similar to those in [Atzori98], but since the concealment is applied to alpha planes instead of texture images and the used error model is not the same, the techniques associated to each processing module are completely different. The four steps in the proposed spatial shape concealment process are:

- **Contour extraction** – This module has the task of extracting the available contours from the corrupted alpha plane;
- **Contour coupling** – This module is responsible for coupling the candidate contours going into each lost area, two by two, according to some criteria;

¹ For a curve $Q(t)$ consisting of several curve segments, if the direction and magnitude of $d^n/dt^n[Q(t)]$ through the n -th derivative are equal at the join point of the various segments, the curve is called C^n continuous or is said to have n -degree parametric continuity.

6. Shape Error Concealment for Object-based Video Coding

- **Contour recovery** – This module is in charge of interpolating a contour between each pair of coupled broken contours through the lost areas;
- **Shape filling** – This module has to perform the final action of filling in the shape information in the alpha plane, based on the recovered contour information.

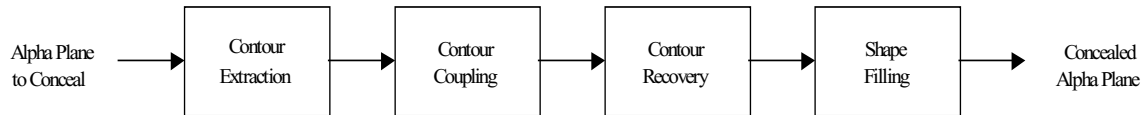


Figure 6.3 – Proposed spatial shape concealment process

In order to understand better the shape concealment process, an illustrative example will be given before detailing each of the steps above in the following sections; each of these steps corresponds to a well defined and rather independent operation. As explained above, the input of the concealment chain depicted in Figure 6.3 is an alpha plane where some blocks have been lost (i.e., the lost area), as illustrated in Figure 6.4 (a). The lost area is shown in gray throughout the example. The first step is to extract the contour of the corrupted alpha plane, which is illustrated in Figure 6.4 (b), where the (2) broken contours can be clearly seen. After that, the (4) broken contour endings have to be coupled, which is a very easy task for a human being, and then interpolated, which is illustrated in Figure 6.4 (c). Finally, after the contour has been fully recovered, the shape can be filled in, as shown in Figure 6.4 (d). The reason why the contour images appear with double the size of the alpha plane images will be explained in Section 6.2.1.1.

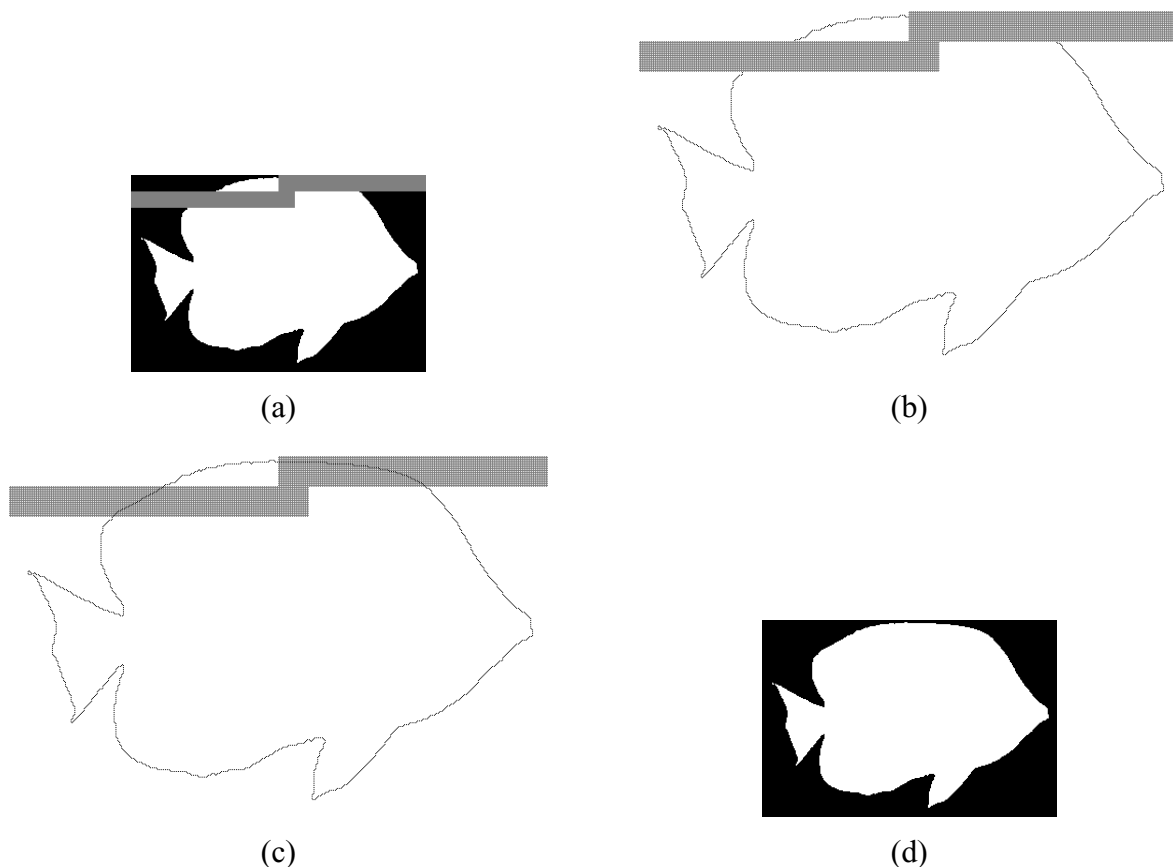


Figure 6.4 – Exemplifying the steps of the spatial shape concealment process: (a) Lost blocks surrounded by the available alpha plane blocks; (b) Lost blocks surrounded by the available contours; (c) Interpolated contours inside the lost blocks; (d) Recovered alpha plane

In the previous example, an alpha plane was shown with only one burst of consecutive lost blocks (i.e., one single connected lost area). However, this does not have to be so and many bursts may appear, as illustrated in Figure 6.5. Since the contour endings around a given lost area cannot be coupled with the contour endings of another lost area, the shape concealment processing of each lost area can and will be done independently.

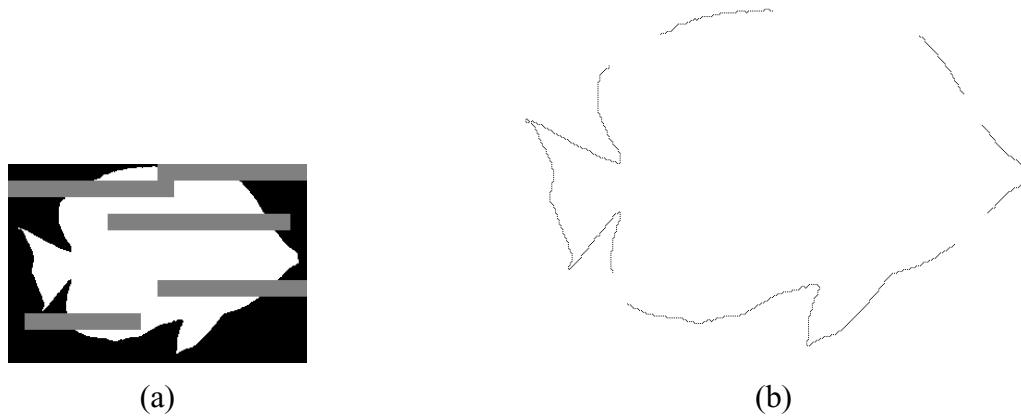


Figure 6.5 – (a) Alpha plane with several bursts of lost blocks and (b) Corresponding broken contours

6.2.1.1 Contour Extraction

The proposed error concealment technique is based on the interpolation of contours not shape, hence the need to go from the binary alpha plane to the contour itself, before starting any concealment processing. This task implies the extraction from the available shape of a contour image with the (correctly) decoded contours as shown in Figure 6.4 (a) and (b). In order to represent a contour, three approaches can be used:

- **Edge representation** – The contour is represented by the set of intervals between two adjacent shapels – *edge sites for 4-connectivity shapels* – where an edge effectively exists, this means where the two neighboring shapels are different: one transparent and the other opaque. Notice that here the term ‘edge’ has a different meaning than the one used before at the beginning of Section 6.2. To represent a contour associated with an alpha plane, all the edge sites are inspected, and those that have different valued neighboring shapels are said to be contour edges. The matrix containing the set of these edges is called the *edge representation of the contour*. As an example, Figure 6.6 depicts a simplified alpha plane where white squares represent opaque shapels (belonging to the object) and black squares represent transparent shapels (outside the object). In Figure 6.7 (a), the associated edge representation of the corresponding contour is shown. The line segments between two shapels are the edge sites and the darker edge sites correspond to the contour edges for the contour in question.
- **Vertex representation** – The contour is represented by the set of points between two diagonally adjacent shapels – *vertex sites for 8-connectivity shapels* – where a vertex effectively exists, this means where the 4 neighboring shapels are not equal. In this case, a vertex site can have four neighboring shapels. Similarly to what happens for the edge representation, in order to represent a contour associated to an alpha plane, all the vertex sites are inspected, and those that do not have 4 equal neighboring shapels are said to be contour vertices. The matrix containing the set of these vertices is called the *vertex*

representation of the contour. Similarly to Figure 6.7 (a), Figure 6.7 (b) depicts the vertex representation of the contour associated with the alpha plane in Figure 6.6. The white circles represent the vertex sites and the black circles represent the contour vertices for the contour in question.

- **Shapel representation** – The contour is represented by the set of shapels that belong to the object but have at least one neighbor shapel that is outside the object – *border shapels*. Again, in order to represent a contour associated with an alpha plane, all shapels are inspected, and those that are border shapels are used to represent the contour. In Figure 6.7 (c), the shapel representation of the contour associated with the alpha plane in Figure 6.6 is shown. The black squares represent border shapels and, therefore, the *shapel representation of the contour*; as for the shape internal white squares, they just represent the non-border shapels.

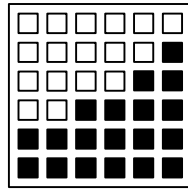


Figure 6.6 - Simplified binary alpha plane (white for shapels belonging to the object)

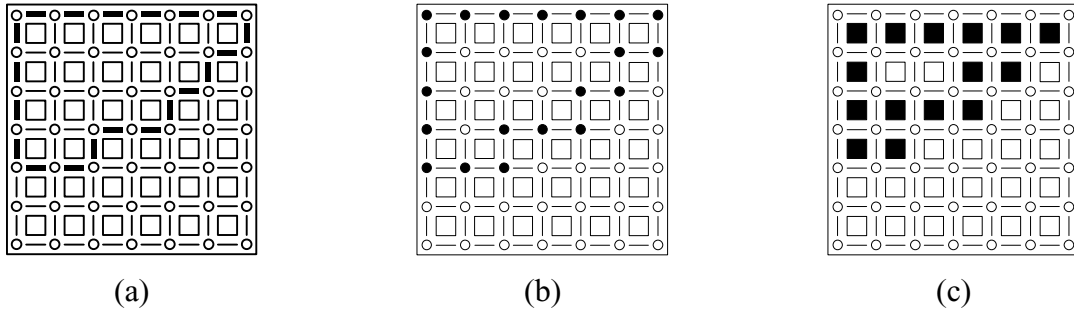


Figure 6.7 – Different representations of the object contour in Figure 6.6 – (a) Edge representation; (b) Vertex representation; (c) Shapel representation

For the proposed spatial shape concealment technique, the edge representation of contours was chosen since this solution avoids a number of topological problems associated with the other two types of representation. A detailed discussion of these different types of representations can be found in [Sequeira97]. The price to pay for this more accurate representation is that the resulting edge representation has to be stored in a larger matrix than the original alpha plane. For instance, assuming that the original alpha plane is an $M \times N$ matrix, then the contour matrix will have to be a $(2M+1) \times (2N+1)$ matrix to hold the edge representation. If the chosen representation were shapel-based, an $M \times N$ matrix would suffice.

Since in this type of contour representation, the contour passes between adjacent shapels, it is necessary to consider the space between these shapels, which will be referred to as edge sites. When 4-connectivity is considered, a pair of adjacent shapels can be either horizontal or vertical, and thus the corresponding edge sites will be either vertical or horizontal. Additionally, it is useful to define the dual lattice C , as the two-dimensional lattice containing all the possible edge sites between shapels. In this lattice, each edge site has only two neighboring (adjacent) shapels. This way, if x is the edge site, then $N_2(x)$ will represent the 2-

neighborhood corresponding to the two adjacent shapels to the edge site in question, regardless of the edge site direction. This means that, if x is an horizontal edge site, then $N_2(x)$ will be the shapels above and below. On the other hand, if x is a vertical edge site, then $N_2(x)$ will be the shapels on the left and on the right.

To extract the contours from the binary alpha plane and represent them with the edge representation, a very simple contour filter has to be implemented. For each edge site, the 2 adjacent shapels are examined: if they are different, a contour exists. Therefore, the output of this filter can be defined as follows:

$$h(x) = \begin{cases} 0, & \text{if the shapels in } N_2(x) \text{ are equal} \\ 1, & \text{if the shapels in } N_2(x) \text{ are different} \end{cases}, x \in C. \quad (6.1)$$

As can be seen in Figure 6.7, some edge sites exist along the border of the contour matrix that only have one neighboring shapel inside this matrix, which shall be called *border edge sites*. In order to guarantee that the contour extraction filter will work conveniently for these border edge sites of the contour matrix, virtual neighboring shapels outside this matrix have to be used. These virtual shapels are taken to be transparent. This way, if a border edge site has an opaque neighboring shapel inside the contour matrix, then a contour will be considered to pass through that edge site, and vice-versa. This effect can be clearly seen in Figure 6.7 (a) and guarantees the closure of the contours. On the other hand, the shapels inside the lost area are considered undefined and, therefore, it is not possible to define the contours in that area.

6.2.1.2 Contour Coupling

After the contours have been extracted from the corrupted (decoded) alpha plane, it is necessary to determine which of these contours actually go into the missing areas. For this, all the edge sites in the immediate border of the lost areas are examined to see which actually contain contours. These contour endings are counted. If no contour endings are found going into a given lost area, this lost area is assumed to be uniform and thus filled with the same value of the surrounding shapels. Due to the closure of the contours extracted from alpha planes, it is guaranteed that each contour going into the lost area must come out. Therefore, the number of contour endings around the lost area is always even, unlike the case where edges are extracted from a natural texture image instead of an alpha plane. This will make things easier when it comes to coupling the contours in pairs, which is the next concealment step.

Now that the contour endings going into the missing area have been determined, it is necessary to decide which endings will be connected to each other. This is referred to as contour coupling; at this stage, missing areas are treated independently, one after the other.

Contour coupling is done based on the angles between the tangent vectors to the contours going into the missing area and the straight segment that connects a pair of contour endings (or end points), as illustrated in Figure 6.8. For each contour ending, a tangent vector (whose magnitude is irrelevant for this purpose) is computed at the point where the contour goes into the lost area (i.e., the contour ending). The direction of this vector is chosen so that it points inside the lost area. To do the coupling itself, it is assumed that contours of natural objects show a slow direction variation. Therefore, it is expected that, when a given contour ending is coupled to the (matching) contour ending that belonged to the same contour segment, the sum of the absolute value of the angles between the tangent vectors and the straight segment that connects them (i.e., δ_A and δ_B in Figure 6.8) will be smaller than for any other combination.

6. Shape Error Concealment for Object-based Video Coding

This way, after all the valid contour coupling arrangements² are tried, the one that globally minimizes the cost function F is chosen, with F defined as:

$$F = \sum_i |\delta_{Ai}| + |\delta_{Bi}|, \quad i \in P. \quad (6.2)$$

In this definition, δ_A and δ_B are the angles between the two tangent vectors at the candidate contour endings A and B, respectively, and the straight segment that connects them, as shown in Figure 6.8. As for P , it is the set containing all the coupled contour pairs for each contour coupling arrangement; therefore, i represents one coupled contour pair in the context of the current coupling arrangement. Only so called valid contour arrangements are tried because some arrangements generate contours that cross each other and therefore create topological inconsistencies.

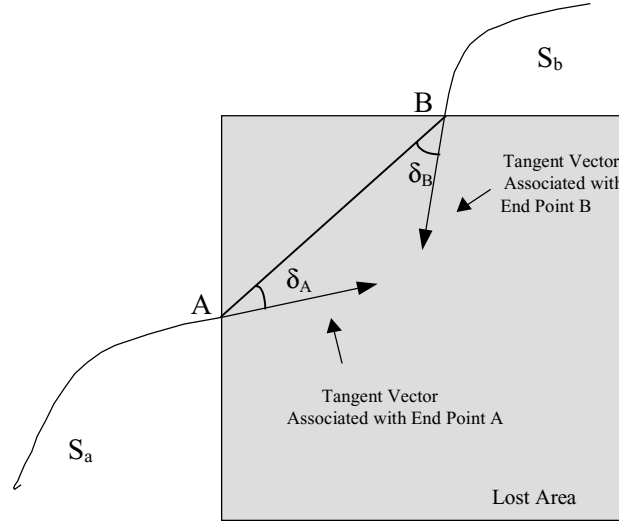


Figure 6.8 - Illustration of contour coupling

This contour coupling approach may not work so well if one or more of the lost contour has an abrupt direction change within the lost area. This would be a violation of the smoothness assumption that was made when developing the proposed technique.

6.2.1.3 Contour Recovery

The missing contours are interpolated with Bézier curves, which are cubic parametric curves, fully determined by four points as will be seen later on in this section. These curves were chosen over the natural cubic splines suggested in [Atzori98] for two main reasons. Both reasons are directly related to the fact that for the same number of points, natural cubic splines involve computations with larger matrices than Bézier curves. Therefore, Bézier curves are easier to manipulate and they are also much faster to compute than natural cubic splines. This latter argument is especially important when video applications are considered, since real-time is a mandatory decoding requirement. On the other hand, natural cubic splines have one more degree of continuity (at join points) than Bézier curves: they have second-order parametric continuity (C^2) instead of the first-order parametric continuity (C^1) associated with Bézier curves, which means that they are smoother. However, this should not

² A valid contour coupling arrangement is a set of contour ending pairs considering all the contour endings for the lost area in question (arrangement) that does not create a topological inconsistency in the alpha plane.

be a problem for a Bézier based solution since the contours are being interpolated over relatively small areas and, therefore, the difference should not even be visible. For a more detailed discussion on the advantages and disadvantages of Bézier curves over natural cubic splines, the reader may refer to [Foley94].

After two contour endings have been coupled according to the procedure defined in the previous section, the recovery of the contour inside the missing area with a Bézier curve is performed by following three steps:

- Determination of the four points that fully specify the Bézier curve relevant for the contour in question;
- Definition of an analytical continuous expression of the Bézier curve from the points determined in the previous step;
- Finally, computation of a discrete representation of the continuous Bézier curve determined in the previous step, in order to obtain the edge representation of the interpolating contour inside the lost area.

These three steps correspond to well-defined problems for which solutions are proposed in the next subsections.

6.2.1.3.1 Determination of the four Bézier points

As previously mentioned, Bézier curves are fully determined by four points, which include two end points and two additional control points. Each one of the control points is (always) associated to one end point. The interpolating curve goes from one end point to the other, with each control point indirectly specifying the tangent vector at its associated end point. The idea behind the technique proposed here is to use a Bézier curve to join the two coupled contour segments that go into the lost area; for that, the end points and the control points for each interpolating curve have to be determined.

Since the end points basically tell where the Bézier curve begins and ends, these shall be placed at the two points where the two available contour segments that are to be joined go into the lost area. These two end points shall be referred to as points A and B. This, by itself, will only guarantee that the interpolating curve and the two broken contour segments have G^0 geometric continuity³, which is the same as saying that they join, but no smoothness is guaranteed. For that, the control points have to be placed in such a way that the tangent vector to the curve on one side of the end point (available segment outside the lost area) is a scalar multiple of the tangent vector on the other side (interpolating curve inside the lost area). Here, the control points will be placed in such a way that the tangent vectors on either side of the end points are equal, which will guarantee first-order parametric continuity or C^1 continuity [Foley94] between the two available contour segments and the interpolating curve inside the lost area.

In order to determine the two control points, part of the two coupled contour segments outside the lost area, which shall be referred to as S_a and S_b , are each fitted to a cubic polynomial curve. Since the procedure is analogous for S_a and S_b , it shall be described only for one of them, for instance S_a . The actual length of the S_a contour segment starting from A, measured in terms of the number of contour points (i.e., consecutive edges in the edge

³ If two curve segments join together, the curve has G^0 geometric continuity. If the directions (but not necessarily the magnitudes) of the two segments' tangent vectors are equal at the join point, the curve has G^1 geometric continuity. G^1 continuity means that the geometric slopes of the segments are equal at the join point. More details on geometric continuity can be found in [Foley94].

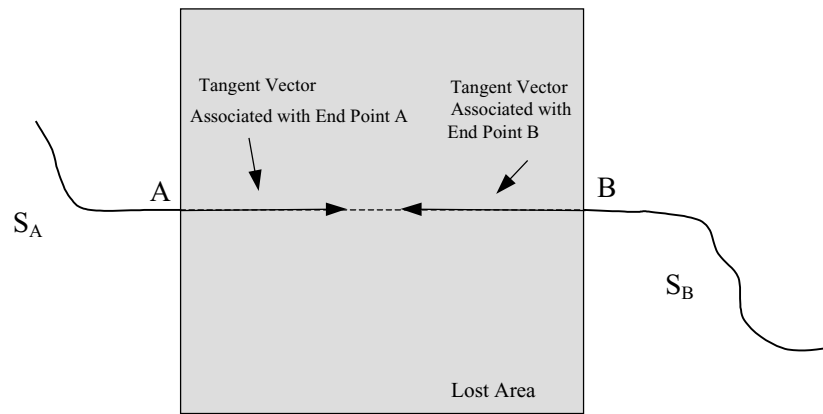
representation of the contour), that is used to do the above mentioned curve fitting is adaptively chosen in order to guarantee that the fitting error does not exceed a given threshold. This way, the curve fitting, which can be done by using any of the available curve fitting techniques, such as the one in [Glassner94] based on the least-squares method, is first tried with a certain initial length (or number of points). Then, the length used is progressively reduced (2 points at a time) until the curve fitting error, which is measured by the maximum squared distance between the fitted curve and the actual curve [Glassner94], becomes lower than a specified threshold. If a small threshold value is specified, which means that the fitted curve has to be a very accurate representation of the actual curve, the used S_a contour segment will typically end up being rather short (i.e., a small number of points). This happens because it is very hard to accurately approximate a long contour segment with a cubic curve, unless of course the curve already has a cubic behavior. If, on the other end, a large threshold value is specified, which means that the fitted curve can be a coarse approximation of the actual curve, the used S_a contour will typically end up being long (i.e., a large number of points), close to the initial length adopted. The value 25, which has been determined experimentally, represents a good compromise and, thus, it has been used for this threshold to provide the best shape concealment performance. It is proposed that the initial number of points N_I to be used for curve fitting on both S_a and S_b contour segments be determined by:

$$N_I = \text{round} \left(\frac{d_{AB}}{\cos \left(\frac{|\delta_A| + |\delta_B|}{2} \right)} \right) \quad (6.3)$$

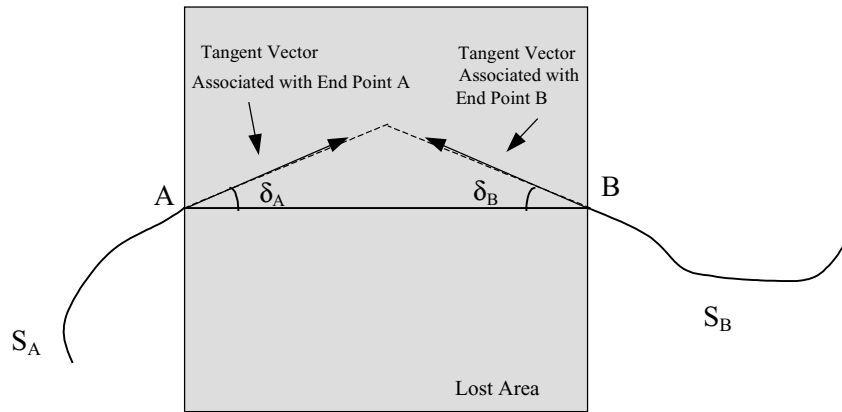
where d_{AB} is the distance between points A and B, δ_A and δ_B have the same definition as in Section 6.2.1.2 and $\text{round}()$ is a function that rounds its argument to the nearest integer. If the determined initial number of points N_I exceeds five times d_{AB} , then it will be truncated to that value in order to avoid excessively large N_I values. Additionally, if for some reason the S_a contour segment includes less points than the determined value of N_I , then N_I will be truncated to the number of available points.

To better understand Equation (6.3), it is important to mention that the number of points that is used for fitting a curve to S_A and S_B should be as close as possible to the number of points that is used to represent the interpolating contour that connects A to B because this will ensure that maximum smoothness is obtained in the transitions between existing contour segments and interpolating contour segments. The problem with this is that the interpolating contour is not known beforehand and, therefore, the actual number of points needed (i.e., its length) has to be estimated. This is where the cases in Figure 6.9 have to be considered. In Figure 6.9 (a), the tangent vectors associated with the two end points A and B are both parallel to the straight segment that connects point A to point B and, therefore, the distance d_{AB} between A and B is a very good estimate of the actual length of the interpolating contour. Therefore, in this case, the initial number of points to use for curve fitting should be equal to the distance d_{AB} . In Figure 6.9 (b), the tangent vectors are such that $\delta_A = \delta_B = \delta$ and, therefore, by extending the vectors it is possible to create an isosceles triangle. In this case, the sum of the top two sides of the triangle provides a good estimate of the length of the interpolated contour, which is given by Equation (6.3) by setting $\delta_A = \delta_B = \delta$. Finally, in Figure 6.9 (c), the tangent vectors are such that $\delta_A \neq \delta_B$ and, therefore, by extending the vectors it is possible to create a triangle, which is not isosceles. In this case, the sum of the top two sides of the triangle is also a good estimate of the length of the interpolating contour. Although it is

possible (and rather easy) to determine an exact expression for the sum of the top two sides of the triangle, this will still be only an estimate of the length of the interpolated contour. Since Equation (6.3), which corresponds to the sum of the top two sides of an isosceles triangles where the left and right angles are equal to the average of δ_A and δ_B , also provides a good enough estimate, it will be used instead. The absolute values in Equation (6.3) are used to take into account the cases where δ_A and δ_B have opposite signs, as illustrated in Figure 6.9 (d). As can be seen, even in this case the length of the interpolating contour can be estimated from the sum of the top two sides of the represented triangle, which is obtained by mirroring the tangent vector associated with end point B with respect to the line segment that connects points A and B.

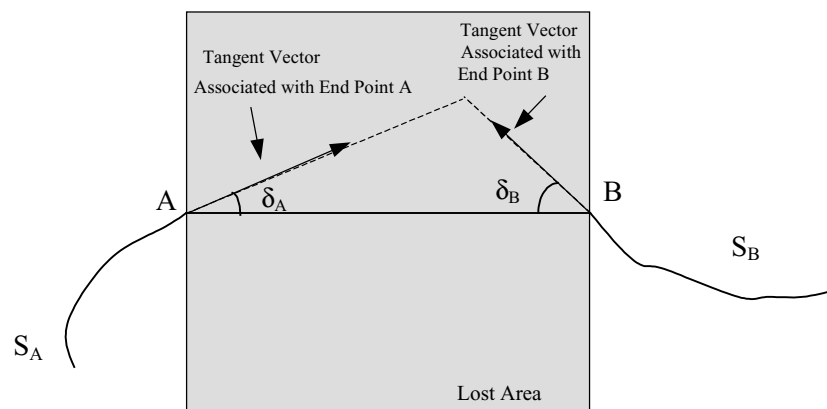


(a)

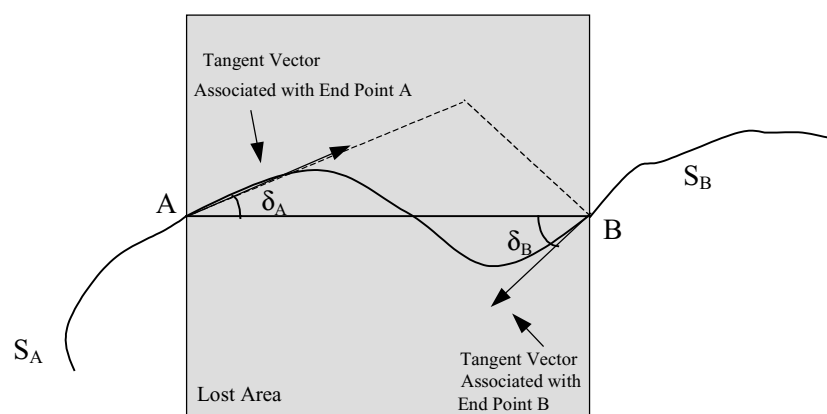


(b)

6. Shape Error Concealment for Object-based Video Coding



(c)



(d)

Figure 6.9 – Relevant cases for the determination of the initial number of points to be used for curve fitting

After the cubic polynomial that best fits S_a has been found, it can be re-written as a Bézier curve, with known end points and control points; to determine the end points and control points of the Bézier curve from the cubic polynomial coefficients, Equation (6.16) of the next section can be used. This way, it becomes possible to determine the control point of the interpolating Bézier curve associated with A. This is simply done by taking the symmetric (in a geometrical sense) of the control point of S_a (associated with A) with respect to A⁴. This guarantees that the tangent vector on each side of the end point A is the same, without having to actually compute the tangent vectors [Foley94]; then the same technique is used for point B. An illustration of this procedure is shown in Figure 6.10. At this stage, the 4 points necessary to determine the Bézier interpolating curve for the missing contour are finally available: the two end points, A and B, and the two control points inside the lost area.

⁴ The control point of the interpolating Bézier curve associated with A is the point located on the line that connects A and the control point of S_a at the same distance from A but inside the lost area.

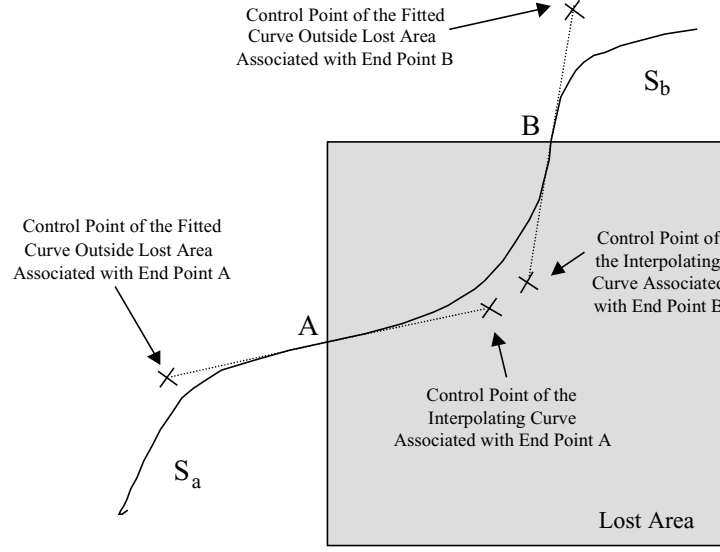


Figure 6.10 - Illustration of contour recovery

6.2.1.3.2 Determination of the Bézier continuous interpolating contour

After the four necessary points have been determined, generating the interpolating Bézier curve is straightforward and is briefly described below. Since Bézier curves belong to the class of cubic parametric curves, they can be written as:

$$Q(t) = [x(t) \ y(t)]^T \quad (6.4)$$

where $x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$, $y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$, and $t \in [0,1]$. By defining the matrix of coefficients of the two polynomials as

$$C = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \end{bmatrix} \quad (6.5)$$

and the parameter vector as

$$T = [t^3 \ t^2 \ t \ 1]^T, \quad (6.6)$$

it is possible to rewrite Equation (6.4) as

$$Q(t) = [x(t) \ y(t)]^T = C \cdot T. \quad (6.7)$$

However, this type of representation is not convenient here because it does not depend on the four points mentioned before. In order to do this, Bézier curves must be represented as:

$$Q(t) = G_B \cdot M_B \cdot T \quad (6.8)$$

where

$$G_B = [P_1 \ P_2 \ P_3 \ P_4], \quad (6.9)$$

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \ i = 1, 2, 3, 4, \quad (6.10)$$

6. Shape Error Concealment for Object-based Video Coding

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (6.11)$$

and T has the same definition as above.

In the expression above, G_B is called the geometry matrix and includes the four necessary points: P_1 and P_4 are the two end points, corresponding to points A and B in Figure 6.10, whereas P_2 and P_3 are the two control points (associated to P_1 and P_4 , respectively); P_2 and P_3 indirectly determine the tangent vectors at the end points P_1 and P_4 . On the other hand, M_B is the Bézier basis matrix that does not depend on any parameters and T is a vector with cubic polynomials in t , with t varying in the range $[0,1]$. This way, by replacing the adequate points in matrix G_B , an analytical expression is obtained for the interpolating Bézier curve, which goes from A to B when t varies from 0 to 1. However, this is not yet what is wanted since an edge representation of the interpolating contour is required instead.

Before proceeding with the edge representation of the interpolating contour, it is worth rapidly explaining how the four points in the Bézier geometry matrix in Equation (6.8) are related to the polynomial coefficients in Equation (6.7). This relationship can be obtained by equating the right-hand side of both these expressions:

$$C \cdot T = G_B \cdot M_B \cdot T, \quad (6.12)$$

which can be simplified to

$$C = G_B \cdot M_B. \quad (6.13)$$

This way, with Equation (6.13), the determination of the polynomial coefficients matrix when the four Bézier points that make up the geometry matrix are known is quite straightforward. On the other hand, to determine the Bézier points based on the polynomial coefficients, the following expression can be used:

$$G_B = C \cdot M_B^{-1} \quad (6.14)$$

where

$$M_B^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \quad (6.15)$$

By replacing Equation (6.15) in Equation (6.14), the following result is obtained for the G_B geometry matrix:

$$G_B = [P_1 \ P_2 \ P_3 \ P_4] = \begin{bmatrix} d_x & \frac{1}{3}c_x + d_x & \frac{1}{3}b_x + \frac{2}{3}c_x + d_x & a_x + b_x + c_x + d_x \\ d_y & \frac{1}{3}c_y + d_y & \frac{1}{3}b_y + \frac{2}{3}c_y + d_y & a_y + b_y + c_y + d_y \end{bmatrix} \quad (6.16)$$

which can be used to determine the four Bézier points when only the polynomial coefficients are known, as was the case in the previous section.

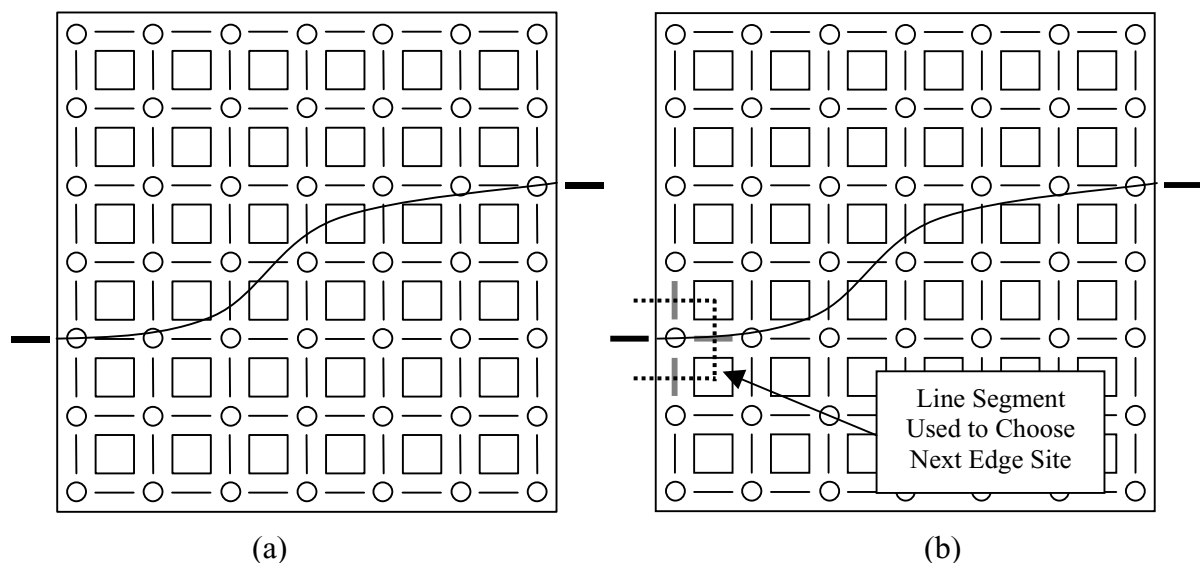
6.2.1.3.3

Determination of the edge representation of the Bézier interpolating contour

In the last section, the analytical expression of a continuous interpolating curve, joining points A and B, was determined. But, since an edge representation of contours is being used here, an algorithm had to be developed to allow going from this continuous analytical expression of a contour to a discrete edge representation. This conversion algorithm is best described as the following succession of steps:

1. Choose arbitrarily one of the two end points to be the starting point of the algorithm; which end point is actually chosen as the starting contour edge site is irrelevant for the result of the algorithm;
2. From this edge site, the contour has three possible candidate edge sites where it can go; these three candidates are checked, and the one closest to the continuous Bézier curve is chosen. The closest edge site is the one whose line segment is crossed by the continuous Bézier curve; the line segment for a certain edge site is an orthogonal line to the edge site at its center, joining the centers of the two neighboring shapels. Due to its smoothness, the continuous Bézier curve will not cross the line segments of more than one candidate edge sites. For this to happen, a very sharp direction change would be necessary in the continuous Bézier curve, which is smooth by definition;
3. After the closest edge site has been determined, the current interpolating position is advanced to that edge site. If the current position is the other end point, the algorithm stops and the edge representation of the interpolating contour is found. Otherwise, the algorithm goes back to step 2.

An illustration of the way the algorithm works is given in Figure 6.11. In Figure 6.11 (a), the continuous Bézier interpolating contour is shown; it can be seen that the continuous curve does not fall on the dual lattice C , and therefore is not adequate as an edge representation of the contour. In Figure 6.11, the end point chosen as the starting point for the edge representation algorithm was the one on the left; this choice is irrelevant and the same results would be obtained if the other end point were selected. In Figure 6.11 (b), the three candidate edge sites for the starting point are shown in gray, as well as the line segment used to select the horizontal edge as the closest one to the continuous Bézier curve. In Figure 6.11 (c), the position of the algorithm is moved to the edge site previously selected in Figure 6.11 (b), and the next three candidates are once again shown in gray. Finally, in Figure 6.11 (d), the complete result for the edge representation of the interpolating contour is shown.



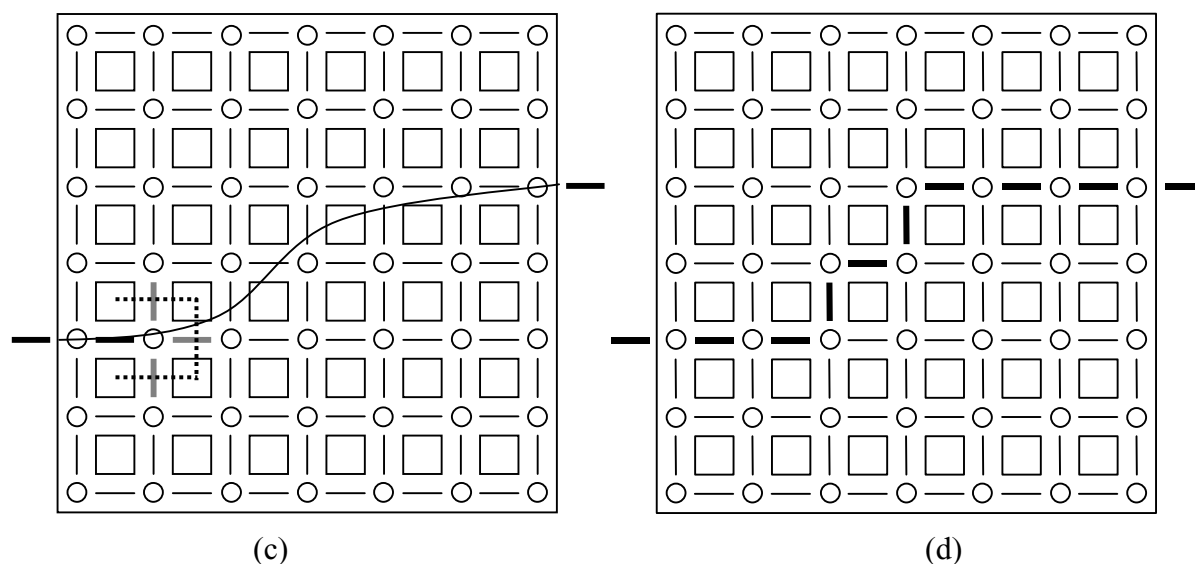


Figure 6.11 - Steps to obtain the discrete edge representation of the continuous Bézier interpolating contour: (a) Continuous interpolating contour; (b) First three edge site candidates; (c) First chosen edge site and next three candidate edge sites; (d) Final edge representation of the interpolating contour

6.2.1.4 Shape Filling

After the contour has been recovered, filling in the shape is a relatively easy task; it is simply a matter of taking the shapel in the top-left corner and scanning the contour image, line after line, from left to right and top to bottom while taking into account the contour position, which indicates transitions in the value of transparency. Each time a contour is found, the value of the shapel is set to the opposite of the previous one: transparent if the previous one was opaque and vice-versa. The first shapel in each line is determined based on the first shapel in the previous line, by inspecting the edge site that separates them.

The above procedure assumes that the shapel in the top-left corner is available, which is not always the case (e.g., when the top-left block in the image has been lost). However, for the cases in which this shapel is not available, it can be easily determined. To do so, the corrupted alpha plane is scanned (from left to right, top to bottom) until the first available opaque shapel that is encompassed by the recovered contour is reached. The transparency value of this shapel can then be used to determine the transparency value of the first shapel in the same line by proceeding backwards, which in turn can be used to determine the value of the top-left corner by proceeding upwards, if necessary (always checking the existence of contours and inverting the value of the shapel).

Some problems might arise if two or more interpolating contours overlap inside the lost area for some of the edge sites. For instance, in the case of two overlapping contours, a double transition will exist in some edge sites; therefore, when this edge site is reached the value of the next shapel should not be set to the opposite of the previous shapel, but to the same value. If this effect is not taken into account, strange concealed alpha planes can result as shown in Figure 6.12, which represents a lost area with two interpolated contours (contours A and B) that overlap on the third line. In Figure 6.12 (a), the values of the neighboring shapels are filled in according to the scan order described above. However, on the third line, the existing edge is interpreted as a single transition and the shapels change from white to black, which

results in an undesired artifact (shown in gray in Figure 6.12 (a)). In Figure 6.12 (b), the correct shape filling is shown taking into account the double transition.

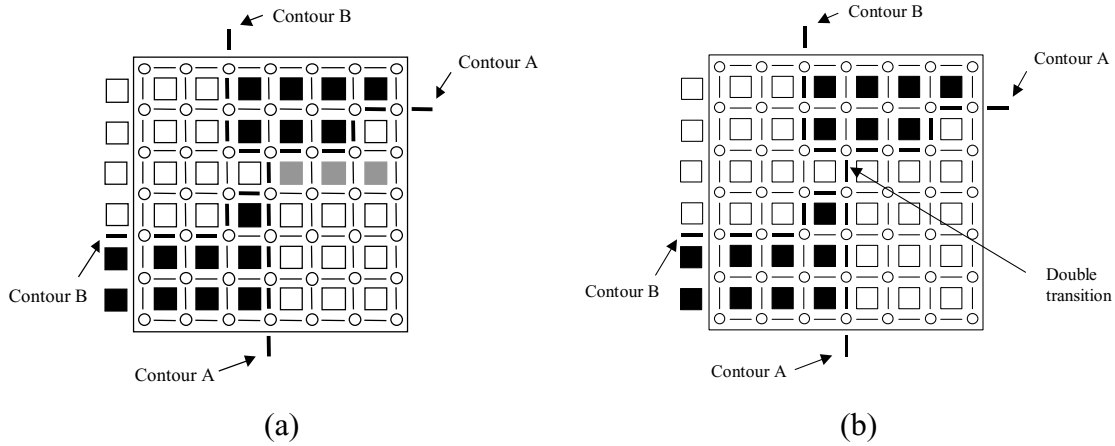


Figure 6.12 - Example of shape filling with overlapping contours: (a) Shape filling with artifact on the third line (shown in gray); (b) Correct shape filling

This problem can be easily solved by considering contour overlapping when the contours are being interpolated. Therefore, if two contours overlap for an edge site, this corresponds to a double transition, which in terms of shape filling is exactly the same as having no transition at all. If on the other hand, three contours overlap for a certain edge site, this corresponds to a triple transition, which in terms of shape filling has exactly the same behavior as a simple transition. This rule can be generalized for any number of overlapping contours. When an even number of contours overlap for an edge site, no shapel value transition exists; when an odd number of contours overlap for an edge site, it is the same as having only one contour going through that edge site and there is a shapel value transition. For the example in Figure 6.12, the two contours A and B overlap for one edge site. According to the rule above, this means that no shapel value transition exists, which is illustrated in Figure 6.13. Figure 6.13 (a) shows that contour overlapping occurs for one edge site and the corresponding correct shape filling is shown in Figure 6.13 (b).

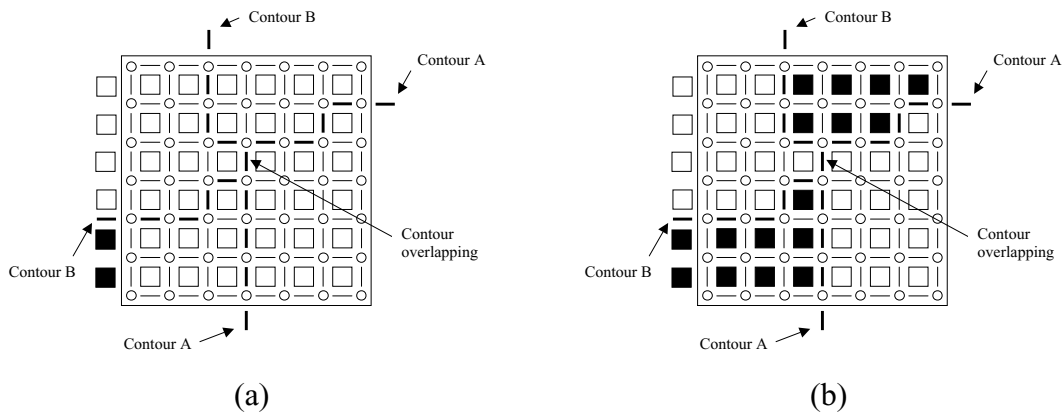


Figure 6.13 - Correct shape filling with overlapping contours: (a) Contour overlapping in one edge site; (b) Correct shape filling

6.2.2 Performance Evaluation

In order to evaluate the proposed spatial shape concealment technique, several MPEG-4 bitstreams have been tested as examples, each bitstream containing one video object encoded at a given bit rate, syntactically according to the MPEG-4 Core Visual Object Type [MPEG4-V]. The exact bit rate value used is unimportant because it does not influence the quality of the shape since lossless shape coding was used⁵. On the other hand, the quality of the texture data is highly dependent on the used bit rate. Additionally, the used MPEG-4 bitstreams only contain intra coded VOPs, for the reasons explained next. Although the proposed concealment technique can theoretically work for any type of VOP, it should be reminded that it is based on the interpolation of contours, which can only be done effectively over relatively small areas of the alpha plane. Since in intra coded VOPs the video packets that will potentially be corrupted typically correspond to much smaller areas of the alpha plane than in inter coded VOPs⁶, this makes the technique especially suited to conceal shape information in intra coded VOPs. In fact, in inter coded VOPs, large percentages of the alpha plane can be lost (e.g., 50%), making it quite unrealistic to use (only) contour interpolation. For this case, the use of a technique that also uses the information from other time instants, such as the one proposed in Section 6.3, is probably much more effective. This way, by using bitstreams with only intra coded VOPs, it becomes easy to test the proposed technique on many different VOPs.

As for the remaining test conditions used for the performance evaluation of the proposed spatial shape error concealment technique, they are detailed in the following, grouped into several categories:

- **Test Video Object Sequences** – The tested video object sequences are the CIF versions of the Akiyo, Bream and Stefan video objects; all 300 frames at 30 fps have been used for each sequence. In these video object sequences, as explained above, all VOPs have been intra coded.
- **Error Resilience Tools** – In order to make the test bitstreams more resilient to errors, several MPEG-4 error resilience tools have been used at the encoder. The used tools are resynchronization markers (which divide the VOP data into independently decodable video packets) and data partitioning with reversible variable length codes. The size of the video packets in bits is usually chosen (e.g., in MPEG) so that there are on average four VPs in each VOP, as done in Chapter 5. However, this criterion is used for bitstreams that are mainly composed of inter coded VOPs, which is not the case here. When using this criterion for a bitstream that is mainly composed of inter coded VOPs, a possible way to determine the adequate size of the video packet in bits is to determine first the average number of bits spent on each VOP (by dividing the average bit rate by the frame rate) and then dividing it by four. This way, by creating video packets based on this number of bits, every time an intra coded VOP appears in the bitstream, since it usually consists of many more bits than the inter coded VOPs, it will also have many more VPs. This is actually very convenient because, due to their importance, intra coded VOPs need the extra error resilience provided when more VPs are used. Here, since the same cannot be done because only intra coded VOPs are

⁵ MPEG-4 shape coding is always lossless; however, distortion is sometimes introduced on purpose by simplifying/filtering the shape data before encoding (e.g., for rate control).

⁶ To encode an intra coded VOP, the amount of bits needed is typically much larger than for inter coded VOPs. However, the size of the video packets is typically chosen to be approximately the same, in terms of the number of bits, for both types of VOP. This means that for intra coded VOPs each video packet will correspond to a much smaller fraction of the whole VOP than for inter coded VOPs.

used, a VP size of 600 bits was used for the three video objects. This VP size was chosen so that the number of VPs in each intra coded VOP is similar to the one that would be obtained for intra coded VOPs in a typical bitstream mainly composed of inter coded VOPs.

- **Error Simulation** – To simulate channel errors, while decoding the bitstream, the decoder randomly ignores the video packets with a given packet loss rate (packet losses are independent and identically distributed). This procedure was adopted instead of adding the errors directly to the bitstreams, because it allows to evaluate the performance of the concealment technique independently of the decoder error detection capabilities. Then, the proposed spatial shape error concealment technique has been applied to the corrupted decoded alpha planes to recover the missing shape data. For these tests, four different video packet loss rates have been considered: 1%, 5%, 10% and 20%, corresponding to increasing amounts of channel errors. These rates were adopted because, among the packet loss rates that are practically relevant, they represent the range of rates where the proposed shape concealment technique will be most useful; at rates below 1%, artifacts are so few that the improvement obtained with the proposed technique will be negligible, while at higher rates a video communication with acceptable quality becomes extremely difficult, if not impossible. For each one of these loss rates, each one of the tested video objects has been decoded 50 times (i.e., corresponding to 50 different error patterns or runs).
- **Quality Evaluation Tools** – In order to evaluate the performance of the proposed technique in terms of shape recovery, a shape quality metric is needed. During the development of the MPEG-4 standard, a shape quality metric was used within MPEG to evaluate the performance of several proposed lossy shape coding techniques [N1471] and segmentation⁷ algorithms [Wollborn98]. This metric is based on the ratio between the number of shapels that are different in the original and reconstructed alpha planes and the total number of original shapels:

$$Dn = \frac{\text{Number of different shapels in the original and reconstructed VOPs}}{\text{Number of opaque shapels in the original VOP}}. \quad (6.17)$$

Similarly to what was done in Chapter 5, Dn can also be expressed as a percentage,

$$Dn[\%] = 100 \times Dn. \quad (6.18)$$

Although this metric may not be able to fully express the subjective impact of a corrupted alpha plane on the user, it is still very useful because it provides a simple objective way to measure the amount of shape distortion. Therefore, it will be used here to evaluate the performance of the proposed shape concealment technique. In addition to the numeric shape quality values associated to the metric above, the concealment results will be illustrated by several reconstructed alpha planes, which should allow the reader to subjectively evaluate the performance provided. When visual results are shown, these will include the original, the corrupted and the concealed alpha planes (in the bounding box of the video object).

In Table 6.1, three different types of Dn values are shown for the mentioned video object sequences for four different video packet loss rates: Dn_{low} , Dn_{avg} and Dn_{high} . While Dn_{low} and Dn_{high} correspond, respectively, to the average Dn value associated with the best and the worst runs in terms of shape quality, Dn_{avg} corresponds to the mean of the average Dn values

⁷ Although segmentation techniques are not standardized by MPEG, some had to be developed in order to create segmentation masks for the object-based coding of the test sequences.

6. Shape Error Concealment for Object-based Video Coding

associated with the 50 different runs for each test case. The average Dn value associated with a given run is simply the temporal average of the Dn values for all the VOPs in the video object sequence, defined as:

$$Dn = \frac{1}{N_{VOP}} \sum_{i=0}^{N_{VOP}-1} Dn_i \quad (6.19)$$

where N_{VOP} is the total number of VOPs in the test sequence and i is the index corresponding to the VOP number. This way, the Dn_{low} and Dn_{high} figures correspond, respectively, to the temporal averages of the Dn values in the best and worst runs (of the 50) in terms of shape quality. The Dn_{avg} figure corresponds to the mean of the 50 temporal averages of the Dn values. Since it is possible for the decoder to lose a complete VOP while decoding a bitstream (e.g., due to errors in the VOP header data), if this happens, the lost VOP is replaced with the previously decoded VOP, which will then be used for the Dn computations, as was done in Chapter 5. This guarantees that at the end the number of original and decoded VOPs is the same and thus a synchronized comparison is performed.

As can be seen in Table 6.1, even for relatively high video packet loss rates, the Dn values remain relatively low. In most cases, the Dn values are so low that they correspond to hardly noticeable artifacts (i.e., Dn_{avg} values around 1% and below). This is the case for the Akiyo and Bream video objects with packet loss rates of 1%, 5% and 10% and for the Stefan video object with packet loss rates of 1% and 5%. For higher packet loss rates, such as 20% for Akiyo and Bream and 10% for Stefan the artifacts start to become more visible (i.e., Dn_{avg} values higher than 1% but still below 3%). And, finally, for the Stefan video object with a packet loss rate of 20%, the artifacts become clearly visible but the provided shape quality is still perfectly tolerable for critical environments such as mobile networks (the Dn_{avg} value is 6.16%). These results clearly reflect the intrinsic shape concealment difficulty of the tested video objects, by ranking the Akiyo object as the easiest to conceal, followed by Bream and, finally, Stefan. This can be explained by considering the shape smoothness of the tested objects: Akiyo is the smoothest followed by Bream and, finally, Stefan.

Table 6.1 – Dn values for the tested video object sequences

Video packet loss rate	Dn [%] ($Dn_{low}/Dn_{avg}/Dn_{high}$)								
	Akiyo			Bream			Stefan		
1%	0.01	0.03	0.04	0.03	0.04	0.09	0.10	0.20	0.32
5%	0.12	0.16	0.25	0.19	0.26	0.40	0.68	1.03	1.57
10%	0.28	0.40	0.59	0.50	0.65	0.84	1.79	2.35	2.83
20%	0.92	1.25	1.57	1.47	1.97	2.63	5.30	6.16	7.07

In order to illustrate the results in Table 6.1, some corrupted and concealed alpha planes have been chosen. To start this illustration, the Akiyo video object, which is the easiest to conceal among the tested ones and whose first VOP alpha plane is shown in Figure 6.14 (a), was chosen. In the remainder of Figure 6.14, three different corrupted versions of the alpha plane in Figure 6.14 (a) are shown, followed by the corresponding concealed alpha planes. These three corrupted versions correspond to three of the 50 different error patterns used above for a video packet loss rate of 20%, which corresponds to the most critical error condition tested. As can be seen in Figure 6.14, the obtained results still have a pleasing subjective impact on

the viewer, in the sense that no annoying artifacts appear. In fact, this is even true for Figure 6.14 (c), which exhibits a clearly visible shape distortion in the lower corners of the Akiyo object that have been rounded out. This is a case where the slow direction change assumption has been violated, not due to the object itself but to the corners of the image. As for the D_n values for the shown concealed alpha planes, they are very low (typically below 1%) except for error pattern 2, which is especially critical because the two lower corners have been lost: 0.59% for pattern 1, 2.89% for pattern 2 and 0.59% for pattern 3.

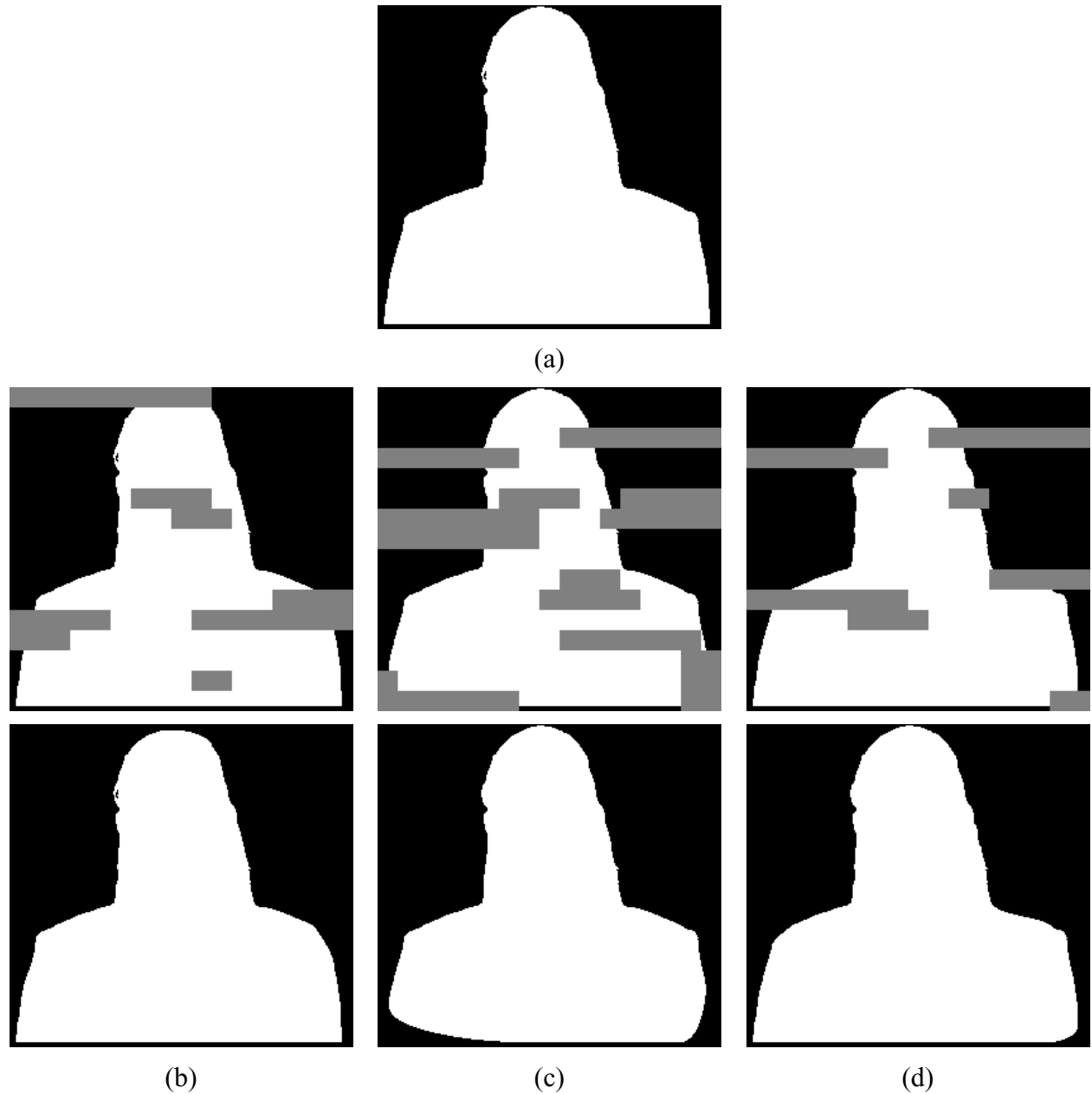


Figure 6.14 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Akiyo video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3

The next video object chosen to illustrate the performance of the proposed error concealment technique is the Bream video object, whose first VOP alpha plane is shown in Figure 6.15 (a). Similarly to what was done for the Akiyo video object, Figure 6.15 shows three different

corrupted versions of the alpha plane in Figure 6.15 (a), followed by the corresponding concealed alpha planes. These three versions also correspond to three of the 50 different error patterns used above for a video packet loss rate of 20%. As can be seen in Figure 6.15, the obtained results also have a pleasing subjective impact on the viewer, in the sense that no annoying artifacts appear. As for the Dn values for the shown concealed alpha planes, the results are also very low (typically below 1%) with the exception of error pattern 3: 0.47% for error pattern 1, 0.21% for error pattern 2 and 1.90% for error pattern 3. The Dn value associated with error pattern 3 is slightly higher because one corner has been removed where the lower fin joins the body of the fish.

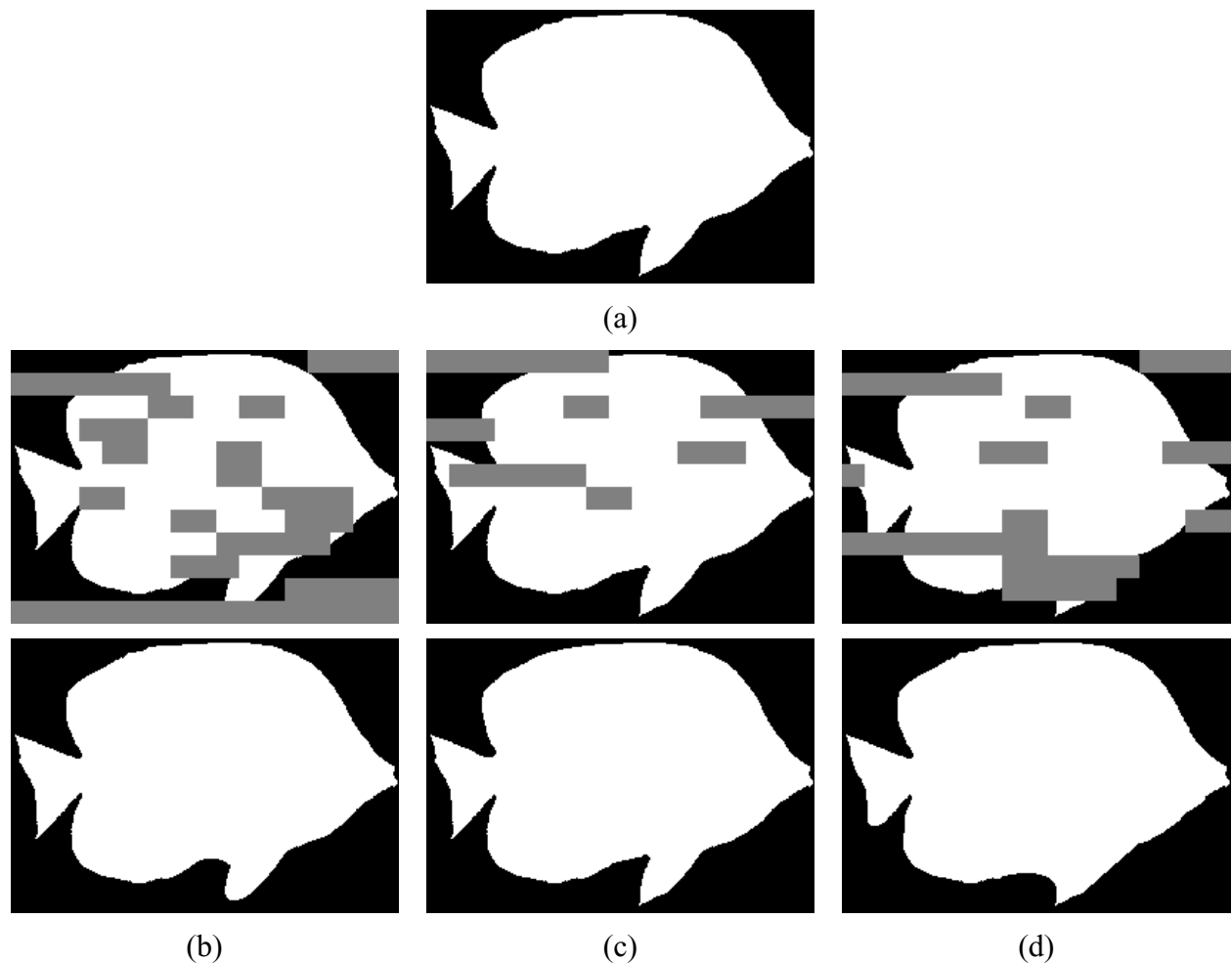


Figure 6.15 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Bream video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3

The last video object chosen to illustrate the performance of the proposed shape error concealment technique is the Stefan video object, whose first VOP alpha plane is shown in Figure 6.16 (a). For this video object, four different corrupted versions of the alpha plane in Figure 6.16 (a) are shown in the remainder of Figure 6.16, followed by the corresponding concealed alpha planes. These four versions correspond to four of the 50 different error patterns used above for a video packet loss rate of 20%. In this case, the artifacts are rather visible but the results may still be acceptable for critical environments such as mobile networks, depending for example if the object is part of a segmented scene and thus has to fit with other objects in the scene. In terms of the Dn metric, the values for the shown concealed

alpha planes are, as expected, higher than for the previous video objects because this shape exhibits some faster direction changes which makes it more difficult to conceal: 5.47% for error pattern 1, 2.86% for error pattern 2, 4.50% for error pattern 3 and 9.42% for error pattern 4.

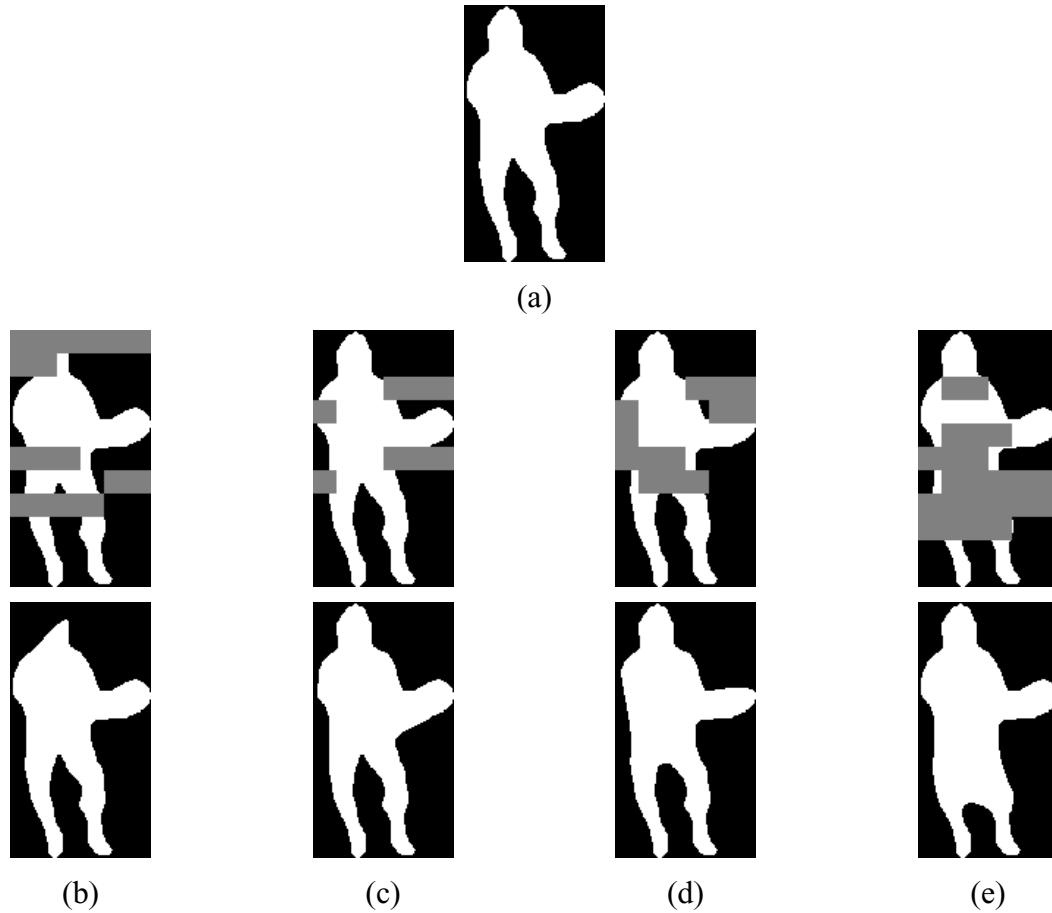


Figure 6.16 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Stefan video object with a video packet loss rate of 20% – (a) Uncorrupted original; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4

Still for the first VOP alpha plane of the Stefan video object in Figure 6.16 (a), another set of four different corrupted and concealed shape versions is shown in Figure 6.17. These four versions correspond to four of the 50 different error patterns used above for a video packet loss rate of 10%. In this case, except for error pattern 3 (which is especially difficult to conceal), the artifacts are much less visible than above. The case depicted in Figure 6.17 (c) is clearly an example of a very critical situation, where the use of shape information from past temporal instants would be beneficial. In terms of the Dn metric values for the shown concealed alpha planes, the results are, as expected, lower than for the previous case of a video packet loss rate of 20%: 0.64% for error pattern 1, 0.60% for error pattern 2, 4.88% for error pattern 3 and 1.50% for error pattern 4. These results would be further improved if the same shape had been transmitted with a larger resolution because the contour variations per block would be smaller, thus making easier the shape concealment task.

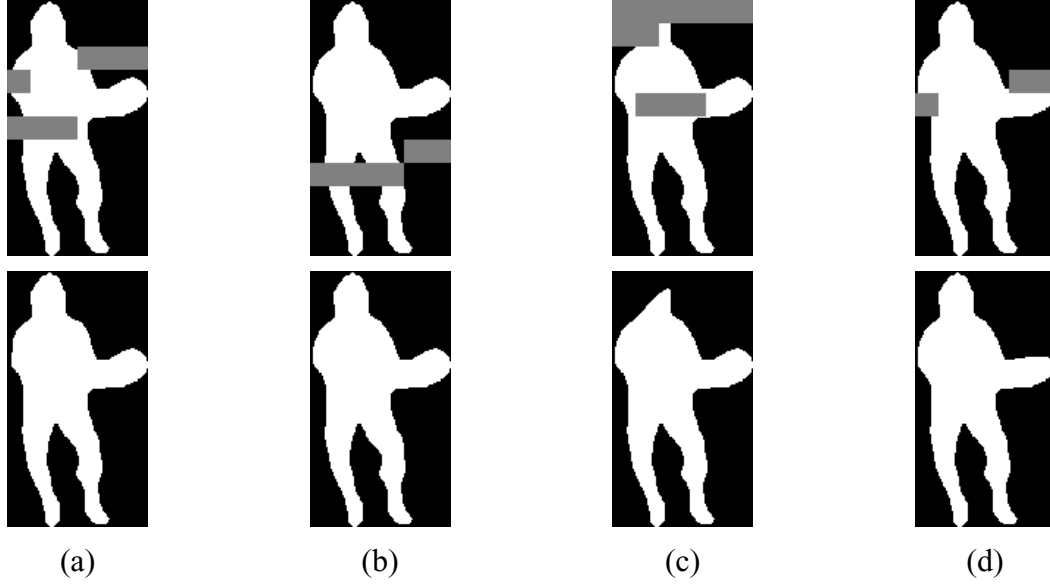


Figure 6.17 – Examples of corrupted and corresponding concealed alpha planes for the first VOP of the Stefan video object with a video packet loss rate of 10% – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4

In addition to these results, the proposed technique has also been compared to the technique proposed in [Shirani00], which was already described in Chapter 3. For comparison purposes, the shape concealment technique proposed in this chapter was applied to one of the corrupted shape examples mentioned in [Shirani00] (i.e., the first VOP of the Bream video object sequence), whose original alpha plane can be seen in Figure 6.18 (a). To make the comparison meaningful, exactly the same 16×16 shape blocks that were lost in [Shirani00] have also been considered lost here. Therefore, the corrupted alpha plane shown in Figure 6.18 (b) is the same that was also used in [Shirani00], which corresponds to a shape block loss rate of 25%. While the concealed alpha plane with the contour interpolation technique proposed here is shown in Figure 6.18 (c), the concealed alpha plane with the alternative technique proposed in [Shirani00] is reprinted in Figure 6.19. Additionally, the same shape similarity metric that was used in [Shirani00] is also used in the following for comparison purposes; this metric is defined as:

$$\eta = 1 - \frac{n_d}{n_t} \quad (6.20)$$

where n_d is the number of shapels that are different in the concealed and the original shapes, and n_t is the total number of shapels in the bounding box (here 272×192). The first difference between this and the Dn metric used in this chapter is that with this metric, the closer the value is to 1 the higher the similarity between the two shapes under comparison. The second one is that while in this metric the number of different shapels between the concealed shape and the original one is divided by the total number of shapels in the bounding box, in the Dn metric it is divided only by the number of opaque shapels. Since the subjective impact a corrupted shape has on the user is clearly dependent on the percentage of erroneous shapels with respect to the size of the object, it is believed that the Dn metric is better at expressing this impact.

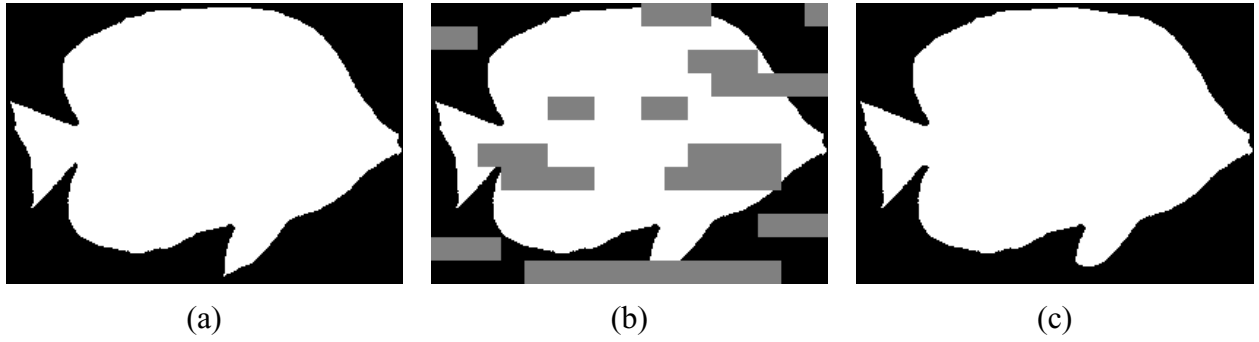


Figure 6.18 – Shape of the first VOP of the Bream video object – (a) Original; (b) Corrupted; (c) Concealed



Figure 6.19 – Shape of the first VOP of the Bream video object concealed with the scheme proposed in [Shirani00]

In terms of numerical results, while the shape concealment technique proposed in this chapter led to a metric value of $\eta=99.7\%$, the technique proposed in [Shirani00] obtained $\eta=99.0\%$ apparently showing a poorer performance. Perceptually, the results obtained with the shape concealment technique proposed in this chapter seem to be more visually pleasing (at least for the cases for which a comparison could be made). For instance, in the results presented in Figure 6.19, the top of the fish is exaggeratedly flat and the lower fin shows a clearly visible artifact.

6.3 Temporal Shape Error Concealment

As was seen in Section 6.2, there are some clear cases where the shape information from previous time instants would be clearly beneficial in terms of shape error concealment. Therefore, the second shape error concealment technique proposed in this chapter is a temporal object-level shape error concealment technique. As the spatial technique proposed in Section 6.2, this technique is also designed for object-level concealment, as opposed to scene-level concealment. Moreover, since it is a temporal (or inter) technique, it has to rely on information from other, typically previous, time instants. This basically means that it uses the available information for the video object being concealed from other time instants. In addition to this, and also as in Section 6.2, the proposed technique relies on the information provided by a previous error detection stage, which means that it has no error detection capabilities and only focuses on the error concealment itself. This approach is adopted for the same reasons already explained in Section 6.2 for the spatial shape error concealment technique.

Similarly to the spatial concealment technique proposed before, the proposed temporal shape error concealment technique can be applied to any corrupted video object whose shape can be represented by means of a binary alpha plane or to the shape support of video objects with gray scale alpha plane, such as for MPEG-4 video with binary or gray scale shapes [MPEG4-V]. Although the proposed technique is very adequate for usage in MPEG-4 video decoders, it is by no means limited or anyhow tuned to this type of decoders.

6.3.1 Proposal for a Temporal Shape Error Concealment Technique

In this section, the idea of using global motion to conceal shape errors at the decoder, as was done in [Salama02] (see Chapter 3), is extended while eliminating the major drawback of relying on specific information received from the encoder. In the technique proposed here, all the processing is performed at the decoder and, therefore, the decoder does not have to rely on any additional information sent from the encoder. This removes the need to normatively define the concealment information to be sent by the encoder (the encoder has now nothing to send) since the proposed technique can be used even if the encoder knows nothing about the concealment techniques implemented at the decoder, which is usually the case when terminals from different manufacturers are used, even in the context of a specific coding standard since concealment techniques are not normative. This way, the additional bit rate that would be used by the encoder to send the extra stream can be used for something else, such as increasing the shape intra refreshment rate or improving the texture quality.

Similarly to what was done for the spatial shape error concealment technique previously proposed in this chapter, it is assumed that the shape alpha planes have been encoded with some kind of block-based technique, such as in the MPEG-4 Visual standard [MPEG4-V]. It is also considered that bitstream errors will manifest themselves in the form of bursts of consecutive erroneous blocks, which have to be detected by an error detection stage before applying the concealment itself.

Since, in this proposal, the global motion parameters are no longer sent by the encoder, this means that they have to be locally computed at the decoder. This is possible because the video data at a given time instant is usually not completely corrupted, only some parts of it are. This happens because, as mentioned above, errors typically manifest themselves as bursts of consecutive erroneous blocks. With the remaining correctly decoded shape and texture data, the decoder can extract the global motion parameters.

After the global motion parameters have been determined, they can be used to global motion compensate the alpha plane of the previous time instant. The obtained global motion compensated alpha plane is considered as the reference alpha plane and it will be used to perform the concealment. This way, the erroneous shape blocks in the corrupted alpha plane being concealed can be simply replaced by the co-located shape blocks in the reference alpha plane. Assuming that the global motion model can accurately describe the shape motion, this alone should be able to produce very good results. However, in many cases, the shape motion cannot be accurately described only by global motion, due to the existence of local motion in some areas of the shape. Therefore, to avoid significant differences when concealing erroneous blocks in areas with local motion, an additional local motion refinement scheme has been introduced. In this scheme, the available blocks surrounding an erroneous shape block are used to determine if any local motion exists; if so, a local motion vector to be used to find a replacement block in the previous alpha plane for the erroneous shape block is estimated (in this case, the global motion compensation will not be used, which means that the local motion supersedes the global motion). As a result of this whole concealment

process, it is possible that small regions appear in the concealed alpha plane, which can have a very negative impact on the user. Therefore, in order to remove those regions, a final post-processing step has also been introduced.

It should be noted that the shape concealment approach here proposed is significantly different from the one used in [Salama02] and much more useful in terms of practical implementation. In [Salama02], as described in Chapter 3, to perform the concealment of a corrupted alpha plane, the decoder needs to perform a series of operations on the broken contour extracted from the corrupted alpha plane and the contour extracted from the previous alpha plane. In particular, it needs to (global) motion compensate the contour obtained from the previous alpha plane, leading to the reference contour. Then, it needs to fix the broken contour obtained from the corrupted alpha plane by copying the missing contour segments from the reference contour. Only after the broken contour has been fixed, can the concealed alpha plane be obtained. Here, however, the concealment is directly done on alpha planes, which avoids a number of complex problems typically associated with the processing of contours. For instance, when fixing the broken contour by copying a contour segment from the reference contour, it is very likely that the copied contour segment will not connect with the already existing correctly decoded contour, especially if the global motion model is not able to perfectly describe the existing motion. This will create very serious problems when generating the concealed alpha plane from the fixed contour, which do not happen with the proposed technique by directly working with the alpha planes. Additionally, in [Salama02], no local motion refinement is used, which has a strong impact on the concealment results for all objects for which simple global motion compensation is not enough.

The block diagram for the proposed temporal shape concealment technique is presented in Figure 6.20; the input is a corrupted alpha plane with several lost blocks, typically arranged in bursts (see Figure 6.21). In order to be concealed, the corrupted alpha plane will go through five different consecutive steps. At the end of the concealment process, the output is a fully concealed alpha plane.

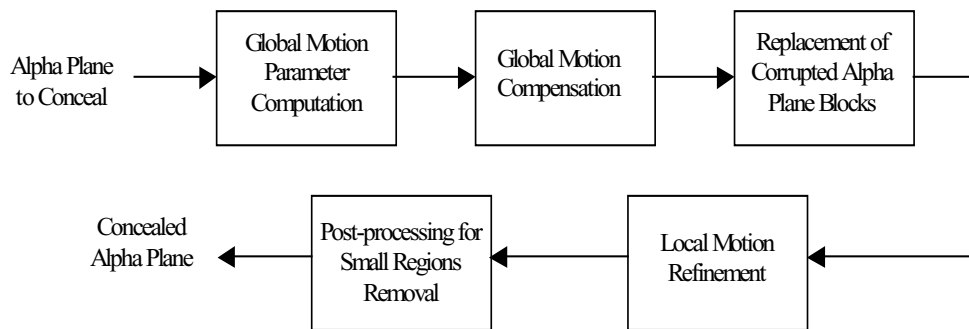


Figure 6.20 – Proposed temporal shape concealment process

Each block in Figure 6.20 corresponds to one of the five main steps of the proposed shape concealment process, which are:

- **Global motion parameters computation** – This module has the task of computing (at the decoder) the global motion parameters by considering the correctly decoded shape blocks in the corrupted alpha plane and the previously decoded (and concealed) alpha plane.

- **Global motion compensation** – This module is responsible for compensating the previously decoded alpha plane with the global motion parameters determined above.
- **Replacement of corrupted alpha plane blocks** – This module is in charge of finding replacements for the corrupted alpha plane blocks in the global motion compensated previous alpha plane; this is simply done by copying the co-located shape blocks from the global motion compensated alpha plane.
- **Local motion refinement** – This module is responsible for refining the shape concealment results in the object areas where local motion is considered to exist.
- **Post-processing for small regions removal** – This module is the last of the concealment process and is in charge of removing any non-relevant small regions that may have appeared due to the previous concealment steps. Since, as shall be explained in Section 6.3.1.5, this operation involves the risk of actually eliminating important information, this filter can always be turned off if the user should so wish.

In order to understand better the shape concealment process, an illustrative example will be given before detailing each of the steps above in the following sections; each of these steps corresponds to a well defined and rather independent operation. As explained above, the input to the temporal error concealment chain depicted in Figure 6.20 is an alpha plane where some blocks have been lost (i.e. the lost area); these blocks are shown in gray throughout the example. For this example, it will be considered that the alpha plane in Figure 6.21 (a) has been corrupted, as illustrated in Figure 6.21 (b). The first step is to consider the previous VOP, shown in Figure 6.21 (c), and determine the global motion parameters. With these parameters, the previous VOP is motion compensated and Figure 6.21 (d) is obtained. After that, the corrupted blocks are replaced with the corresponding ones in the motion compensated previous VOP, which gives the concealed alpha plane shown in Figure 6.21 (e). In this case, since the global motion model is able to accurately describe the shape motion, no refinement was needed for local motion.

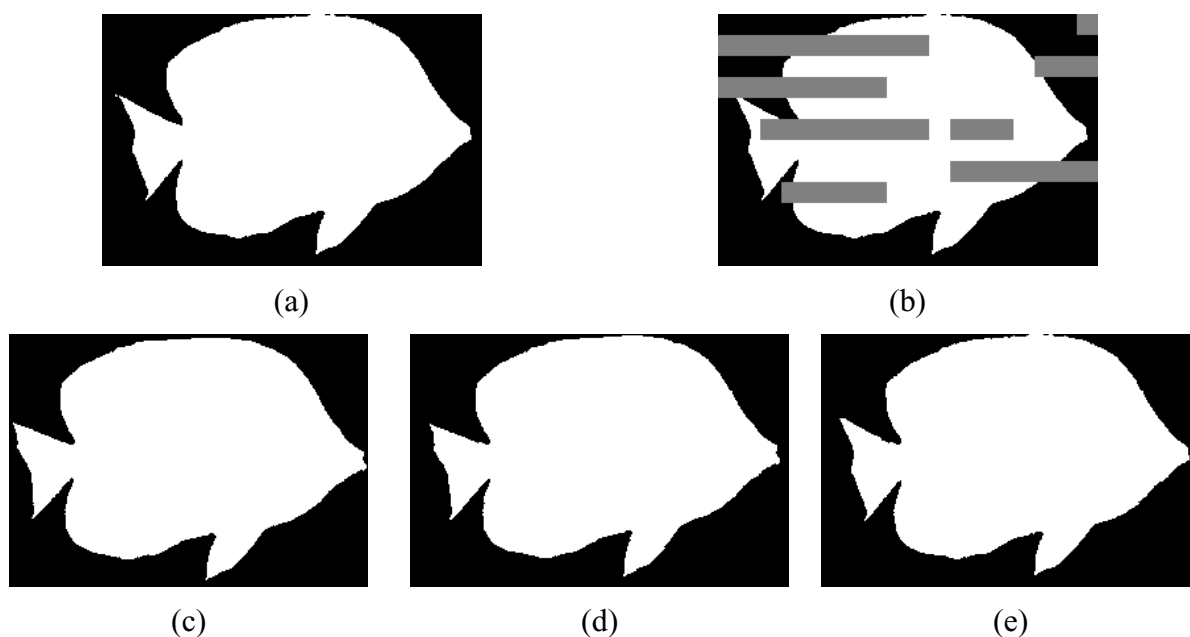


Figure 6.21 – Exemplifying the steps of the temporal shape concealment process for the Bream video object (CIF, 10 fps): (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Previous alpha plane; (d) Motion compensated previous alpha plane; (e) Concealed alpha plane without local motion refinement

However, in some other sequences, the objects may have a lot of local motion in addition to the global motion. For instance, in Figure 6.22, this is clearly the case and concealment should no longer be done by simply replacing the corrupted shape blocks with the co-located blocks from the global motion compensated previous alpha plane. As can be seen from Figure 6.22 (e), this would lead to very annoying artifacts. Therefore, to avoid this, an additional refinement step is needed to deal with the local motion. The results with this refinement are shown in Figure 6.22 (f), which no longer exhibits the most annoying artifacts.

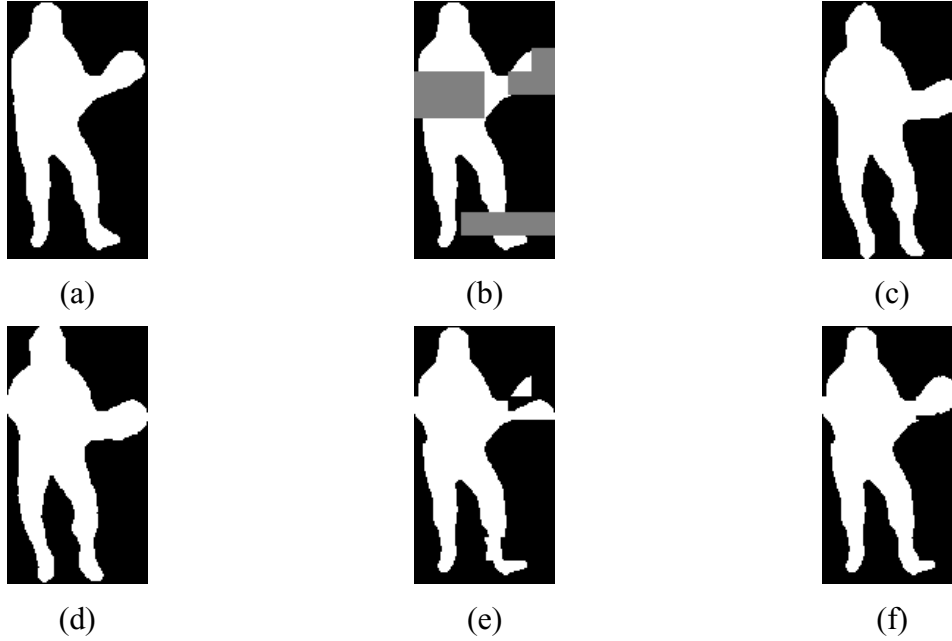


Figure 6.22 – Exemplifying the steps of the temporal shape concealment process for the Stefan video object (CIF, 15 fps): (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Previous alpha plane; (d) Motion compensated previous alpha plane; (e) Concealed alpha plane without local motion refinement; (f) Concealed alpha plane with local motion refinement

Additionally, as mentioned above, it is possible for some small regions to appear in the concealed alpha plane as a result of the concealment process. Since their removal is a conceptually very simple operation and an illustrative example is already given in Section 6.3.1.5, this operation will not be illustrated here.

6.3.1.1 Global Motion Parameters Computation

Before computing the global motion parameters, a global motion model has to be chosen from those available in the literature. The differences between the existing global motion models are basically related to their complexity and, thus, their capacity to describe more complicated motion trajectories. Since the global motion model will be used to describe the motion of the shape data from one time instant to the next, which is typically quite simple, a simple model should be enough. However, it is always possible to replace the used global motion model with a more complex one, since the proposed technique does not directly depend on the type of motion model that is used, as long as one is used. This would eventually allow more sophisticated decoders to have more powerful error concealment capabilities at the cost of some additional computational power.

6. Shape Error Concealment for Object-based Video Coding

Taking into account the considerations above, the affine four parameter model, which is certainly the simplest and most widely used global motion model, was adopted. The choice of this simple model seems adequate for the motion trajectories in question and should facilitate the deployment of this concealment technique by contributing as little as possible to increase the complexity of the decoder. A description of how this model is derived can be found in [Zakhor93]. The types of motion that is possible to adequately describe with this motion model are:

- i) Change of the camera focal length (i.e., zoom or scale);
- ii) Rotation around an axis normal to the camera axis (i.e., pan or tilt);
- iii) Rotation around the camera axis.

In order to help the reader visualize what these three motion types correspond to, the various types of possible camera motions are illustrated in Figure 6.23.

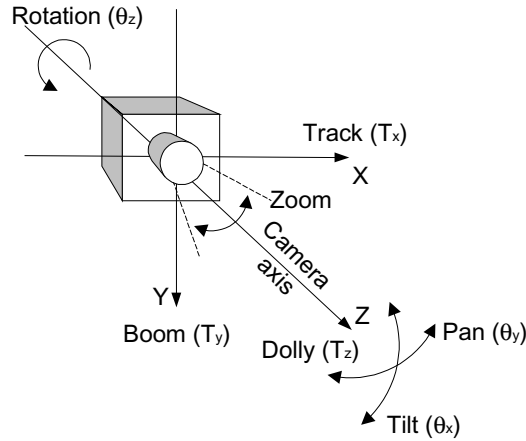


Figure 6.23 – Definition of camera motion [Correia02]

With the selected affine model, when two time instants are considered, as well as forward motion, the transparency value of a shapel with coordinates (x',y') in the most recent time instant can be computed from the shapel with coordinates (x,y) in the previous time instant by the following expression:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \\ -c_2 & c_1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_3 \\ c_4 \end{bmatrix} \quad (6.21)$$

where c_1 , c_2 , c_3 and c_4 are the affine model global motion parameters. Parameter c_1 is associated with *zoom*, parameter c_2 with *rotation* and parameters c_3 and c_4 with *pan* and *tilt*, respectively.

The target of the technique proposed here is to be able to conceal errors in the shape data of video objects and, therefore, it is important that the determined parameters for the adopted global motion model be able to accurately describe the motion of the alpha plane. Since the alpha plane is completely uniform on either side of the video object contour (opaque inside the object and transparent outside), it does not have any internal motion and, therefore, its global motion basically corresponds to the motion of the contour. Thus, to determine the global motion of the shape data, the three following steps have to be performed:

- **Extraction of the available video object contour** – The first step is to extract what is left of the video object contour from the decoded shape data;
- **Determination of the motion vectors associated with each contour point** – The second step is to determine, for each contour point determined in the previous step, the corresponding point in the previous alpha plane, which is the same as determining a motion vector for each contour point;
- **Determination of the global motion parameters** – Finally, the last step is to determine the global motion parameters by applying a linear least squares method to the pairs of coordinates obtained in the previous step and the considered global motion model.

These three steps correspond to well-defined problems for which solutions are described in the next subsections.

6.3.1.1.1 Extraction of the available video object contour

As mentioned above, to determine the motion of the shape data, the first step is to extract what is left of the video object contour from the correctly decoded shape data. Here, as shown in Figure 6.24, the shapel representation of the contour will be used even though the edge representation used in the first part of this chapter could have been used instead. This choice was made because the shapel representation requires less memory and the topological inconsistencies associated with it do not have an impact on this technique.



Figure 6.24 – Bream video object – (a) Alpha plane; (b) Shapel representation of the contour

To extract the contour from the corrupted binary alpha plane and represent it with the shapel representation, a very simple contour filter has to be implemented. For each available opaque shapel in the corrupted alpha plane, the 4 closest neighbors (considering 4-connectivity) are examined: if at least one of them is transparent, the opaque shapel is considered a contour point. With this filter, the obtained (corrupted) contour is 8-connected.

6.3.1.1.2 Determination of the motion vectors associated with each contour point

The next step is to determine, for each point of the extracted contour, the corresponding point in the previous alpha plane; this is the same as determining a motion vector for each contour point. To do this, for each contour point, a block of shapels from the current alpha plane, centered on the contour point coordinates, is considered; this shape block will be referred to in the following as the *shape context* of that contour point. Then, with the shape context, a search is carried out to find (within a given search range) a similar shape block in displaced

locations of the previous alpha plane. After the most similar shape block has been found, the motion vector associated with the considered contour point will be simply given by the difference between the position of the found block in the previous alpha plane and the position of the shape context in the current alpha plane. As for the point in the previous alpha plane corresponding to the considered contour point, it is simply the center of the found shape block.

This search for a shape block in the previous alpha plane similar to the shape context can be quite a complex task because, in many cases, several perfectly matching candidates can be found, some of them leading to disastrous results in terms of global motion parameters. Therefore, to improve the global motion estimation, the texture data (e.g., the luminance) will also be used because the alpha plane alone gives too many inconsistencies, leading to very inaccurate motion parameters. Thus, the same procedure described above is followed, but instead of using the shape context of the considered contour point, a *texture context* is used. The texture context is simply a block of pixels from the current luminance plane, centered on the contour point coordinates. This texture context is used to carry out the search in the previous luminance plane to find a similar texture block.

Here, the used texture context is a block of 16×16 pixels and the search range is 32 pixels in total (16 to each side); however, these values can easily be changed. To measure the similarity between texture blocks, the sum of absolute differences (SAD) is used, as is typically done in motion estimation algorithms. To favor the accuracy of the estimation, an exhaustive search pattern is adopted. However, a faster, although sub-optimal, algorithm such as three step hierarchical search [Bierling88] could be selected to decrease the computational complexity associated with this processing step.

To illustrate how the use of texture can improve the motion estimation results, the Large Boat object of the Coastguard sequence (CIF, 10 fps) is considered. In Figure 6.25 (a), the shape context of the contour point with coordinates (65,132) is shown. If only the alpha plane is used to determine the motion vector associated with this contour point, two ambiguous shape blocks appear within the search range as illustrated in Figure 6.25 (b), which perfectly match the shape context: one centered on coordinates (53,133) and the other one centered on coordinates (70,132). However, when the texture information of the object is used, no ambiguity occurs. In Figure 6.25 (c), the texture context of the contour point with coordinates (65,132) is shown. As can be seen in Figure 6.25 (d), of the two previously found shape blocks one of them has a texture more similar to the texture context of the considered contour point and, therefore, will be chosen as the matching point. This is shown with further detail in Figure 6.25 (e) and (f).

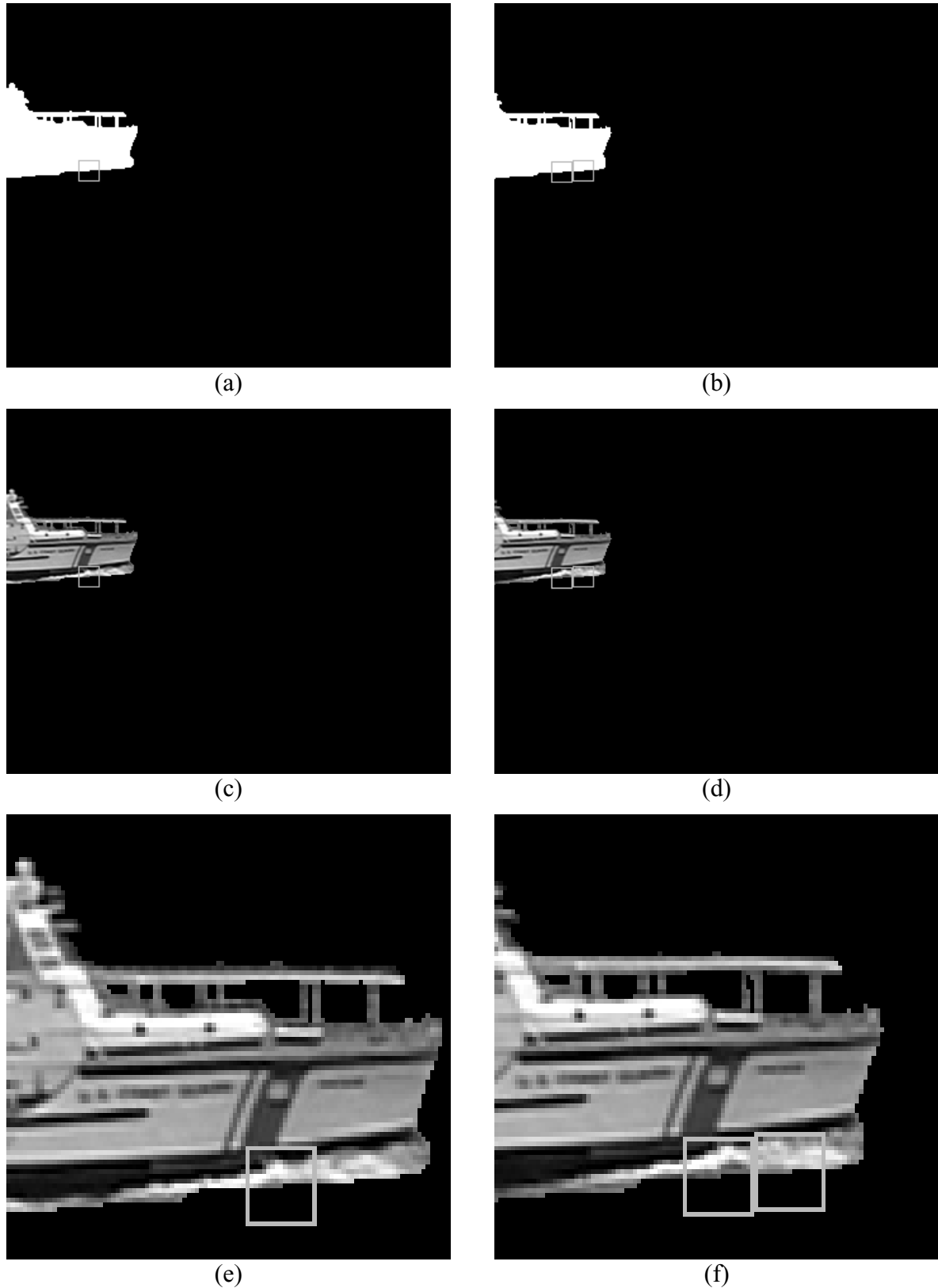


Figure 6.25 – Determination of contour point motion vectors – (a) Current alpha plane with the shape context of the contour point under consideration; (b) Previous alpha plane with two shape blocks perfectly matching the considered shape context; (c) Current texture with the texture context of the contour point under consideration; (d) Previous texture with the two texture blocks whose corresponding shape perfectly matches the considered shape context; (e) Detail of (c); (f) Detail of (d)

6. Shape Error Concealment for Object-based Video Coding

6.3.1.1.3

Determination of the global motion parameters

After corresponding points in the previous alpha plane have been found for all the points in the contour extracted from the current alpha plane, the global motion parameters are determined by finding linear least squares estimates of c_1 , c_2 , c_3 and c_4 . This is done by fitting the model shown in Equation (6.21) to the N pairs of coordinates obtained in the previous step, which can be accomplished by minimizing the following error expression:

$$E = \frac{1}{N} \sum_{i=1}^N \left[(x'_i - (c_1 x_i + c_2 y_i + c_3))^2 + (y'_i - (-c_2 x_i + c_1 y_i + c_4))^2 \right]. \quad (6.22)$$

where (x'_i, y'_i) are the coordinates of the i -th contour point extracted from the current alpha plane and (x_i, y_i) are the coordinates of the corresponding point in the previous alpha plane.

To find the minimum for the error expressed in Equation (6.22), it is simply a question of computing its derivatives with respect to the four global motion parameters (i.e., c_1 , c_2 , c_3 and c_4) and then setting them to zero, as shown in:

$$\left[\frac{\partial E}{\partial c_k} = 0 \right]_{k=1,2,3,4}. \quad (6.23)$$

This leads to the following four equations, which make up an easy to solve linear system of equations that will provide the values of c_1 , c_2 , c_3 and c_4 :

$$\frac{\partial E}{\partial c_1} = 0 \Leftrightarrow c_1 \sum_{i=1}^N (x_i^2 + y_i^2) + c_3 \sum_{i=1}^N x_i + c_4 \sum_{i=1}^N y_i = \sum_{i=1}^N (x'_i x_i + y'_i y_i) \quad (6.24)$$

$$\frac{\partial E}{\partial c_2} = 0 \Leftrightarrow c_2 \sum_{i=1}^N (x_i^2 + y_i^2) + c_3 \sum_{i=1}^N y_i - c_4 \sum_{i=1}^N x_i = \sum_{i=1}^N (x'_i y_i - y'_i x_i) \quad (6.25)$$

$$\frac{\partial E}{\partial c_3} = 0 \Leftrightarrow c_1 \sum_{i=1}^N x_i + c_2 \sum_{i=1}^N y_i + c_3 N = \sum_{i=1}^N x'_i \quad (6.26)$$

$$\frac{\partial E}{\partial c_4} = 0 \Leftrightarrow c_1 \sum_{i=1}^N y_i - c_2 \sum_{i=1}^N x_i + c_4 N = \sum_{i=1}^N y'_i \quad (6.27)$$

To help solving this linear equation system, the equations can be rewritten in the form of the following matrix equation:

$$\begin{bmatrix} \sum_{i=1}^N (x_i^2 + y_i^2) & 0 & \sum_{i=1}^N x_i & \sum_{i=1}^N y_i \\ 0 & \sum_{i=1}^N (x_i^2 + y_i^2) & \sum_{i=1}^N y_i & -\sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N y_i & N & 0 \\ \sum_{i=1}^N y_i & -\sum_{i=1}^N x_i & 0 & N \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N (x'_i x_i + y'_i y_i) \\ \sum_{i=1}^N (x'_i y_i - y'_i x_i) \\ \sum_{i=1}^N x'_i \\ \sum_{i=1}^N y'_i \end{bmatrix} \quad (6.28)$$

To solve this equation system, the singular value decomposition (SVD) method, the method of choice for solving most linear least squares problems, which is detailed in [Press92], was used⁸. This method is not new and, therefore, it will not be detailed here.

A possible problem associated with the computation of global motion parameters is the undesired bias to the results caused by either local motion (i.e., in this case, contour deformation along time) or erroneously determined matching contour point pairs. To diminish this problem, an iterative outlier removal approach is used. This way, after the first set of motion parameters c_1 , c_2 , c_3 and c_4 has been determined, the residual squared motion compensation error is computed for all the available matching point pairs. This residual error measures how well the determined motion parameters are able to transform the point with coordinates (x_i, y_i) from the past into (x'_i, y'_i) at the current time instant. This residual error can be measured using the following expression:

$$E_r = (x'_i - (c_1 x_i + c_2 y_i + c_3))^2 + (y'_i - (-c_2 x_i + c_1 y_i + c_4))^2, \quad (6.29)$$

which corresponds to the squared distance that separates the contour point with coordinates (x'_i, y'_i) extracted from the current alpha plane and the corresponding point in the previous alpha plane with coordinates (x_i, y_i) after it has been motion compensated with the determined motion parameters. All the contour point pairs, whose residual squared error is more than a standard deviation away from the mean squared error, given by Equation (6.22), will be removed from the set of contour points used to determine the global motion parameters. With the new reduced set of contour matching point pairs, new global motion parameters are computed. Then, the outlier removal procedure is repeated until the global motion parameters converge. The global motion parameters are considered to have converged when they do not change substantially from one iteration to the next (i.e., the change in the global motion parameters is so small that the corresponding changes in terms of the motion compensated results are smaller than a pixel and, hence, not visible). To avoid determining unreliable global motion parameters, this procedure is stopped if the number of contour matching point pairs drops below an empirically determined value of 15; in this case, it is considered that no global motion exists and the global motion parameters are set accordingly (i.e., $c_1=1$, $c_2=0$, $c_3=0$ and $c_4=0$), which is basically equivalent to simply copying the alpha plane from the previous time instant.

6.3.1.2 Global Motion Compensation

After the global motion parameters are known, the global motion compensation itself has to be performed. The objective of this module is to take the correctly decoded (or concealed) alpha plane from the previous time instant and global motion compensate it to the current time instant, so that it can be used to conceal the corrupted alpha plane. To do this, the decoder has to consider all the points with coordinates (x, y) in the previous alpha plane and compute their new coordinates (x', y') in the current time instant where the concealment is to be applied, by using Equation (6.21). To make the motion compensation more efficient, only the points with coordinates (x, y) that correspond to opaque shapels in the previous alpha plane have to be considered. After all the opaque shapels from the previous alpha plane have been motion compensated to the current time instant, a new alpha plane is created by choosing a matrix of shapels with exactly the same size as the corrupted alpha plane in the current time instant where concealment is to be performed and perfectly aligned with it, as

⁸ If the SVD method does not converge after 30 iterations, as suggested in [Press92], it is considered that the global motion parameters cannot be determined. In this case, the corrupted VOP is simply replaced by the one from the previous time instant.

illustrated in Figure 6.26. Strictly speaking, this new alpha plane (shown with a dashed line in Figure 6.26 (c)) does not correspond to the global motion compensated version of the previous alpha plane (which is shown with a solid line), but rather to a new alpha plane where only the opaque shapes have been global motion compensated from the previous alpha plane. However, in the remainder of the chapter, this new alpha plane is the one that will be used and it will be referred to as the global motion compensated previous alpha plane. Of course, motion compensation can be applied to the shape data as well as to the texture data.

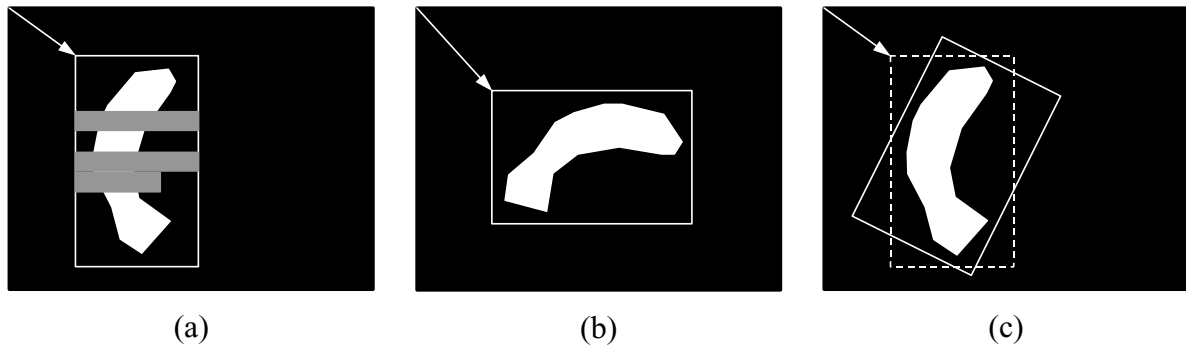


Figure 6.26 – Creation of the global motion compensated previous alpha plane - (a) Corrupted alpha plane at current time instant; (b) Previous alpha plane to be global motion compensated; (c) Actual global motion compensated previous alpha plane (solid) and the new alpha plane created with only the global motion compensated opaque shapes (dashed)

To illustrate the global motion compensation operation, several examples are given in Figure 6.27. Here, the shape in Figure 6.27 (a) is used as the departing alpha plane, from which all other shapes will be derived. The three elementary motion types that can be described by the used motion model, *pan/tilt*, *zoom* and *rotation*, are illustrated in Figure 6.27 (b), Figure 6.27 (c) and Figure 6.27 (d), respectively. Finally, Figure 6.27 (e) shows a case where all three types of elementary motion have been combined.

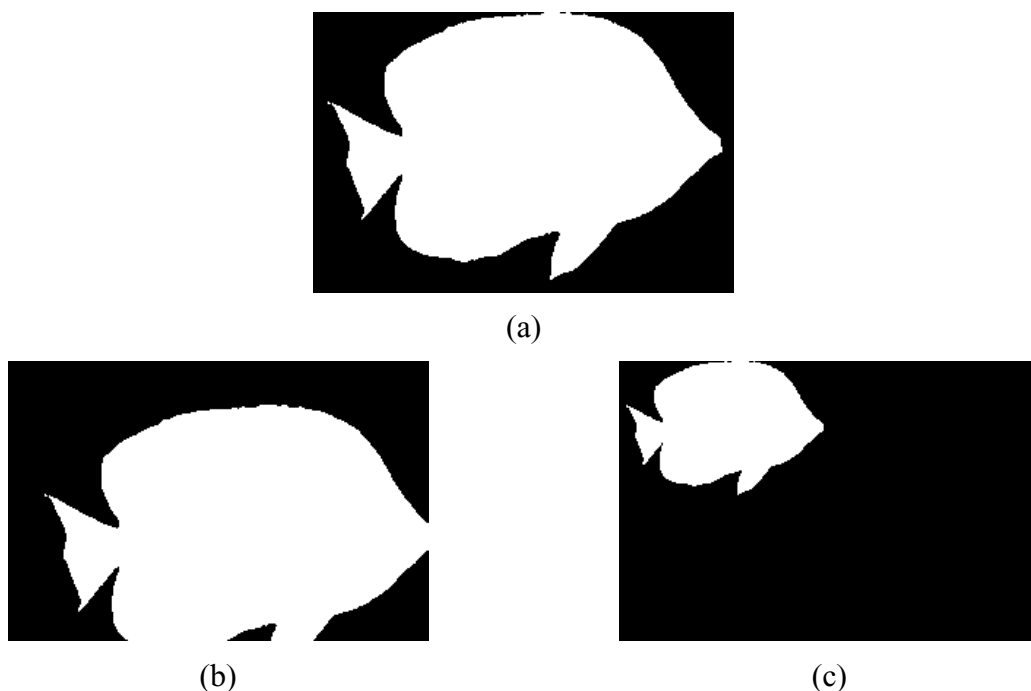




Figure 6.27 – Examples of global motion compensated Bream alpha planes with different parameter values: (a) Original alpha plane without motion compensation; (b) Pan and tilt: $c_1=1.0$, $c_2=0.0$, $c_3=15.0$ and $c_4=30.0$; (c) Zoom: $c_1=0.5$, $c_2=0.0$, $c_3=0.0$ and $c_4=0.0$; (d) Rotation: $c_1=1.0$, $c_2=0.5$, $c_3=0.0$ and $c_4=0.0$; (e) Combination of zoom, rotation, pan and tilt: $c_1=0.25$, $c_2=0.5$, $c_3=50.0$ and $c_4=150.0$

6.3.1.3 Replacement of Corrupted Alpha Plane Blocks

After the previous alpha plane has been (global) motion compensated, the concealment itself can start. As mentioned at the beginning of Section 6.3.1, the first step is to assume that the global motion model with the determined parameters correctly describes the motion of the shape data; afterwards, if necessary, the shape concealment will be refined by applying a local motion correction. This way, in the corrupted alpha plane, all the blocks that were considered corrupted are first replaced with the corresponding alpha plane blocks from the global motion compensated previous alpha plane. This is a very easy task since the corrupted alpha plane and the global motion compensated previous alpha plane have exactly the same size and are perfectly aligned. The same can be done for the texture data.

To better understand this procedure, Figure 6.28 should be considered. In Figure 6.28 (a), the corrupted alpha plane that is going to be concealed is shown. The gray areas correspond to the corrupted alpha blocks that have to be concealed. In Figure 6.28 (b), the previous global motion compensated alpha plane is shown, where the shaded areas are the alpha blocks that correspond (i.e., have the same position) to the corrupted alpha blocks in the corrupted alpha plane. Therefore, these blocks simply have to be copied to the corrupted alpha plane. The result is shown in Figure 6.28 (c), where the concealed blocks are shaded.

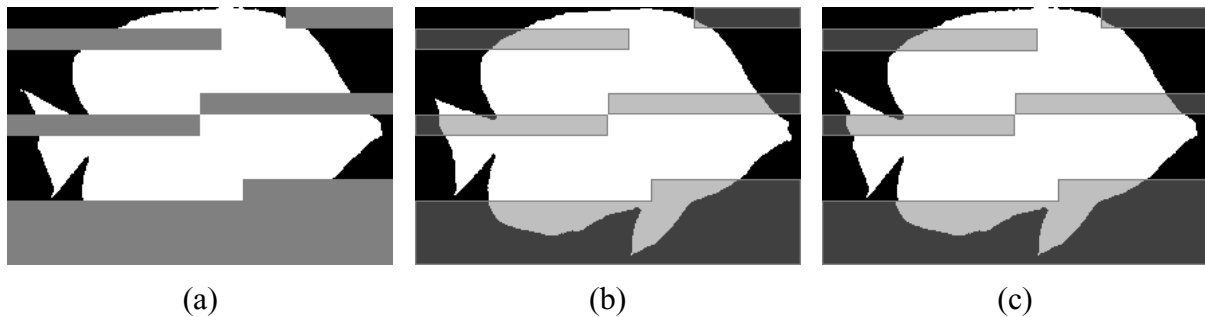


Figure 6.28 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane

6.3.1.4 Local Motion Refinement

The case illustrated in the previous section represents a simple situation where practically no local motion exists and, therefore, nothing else would have to be done to improve the shape concealment results. However, there are cases, such as the one illustrated back in Figure 6.22, where local motion exists and the achieved shape concealed quality with the global compensation method alone is not acceptable. Thus, to deal with these more complex cases, a concealment refinement procedure has to be used, which includes the compensation of the local motion. However, before trying to refine the concealment, it is important to distinguish two types of concealed blocks: blocks that have at least one correctly (shape) decoded surrounding block and blocks that only have surrounding blocks that have been (shape) concealed themselves. Since the former have much more reliable surrounding shape data (because it has been correctly decoded), the refinement should start with those. The concealment refinement is applied to the blocks that do not have any correctly decoded neighbors only afterwards. Thus, the refinement procedure consists of the following two steps:

- **Local motion refinement of shape blocks with correct neighbors** – This first step consists of the local motion refinement of the shape concealment for blocks that have at least one correctly decoded surrounding block;
- **Local motion refinement of shape blocks with no uncorrupted neighbors** – The second step consists in the local motion refinement of the shape concealment for blocks that have no correctly decoded surrounding blocks, only blocks that have already been concealed.

These two steps correspond to different problems for which solutions are proposed in the next subsections.

6.3.1.4.1 Concealment refinement of shape blocks with correct neighbors

As explained above, the first concealed shape blocks to be refined are the ones that have at least one correctly decoded neighboring shape block among their eight closest neighbors (see Figure 6.29). This is understandable since these are the blocks that have the most reliable surrounding shape data on which the local motion refinement can be based. This way, the shape blocks in the concealed alpha plane are scanned from top to bottom, left to right. For each concealed shape block that has at least one correctly decoded neighboring block, a set of successive operations has to be performed in order to obtain the refinement, as follows:

1	2	3
4		5
6	7	8

Figure 6.29 – The eight closest neighbors surrounding the concealed shape block shown in dark gray

1. **Determination of the need for local motion refinement** – The first step has to determine if the considered shape block has been acceptably concealed with the global motion compensation alone or not. This is done by inspecting the surrounding correctly decoded shape blocks and comparing them with the shape blocks that would have been obtained if global motion compensation concealment had also been applied to them. If the number of different shapels exceeds 90 for all the surrounding correctly decoded shape blocks (approximately 4.4% of the total number of shapels in the eight surrounding blocks) or 30 for any individual block (approximately 11.7% of the number of shapels in a single shape block), the shape block at hand is considered not to have been adequately concealed. These thresholds have been determined by exhaustive experimentation and were chosen because they lead to the best results. If the shape block at hand is considered as having been adequately concealed, no further processing is needed for it.
2. **Determination of local motion vectors for the surrounding shape blocks** – If the considered shape block is considered not adequately concealed, the local motion concealment refinement has to be applied. For this, a motion vector is determined for each one of the non-transparent correctly decoded shape blocks surrounding the shape block in question. This is done by applying a typical block-matching motion estimation algorithm to the current luminance plane and the previous luminance plane. The search range used to determine the motion vectors is the same that was used above for the determination of the contour points motion vectors (i.e., 32 pixels in total, 16 on each side).
3. **Determination of an average local motion vector for the shape block being refined** – Based on the motion vectors determined for all the non-transparent correctly decoded neighboring shape blocks in step 2, an average motion vector can be determined for the shape block whose concealment is being refined. If some of the neighboring shape blocks were not correctly decoded but have already been (concealed and) local motion refined and, therefore, already have local motion vectors associated to them, they are also included in the computations of the average motion vector.
4. **Refinement by local motion compensation** – After the average motion vector has been determined in step 3, local motion compensated concealment is tried for the shape block at hand with several motion vectors (i.e., replacing the corrupted shape block with the shape block from the previous alpha plane indicated by the considered motion vector). The tested motion vectors are:
 - The average motion vector determined above in step 3;
 - The motion vector from the top shape block;
 - The motion vector from the bottom shape block;
 - The motion vector from the left shape block;
 - The motion vector from the right shape block.

Of course, the four last motion vectors are only tried if the corresponding shape blocks exist and were either correctly decoded or already (concealed and) refined. Of the tested motion vectors, the one that will be used to perform the final concealment is the one that minimizes a shape border continuity metric for the concealed shape block. This metric is computed by considering the shapels in the four borders of the concealed shape block being refined and the shapels across these borders in the

6. Shape Error Concealment for Object-based Video Coding

concealed alpha plane. For an $N \times N$ block (here N equals 16), this shape continuity metric (SCM) is defined as:

$$SCM = \sum_{i=1}^N |t_i - \hat{t}_i| + \sum_{i=1}^N |b_i - \hat{b}_i| + \sum_{i=1}^N |l_i - \hat{l}_i| + \sum_{i=1}^N |r_i - \hat{r}_i| \quad (6.30)$$

where t_i , b_i , l_i , r_i correspond to shapels in the top border, the bottom border, the left border and the right border of the shape block being refined, respectively. As for \hat{t}_i , \hat{b}_i , \hat{l}_i , \hat{r}_i , they correspond to the shapels outside the shape block being refined and across the border from t_i , b_i , l_i , r_i , respectively. In Figure 6.30, this metric is illustrated for an 8×8 block. The same motion vector can also be used to refine the concealed texture data.

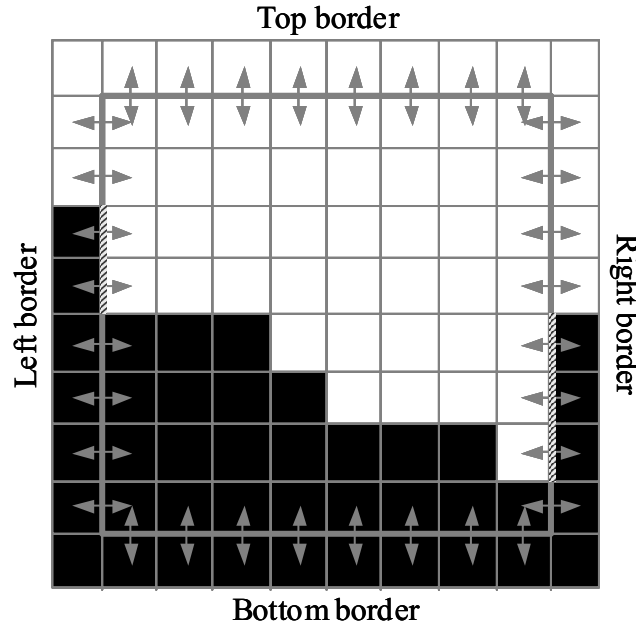


Figure 6.30 – Example of continuity metric computation; the borders across which there are different shapels are shown with a dashed line (this happens for two shapels in the left border and for three pixels in the right border)

After having followed this procedure for all the corrupted shape blocks that have at least one correctly decoded neighboring shape block, the first step of the concealment refinement is complete. Before proceeding to the next step of the refinement, Figure 6.31 should be considered since it illustrates well the advantages of this first step. In Figure 6.31 (a), the corrupted alpha plane that is going to be concealed is shown, where all the corrupted shape blocks (shown in gray) have at least one correctly decoded neighbor. In Figure 6.31 (b), the previous global motion compensated alpha plane is shown. To help the reader, in Figure 6.31 (b), as well as in the remainder of Figure 6.31, the position of the corrupted shape blocks will be shaded. As for Figure 6.31 (c), it shows the concealed alpha plane before the local motion refinement has been applied, while Figure 6.31 (d) shows it after it has been applied. As can be seen from Figure 6.31 (c), when the corrupted shape blocks are concealed by simply copying co-located shape blocks from the previous (global) motion compensated alpha plane, only the areas of the object where hardly any local motion exists are adequately dealt with. However, for the areas where local motion exists, such as the area of the racket, the concealment results are very poor, especially when compared to what was achieved in Figure

6.31 (d). To compare the concealed alpha planes with the original uncorrupted alpha plane, the reader can refer back to Figure 6.22 (a).

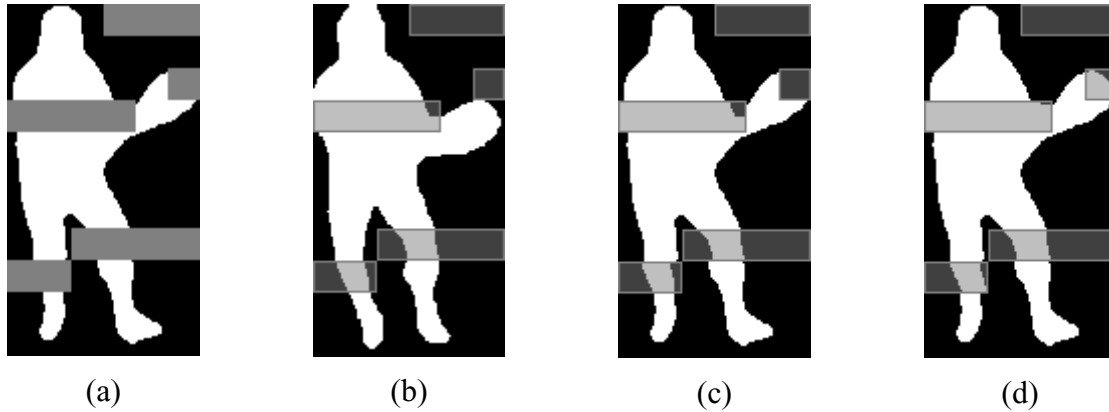


Figure 6.31 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane with global motion compensation only; (d) Concealed alpha plane after shape blocks with correctly decoded neighbors have been local motion refined

6.3.1.4.2

Concealment refinement of shape blocks with no uncorrupted neighbors

Before describing the refinement procedure for the shape blocks that have no uncorrupted neighbors, it should be noted that the set of operations described in this section is only performed if, at least, one shape block has been refined in the previous step of the concealment refinement, described in Section 6.3.1.4.1. This happens because here to refine the concealment of a shape block with no uncorrupted neighbors, the existence of, at least, one previously refined neighbor is necessary. Therefore, if no shape blocks have been refined in the previous section, this means that it will also not be possible/necessary to refine the shape blocks with no uncorrupted neighbors here (which means that these blocks will be obtained only by means of global motion compensation).

After the refinement step described in the previous section, there are still some concealed shape blocks that have not been locally refined yet (i.e., those that do not have any uncorrupted decoded neighbors). Thus, to refine the concealment of the remaining shape blocks, a set of successive operations has to be performed on each of these shape blocks that do not have any correctly decoded neighboring blocks. However, in order to guarantee that all the necessary shape blocks are refined, this set of operations has to be performed first from top to bottom, left to right, and then from bottom to top, right to left. To better understand why, suppose that the first three rows of blocks in an alpha plane have been corrupted. After the shape blocks in the third row, which are the only ones that have correctly decoded neighbors, have been refined as described in the previous section, the first two rows of corrupted shape blocks still remain to be refined. At this point, only the second row of corrupted shape blocks has refined neighbors. Since the concealment refinement proposed in this section depends on the existence of refined neighbors, only the second row will be refined when performing the described set of operations from top to bottom, left to right. As for the first row of corrupted shape blocks it will only have refined neighbors after the second row has been refined and, therefore, it will only be refined when the processing is being done from bottom to top, right to left. This way, the set of necessary successive operations that has to be performed on the shape blocks that do not have any correctly decoded neighboring

blocks, first from top to bottom, left to right, and then from bottom to top, right to left, corresponds to the following succession of steps:

1. **Determination of the existence of locally refined neighbors** – The first step is to determine if any of the surrounding shape blocks have been locally refined or not. If not, nothing is done at this time and no further processing is needed for this shape block.
2. **Determination of an average local motion vector for the shape block being refined** – If, on the other hand, some of the surrounding shape blocks have already been locally refined, their previously determined (i.e., in the previous section) motion vectors should be considered and used to compute an average motion vector.
3. **Refinement by local motion compensation** – After the average motion vector has been determined in step 2 above, motion compensated concealment is tried for the shape block at hand with several motion vectors (i.e., replacing the corrupted shape block with the shape block from the previous alpha plane indicated by the considered motion vector). The tested motion vectors are:
 - The average motion vector determined above in step 2;
 - The motion vector from the top shape block;
 - The motion vector from the bottom shape block;
 - The motion vector from the left shape block;
 - The motion vector from the right shape block.

Of course, the four last motion vectors are only tried if the corresponding shape blocks exist and were already (concealed and) local motion refined. Of the tested motion vectors, the one that will be used to perform the final concealment is the one that minimizes the shape border continuity metric, defined in Equation (6.30), for the concealed shape block. The same motion vector can also be used to refine the concealed texture data.

After having followed this procedure in the two necessary directions for all the shape blocks that do not have any correctly decoded neighboring shape blocks, the concealment refinement related to the local motion correction is complete. Before proceeding to the next and final step in the concealment process, which is simply a filter to eliminate some small artifacts that may have been created, Figure 6.32 should be considered. Similarly to Figure 6.31, Figure 6.32 (a) shows the corrupted alpha plane that is going to be concealed, where the corrupted shape blocks are shown in gray. However, in this case, not all the corrupted shape blocks have at least one correctly decoded neighbor. There is one shape block with no correctly decoded neighbors, which has been indicated in lighter gray. As in Figure 6.31, Figure 6.32 (b) shows the previous global motion compensated alpha plane, where the position of the corrupted shape blocks is shaded. As for the remainder of Figure 6.32, Figure 6.32 (c) shows the concealed alpha plane before any local motion refinement has been done and Figure 6.32 (d) and Figure 6.32 (e) show the concealed alpha plane after the first and the second steps of the local refinement, respectively. As can be seen, the concealment results are gradually improved, approaching the uncorrupted original. To compare the concealed alpha planes with the original uncorrupted alpha plane, the reader can refer back to Figure 6.22 (a).

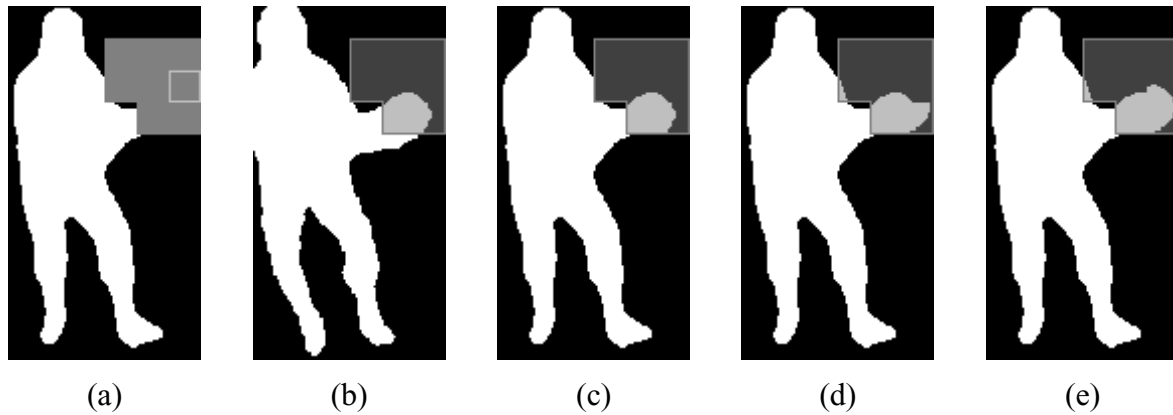


Figure 6.32 – Replacement of corrupted alpha plane blocks: (a) Corrupted alpha plane; (b) Previous global motion compensated alpha plane; (c) Concealed alpha plane with global motion compensation only; (d) Concealed alpha plane after the shape blocks with correctly decoded neighbors have been refined; (e) Concealed alpha plane after all the shape blocks have been refined

6.3.1.5 Post-processing for Small Regions Removal

Sometimes, after the motion concealment operations are over (i.e., global motion compensation concealment in addition to the local motion refinement), small opaque or transparent regions appear in the concealed alpha plane. Although these regions are typically very small and, therefore, have a very small influence on the objective quality metrics, their subjective impact can be quite negative. Thus, it is important that these small regions be dealt with. The following two types of regions have been identified:

- **Small opaque or transparent regions inside the concealed area** – These small regions, located inside the concealed area, are a direct consequence of the concealment operations themselves, in the sense that the shapels of these regions have been “created” by the concealment. After all, during concealment, it is perfectly possible for a group of shapels to be erroneously concealed in such a way as to create an isolated region inside the concealed area. By removing this artificially looking region, the objective quality will be slightly improved, and the impact on the subjective quality will be much better. This is clearly illustrated in Figure 6.33. Unfortunately, in some other cases, this region may have been simply copied from the past and is not supposed to be eliminated.
- **Small opaque or transparent regions outside but in contact with the concealed area** – These other small regions, located outside but in contact with the concealed area, are an indirect consequence of the concealment operations, in the sense that the shapels in these regions have not been “created” by the concealment. After all, since these shapels are outside the concealed areas, they correspond to correctly decoded shapels that have not even been concealed. In fact, these regions are due to the way nearby shapels inside the concealed area have been concealed. Nevertheless, they can create a very negative subjective impact on the user and, therefore, should be eliminated. Of course, by eliminating them, the objective quality of the alpha plane will be slightly worsened. This is clearly illustrated in Figure 6.34.

To eliminate these regions, a rather conservative approach was followed due to the risk associated with this operation; after all, there is a risk of actually removing important

information. Therefore, only regions inside the concealed area or outside but in contact with it, smaller than 25 shapels (approximately 9.8% of the size of a 16×16 shape block), were eliminated. Of course, it is still always possible to turn off this filter if the user should so wish.

To illustrate the elimination of the small regions described above, the following two examples can be considered. In Figure 6.33, the elimination of a small region inside the concealed area is shown, while in Figure 6.34 the eliminated region is outside but in contact with the concealed area. In these examples, the concealed shape blocks have not been shaded, as was done in previous examples, because it would make the small regions difficult to see. The impact of this elimination is very small in terms of objective quality, but the improvement in terms of subjective impact is quite large.

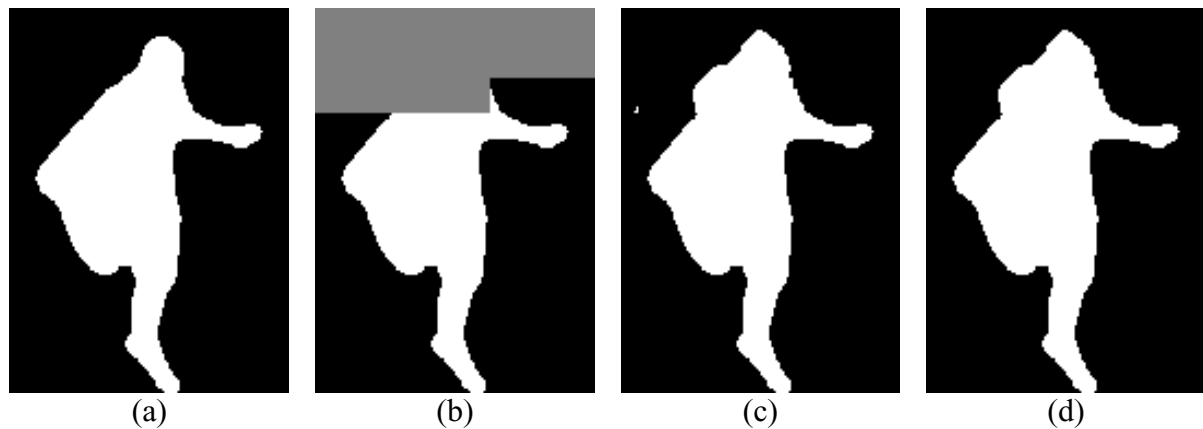


Figure 6.33 – Elimination of small opaque region inside the concealed area: (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Concealed alpha plane before small regions elimination; (d) Concealed alpha plane after small regions elimination

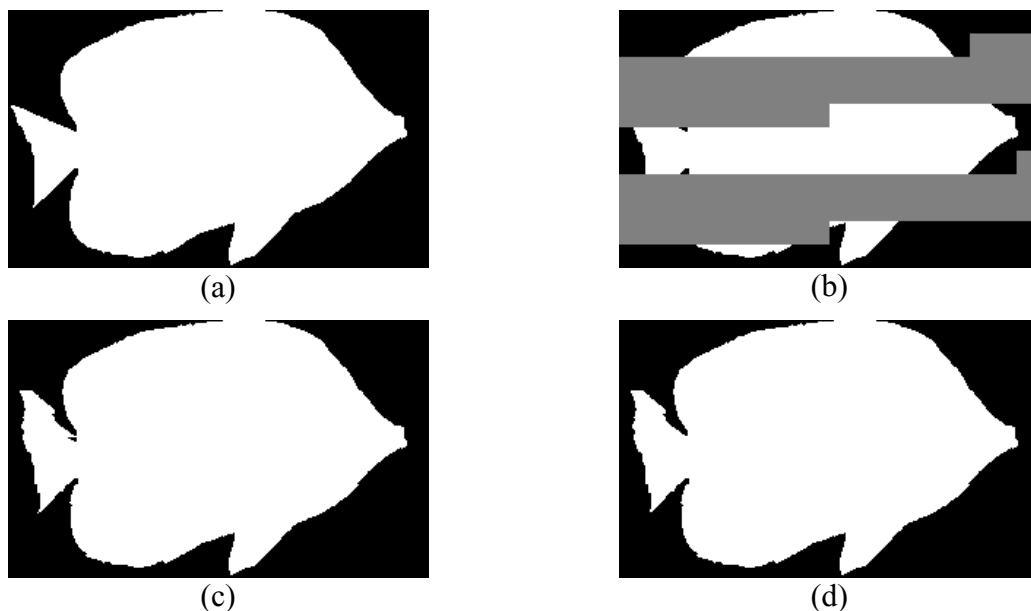


Figure 6.34 – Elimination of small transparent region outside but in contact with the concealed area: (a) Original uncorrupted alpha plane; (b) Corrupted alpha plane; (c) Concealed alpha plane before small regions elimination; (d) Concealed alpha plane after small regions elimination

6.3.2 Performance Evaluation

In order to evaluate the proposed temporal shape error concealment technique, several MPEG-4 bitstreams have been tested, each bitstream containing one video object, encoded according to the MPEG-4 Core Visual Object Type, at a given bit rate. Here, as opposed to what happened for the spatial concealment technique, the bit rate used is important because, although it does not influence the quality of the shape (i.e., lossless shape coding was used), it influences the quality of the texture data, which is used for estimating the global motion parameters. Therefore, the used MPEG-4 bitstreams have to be encoded at a bit rate that leads to an acceptable decoded texture quality, thus leading to more accurate global motion parameter values and improved shape concealment. Notice that MPEG-4 streams are just used as the most representative streams at hand; the proposed technique just requires a block-based coding approach and does not depend in any way on MPEG-4 video coding.

As for the remaining test conditions used for the performance evaluation of the proposed temporal shape error concealment technique, they are detailed in the following, grouped into several categories:

- **Test Video Object Sequences** – The tested video object sequences are the CIF versions of the Akiyo, Bream and Stefan video objects; the lowest acceptable frame rate for these sequences was used because it corresponds to the most critical situation in terms of temporal error concealment (i.e., 10 fps for the Akiyo and Bream sequences and 15 fps for the Stefan sequence). At these frame rates, an acceptable texture quality can be obtained by encoding Akiyo at 64 kbps, Bream at 128 kbps and Stefan at 128 kbps (for the Core Visual Object Type). These encoding parameters used for the various sequences are summarized in Table 6.2.
- **Error Resilience Tools** – In terms of MPEG-4 error resilience tools, the same tools that were used for the spatial shape concealment technique, proposed in Section 6.2, were again used at the encoder (i.e., resynchronization markers and data partitioning with reversible variable length codes), thus making the tested bitstreams more resilient to errors. As for the size of the video packet in bits, this is usually chosen (e.g., in MPEG) so that there are approximately four VPs in each VOP, as done in Chapter 5. However, this criterion was typically used for QCIF sequences, which is not the case here. For higher resolution sequences, as shown in the MPEG error resilience video verification tests of July 1998 [N2061][N2165][N2604], a larger number of video packets should be used and, therefore, eight video packets per VOP will be used here. In terms of bits per video packet, this will of course vary with the used frame and bit rates. This way, for the Akiyo, Bream and Stefan video objects, with the mentioned frame and bit rates, this will correspond to video packet size thresholds of 800 bits, 1600 bits and 1060 bits, respectively. The video packet sizes have also been included in Table 6.2. Additionally, to guarantee that the shape quality does not degrade too much, as well as the texture quality, which is also important in this context, an intra refreshment scheme was used at the encoder. The approach used for the shape intra refreshment is the one that was referred to, in Chapter 5, as the reference shape intra refreshment technique⁹. This approach basically consists in periodically intra refreshing (i.e., with a fixed rate) the shape data. For the results presented in this section, two different shape refreshment periods were tried: 1 and 3.

⁹ The adaptive shape intra refreshment technique proposed in Chapter 5 is not used here because it can only be used to distribute the available intra refreshment resources among several video objects in a scene. Here, only one object is considered at a time.

The approach used for the texture intra refreshment is the one that was referred to, in Chapter 5, as the reference texture intra refreshment technique¹⁰. This approach basically consists in assigning a number of texture macroblocks to be refreshed at each time instant, which is then divided among the various video objects proportionally to their size; within each video object, the refreshment is done in a cyclic fashion. Here, it will be considered, as in Chapter 5, that one tenth of the scene should be refreshed at each time instant. For instance, for the CIF format, this corresponds to 48 macroblocks at each time instant for the whole scene. This number is obtained by taking 10% of the 396 macroblocks of the CIF format (i.e., 40) and adding an additional 20% to take into account any object overlapping (see Chapter 5). Since, in this chapter, only individual video objects are considered, the number of texture macroblocks to be refreshed at each time instant for a given object is determined by taking the proportion of 48 associated to the percentage of the scene occupied by the object.

- **Error Simulation** – As in Section 6.2.2, instead of adding errors directly to the bitstreams, the decoder simply ignores the video packets randomly with a given packet loss rate, thus allowing to evaluate the performance of the concealment technique independently of the decoder error detection capabilities. Then, the proposed temporal shape error concealment technique has been applied to the corrupted decoded alpha planes to recover the missing shape data. In terms of video packet loss rates, for these tests, the same values that were used back in Section 6.2.2 are used again (i.e., 1%, 5%, 10% and 20%), which correspond to increasing amounts of channel errors; these error rates were chosen for the reasons already explained in that section. For each one of these loss rates, each one of the tested video objects has been decoded 50 times (i.e., corresponding to 50 different error patterns or runs).
- **Quality Evaluation Tools** – In order to evaluate the performance of the proposed technique in terms of shape recovery, the Dn metric, described in Section 6.2.2, will be used. As before, in addition to the numeric shape quality values associated to the metric above, the concealment results will be illustrated for several reconstructed alpha planes, which should allow the reader to subjectively evaluate the performance provided. When visual results are shown, these will include the original, the corrupted and the concealed alpha planes (in the bounding box of the video object). Since the proposed technique can also be applied to the texture of the video object, whose decoded texture quality is here very important (for the estimation of the global motion parameters), numeric texture quality values will also be presented using the usual object-based $PSNR$ metric, already presented in Chapter 5, as well as visual results. The visual results will include the original, the corrupted and the concealed video object texture (in its bounding box).

¹⁰ The same comment that was made regarding the use here of the adaptive shape intra refreshment technique also applies to the adaptive texture intra coding refreshment proposed in Chapter 5, which can also not be used here.

Table 6.2 – Summary of encoding parameters

	Akiyo	Bream	Stefan
Spatial Resolution	CIF	CIF	CIF
Frame Rate (fps)	10	10	15
Bit Rate (kbps)	64	128	128
VP Size (bits)	800	1600	1060

In the following numerical results, three different types of Dn and $PSNR$ values are shown for the tested video object sequences, and for four different video packet loss rates: Dn_{low} , Dn_{avg} , Dn_{high} and $PSNR_{low}$, $PSNR_{avg}$, $PSNR_{high}$. While Dn_{low} and Dn_{high} values correspond, respectively, to the average Dn value associated with the best and the worst runs in terms of shape quality, Dn_{avg} corresponds to the mean of the average Dn values associated with the 50 different runs for each test case. Similarly, $PSNR_{low}$ and $PSNR_{high}$ values correspond, respectively, to the average $PSNR$ value associated with the worst and the best runs in terms of texture quality and $PSNR_{avg}$ corresponds to the mean of the average $PSNR$ values associated with the 50 different runs for each test case. As for the Dn and $PSNR$ values associated with a given run, they are simply the temporal average of the Dn and $PSNR$ values, respectively, for all the VOPs in that video object sequence, defined as:

$$Dn = \frac{1}{N_{VOP}} \sum_{i=0}^{N_{VOP}-1} Dn_i \quad (6.31)$$

$$PSNR = \frac{1}{N_{VOP}} \sum_{i=0}^{N_{VOP}-1} PSNR_i \quad (6.32)$$

where N_{VOP} is the total number of VOPs in the test sequence and i is the index corresponding to the VOP number. If, while decoding a bitstream, the decoder loses a complete VOP, the lost VOP is replaced with the previously decoded VOP, which is then used for the Dn and $PSNR$ computations, as was done in Section 6.2.2.

6.3.2.1 Results with a Shape Intra Refreshment Period of 1

In the first set of results, the shape data in all of the test sequences will be refreshed at each time instant. With this alpha plane refreshment rate, the error-free decoded texture qualities presented in Table 6.3 are obtained (for the used bit rates). The obtained error-free Dn is not included in the table since it is obviously 0.00% because the used shape coding is lossless.

Table 6.3 – Error-free PSNR obtained for the chosen encoding parameters

	Akiyo	Bream	Stefan
Error-free PSNR [dB]	33.85	30.01	30.46

As can be seen in Table 6.4, even for relatively high video packet loss rates, the Dn values remain relatively low. In most cases, the Dn values are so low that they correspond to hardly noticeable artifacts (i.e., Dn_{avg} values around 1% and below). This is the case for the Akiyo video object with packet loss rates of 1%, 5%, 10% and 20%, for the Bream video object with

6. Shape Error Concealment for Object-based Video Coding

packet loss rates of 1%, 5% and 10%, and for the Stefan video object with packet loss rates of 1%. For higher packet loss rates and less simple shapes, such as 20% for Bream and 5% for Stefan, the artifacts start to become more visible (i.e., Dn_{avg} values higher than 1% but still below 3%). And, finally, for the Stefan video object with a packet loss rate of 10%, the artifacts become clearly visible but still very acceptable (i.e., the Dn_{avg} value is 3.42%). For a packet loss rate of 20%, however, the Stefan artifacts become quite annoying (i.e., the Dn_{avg} value is 7.45%). These results clearly reflect the intrinsic shape concealment difficulty of the tested video objects, by ranking the Akiyo object as the easiest to conceal, followed by Bream and, finally, Stefan. This can be explained by considering the type of motion associated with the tested objects: the Akiyo object hardly moves, the Bream object moves quite a lot but suffers almost no deformation along time and, finally, the Stefan object moves a lot but also suffers a lot of deformation along time. In terms of texture quality, whose results are included in Table 6.5, the same comments that were made for the Dn values can be also made for the $PSNR$ values.

Table 6.4 – Dn values for the tested video object sequences

Video packet loss rate	Dn [%] ($Dn_{low}/Dn_{avg}/Dn_{high}$)								
	Akiyo			Bream			Stefan		
1%	0.00	0.02	0.08	0.01	0.07	0.23	0.09	0.32	0.57
5%	0.03	0.08	0.23	0.19	0.34	0.61	0.92	1.67	2.55
10%	0.08	0.15	0.31	0.49	0.70	1.01	2.25	3.42	5.09
20%	0.19	0.29	0.41	1.16	1.49	2.02	6.17	7.45	9.79

Table 6.5 – $PSNR$ values for the tested video object sequences

Video packet loss rate	$PSNR$ [dB] ($PSNR_{low}/PSNR_{avg}/PSNR_{high}$)								
	Akiyo			Bream			Stefan		
1%	32.04	33.16	33.72	28.11	29.04	29.83	27.77	28.99	29.89
5%	29.64	30.74	31.57	25.04	26.48	27.51	23.82	25.05	26.20
10%	27.43	28.73	29.90	23.43	24.54	25.69	21.35	22.50	23.66
20%	25.26	26.12	27.55	21.11	22.00	22.93	18.94	19.59	20.40

In order to illustrate the results in Table 6.4 some corrupted and concealed alpha planes and textures have been chosen. To start this illustration, the Akiyo video object, which is the easiest to conceal among the tested ones, was chosen; the shape and texture of a given decoded VOP (corresponding to VOP 3 in the original 300 VOP sequence) is shown in Figure 6.35 (a) and (b), respectively. In the remainder of Figure 6.35, three different corrupted versions of the alpha plane in Figure 6.35 (a) and the texture in Figure 6.35 (b) are shown, in addition to the corresponding concealed alpha planes and textures. These three corrupted versions correspond to three of the 50 different error patterns used above for a video packet loss rate of 20%, which corresponds to the most critical error condition tested.

As can be seen in Figure 6.35, the obtained results are very good in the sense that the existing shape artifacts are unnoticeable. The only exception to this is the small hole on the left hand side of the hair, which has been eliminated in Figure 6.35 (c) and (d) by the small regions elimination filter; this would be avoided by simply turning off the filter, as shown in Figure 6.36. These concealment results were expected since the Akiyo object hardly moves and suffers no deformation along time. As for the Dn values corresponding to the shown concealed alpha planes, they are very low (typically around or below 1%): 0.77% for pattern 1, 0.18% for pattern 2, and 1.15% for pattern 3. In terms of texture, the obtained results are also very good and the only noticeable artifact is also the filled hole in the hair (which has been filled by linearly interpolating, for each pixel, the texture values of the four closest neighbors above, below, to the left and to the right). However, in the texture this artifact is even harder to see because the hair around it is very dark. As for the corresponding $PSNR$ values, they are 28.24 dB for pattern 1, 32.85 dB for pattern 2 and 28.67 dB for pattern 3. The uncorrupted texture has a $PSNR$ value of 35.24 dB. The large drop in the $PSNR$ values is explained by the fact that large areas of the object are being concealed with global motion compensation. This means that the concealed texture can be slightly displaced (e.g., by one pixel) with respect to the original texture. This displacement goes unnoticed to the human user, leading to a very good subjective impact, but it can have a very large effect on the $PSNR$ values, which are computed based on the differences between co-located pixels.



(a)



(b)

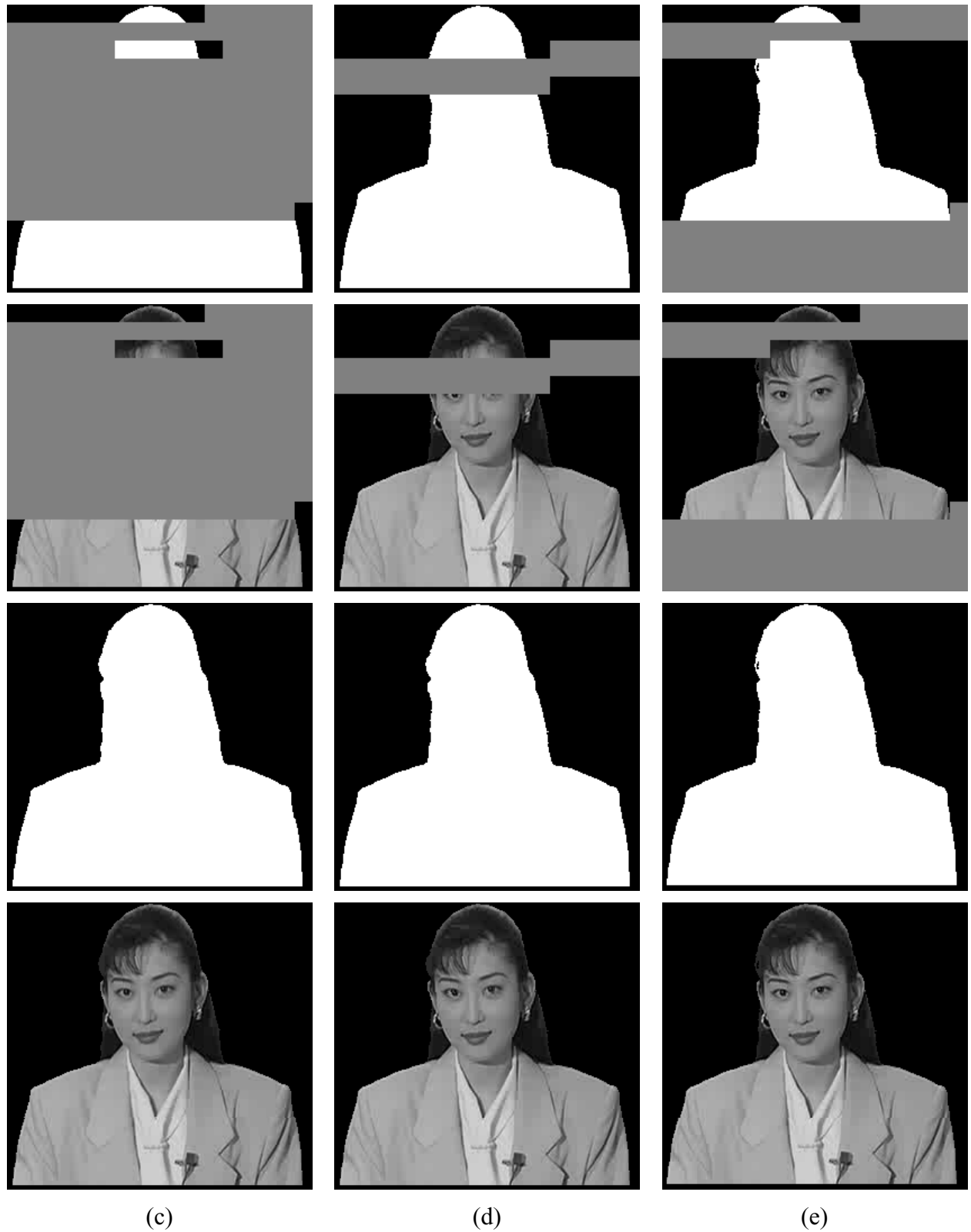


Figure 6.35 – Examples of corrupted and corresponding concealed alpha planes and textures for the Akiyo video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3

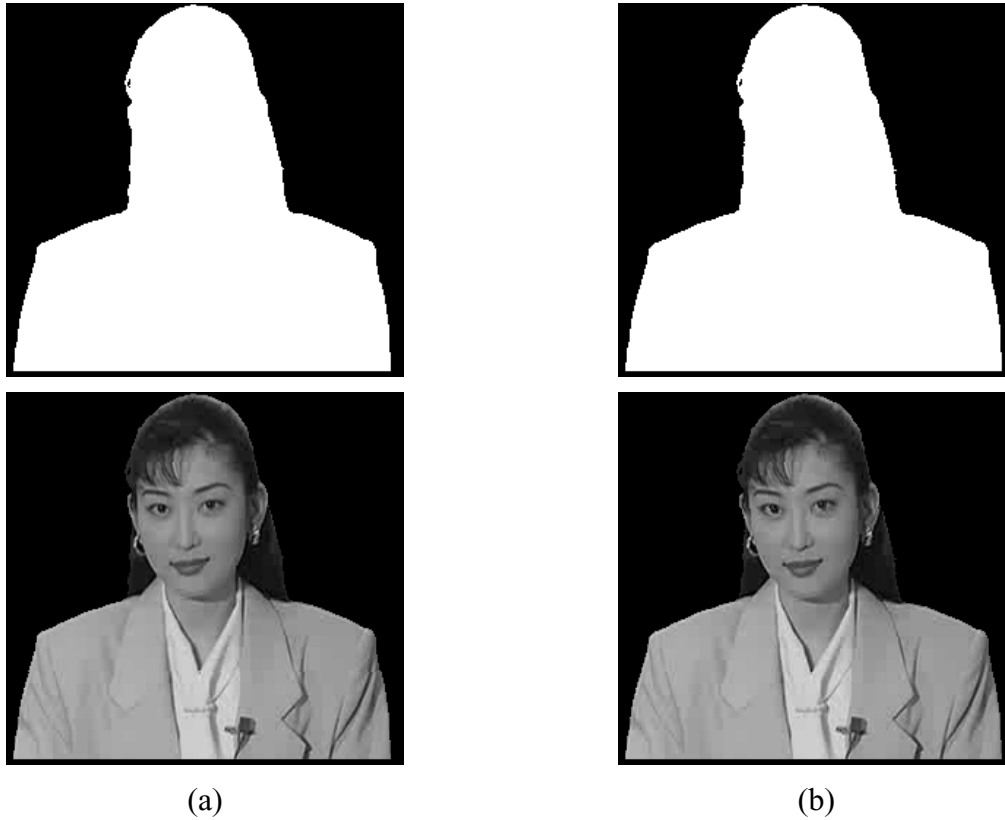


Figure 6.36 – Concealed alpha planes and textures for the Akiyo video object without the small region elimination filter (video packet loss rate of 20%) – (a) Error pattern 1; (b) Error pattern 2

The next video object chosen to illustrate the performance of the proposed error concealment technique is the Bream video object; the shape and texture of a given decoded VOP (corresponding to VOP 3 in the original 300 VOP sequence) is shown in Figure 6.37 (a) and (b). Similarly to what was done for the Akiyo video object, in Figure 6.37, three different corrupted versions of the alpha plane in Figure 6.37 (a) and the texture in Figure 6.37 (b) are shown, in addition to the corresponding concealed alpha planes and textures. These three versions also correspond to three of the 50 different error patterns used above for a video packet loss rate of 20%. As can be seen in Figure 6.37, in terms of shape, the obtained results have a pleasing subjective impact on the viewer, in the sense that no annoying artifacts appear, except for error pattern 2, where the lower part of the back fin of the fish has been slightly displaced to the left. In fact, even for error pattern 1, which corresponds to the highest D_n value, the subjective impact is quite satisfactory. The D_n metric values corresponding to the shown concealed alpha planes are: 3.49% for error pattern 1, 1.51% for error pattern 2 and 0.46% for error pattern 3. In terms of texture, the same comments that were made for shape can also be made; the only annoying artifact that can be spotted is the texture associated with the lower part of the back fin, which has also been displaced with the shape. As for the corresponding $PSNR$ values, they are 28.23 dB for pattern 1, 28.06 dB for pattern 2, and 29.53 dB for pattern 3. The uncorrupted texture has a $PSNR$ value of 32.01 dB. The large drop in the $PSNR$ values can be explained in the same way as for the Akiyo video object.

6. Shape Error Concealment for Object-based Video Coding

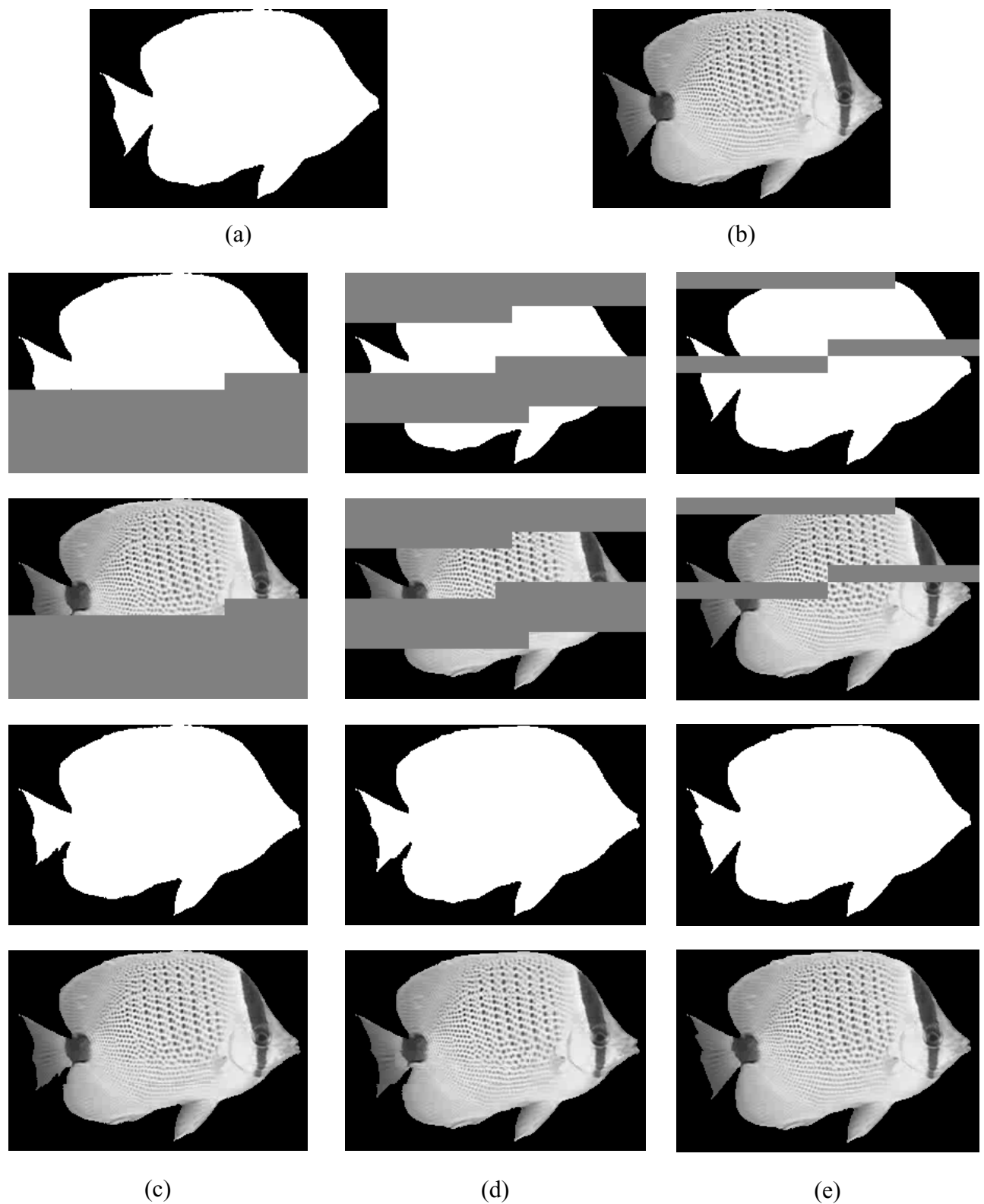


Figure 6.37 – Examples of corrupted and corresponding concealed alpha planes and textures for the Bream video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3

The last video object chosen to illustrate the performance of the proposed error concealment technique is the Stefan video object; the shape and texture of a given decoded VOP (corresponding to VOP 2 in the original 300 VOP sequence) is shown in Figure 6.38 (a) and (b). For this video object, four different corrupted versions of the alpha plane in Figure 6.38 (a) and the texture in Figure 6.38 (b) are shown in the remainder of Figure 6.38, in addition to the corresponding concealed alpha planes and textures. These four versions correspond to four of the 50 different error patterns used above for a video packet loss rate of 20%. In this case, the shape artifacts are rather visible for error patterns 2 and 4, but the results are still tolerable for critical environments such as mobile networks. In terms of the Dn metric, the values are, as expected, higher than for the previous video objects because the motion associated with this video object is much more complex. The obtained Dn values corresponding to the shown concealed alpha planes are: 2.26% for error pattern 1, 11.50% for error pattern 2, 8.86% for error pattern 3, and 7.58% for error pattern 4. In terms of texture, the artifacts are also rather visible, which is easily understandable since they are associated with shape displacements (in the head for error pattern 1 and in the racket for error patterns 2 and 4). However, as said above, they may be rather tolerable for critical environments, although it will depend on the specific application. As for the corresponding $PSNR$ values, they are 26.64 dB for pattern 1, 24.41 dB for pattern 2, 26.91 dB for pattern 3, and 21.91 dB for error pattern 4. The uncorrupted texture has a $PSNR$ value of 30.31 dB. The explanation for the large drop in the $PSNR$ values is basically the same as for the other two video objects, but here in addition to the global motion compensation a lot local motion refinement has also been applied.



(a)



(b)

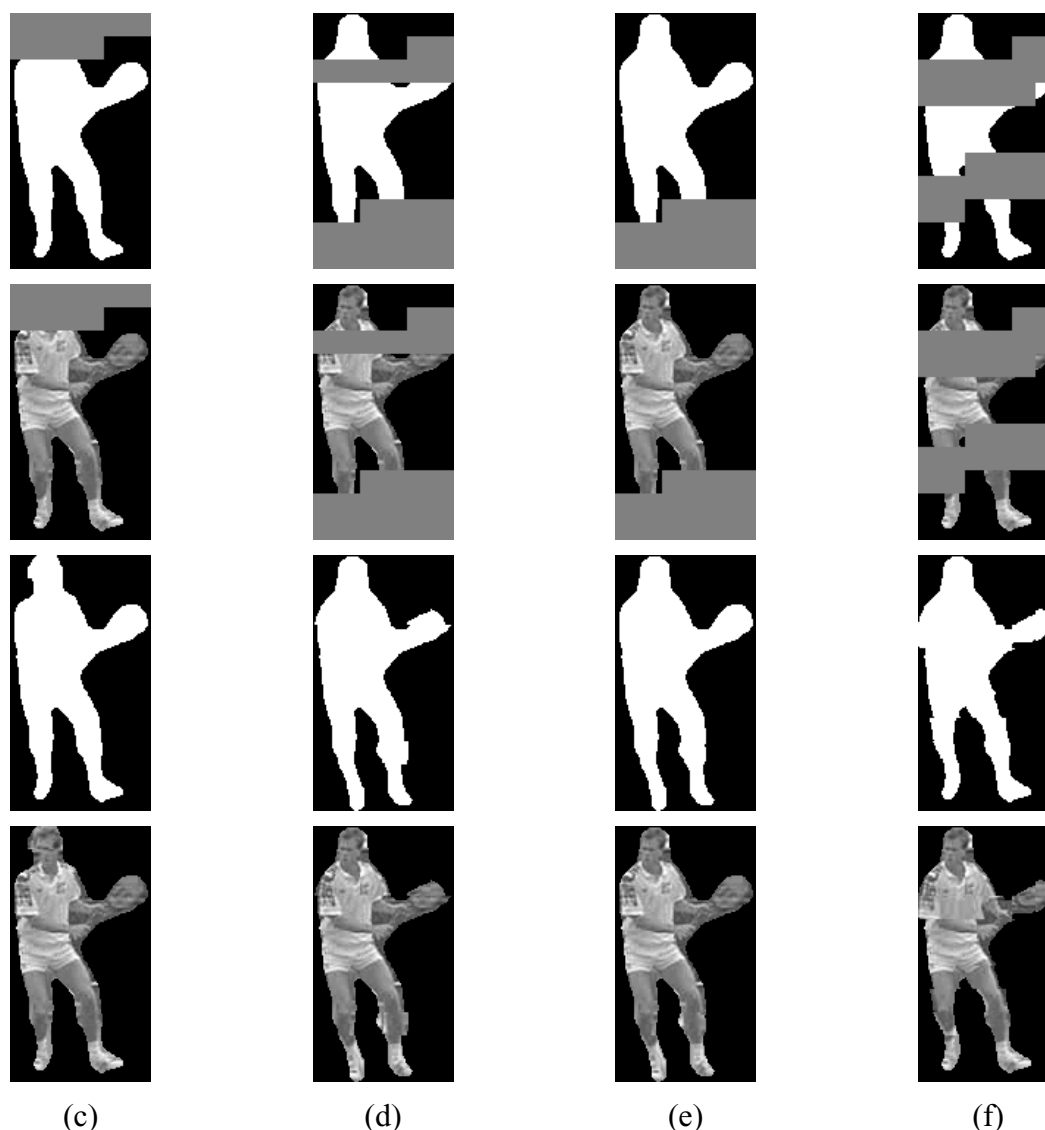


Figure 6.38 – Examples of corrupted and corresponding concealed alpha planes and textures for the Stefan video object with a video packet loss rate of 20% – (a) Uncorrupted shape; (b) Uncorrupted texture; (c) Error pattern 1; (d) Error pattern 2; (e) Error pattern 3; (f) Error pattern 4

In order for the reader to have a better idea of the improvements due to the local motion correction, the corrupted alpha planes and textures shown in Figure 6.38 have also been concealed without it (i.e., only global motion compensation was used). The results are shown below in Figure 6.39 and, as can be seen, the artifacts in areas with considerable local motion are much more evident than they were in Figure 6.38. This is especially the case for the racket area, in Figure 6.39 (b) and (d), and for the left leg, in Figure 6.39 (c).

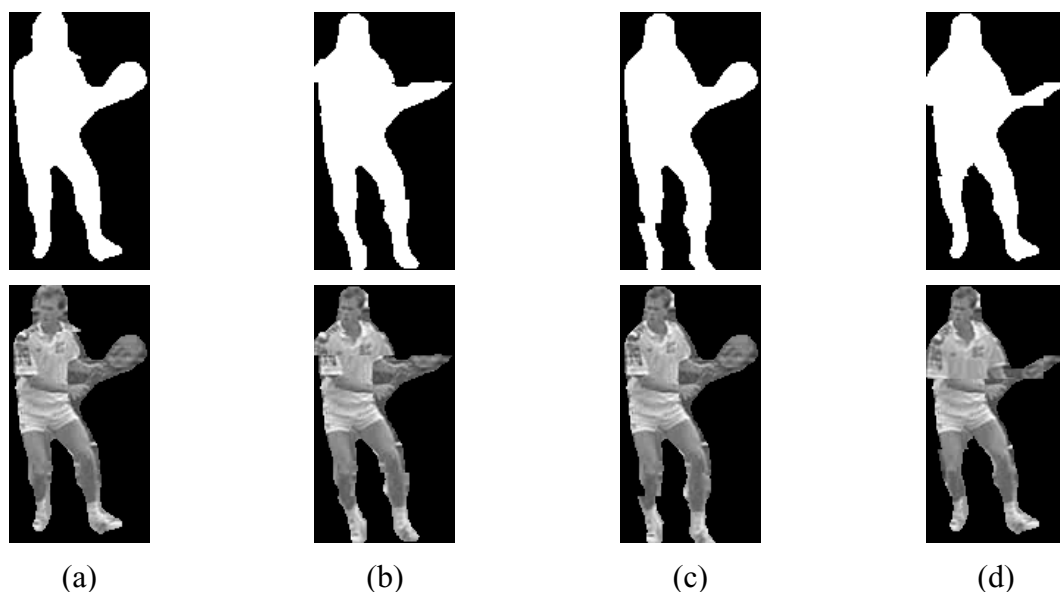


Figure 6.39 – Examples of concealed alpha planes and textures for the Stefan video object without local motion refinement (video packet loss rate of 20%) – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4

Still for the shape and texture of the Stefan video object in Figure 6.38 (a) and Figure 6.38 (b), respectively, another four different corrupted and concealed versions are shown in Figure 6.40. These four versions correspond to four of the 50 different error patterns used above for a video packet loss rate of 10%. In this case, the artifacts are still visible but the situation is not as bad as before. The type of artifacts that appear when a packet loss rate of 10% is used is basically the same as when a packet loss rate of 20% is used. The main difference is that, for the 20% case, more artifacts appear in each time instant. In some of these cases, additional information, from the same object or from other objects in the scene, from the current time instant as well as other previous time instants could be very useful. For instance, if another object, such as a background, with a complementary shape were available, the task of the decoder could be made much easier in terms of concealing the shape. Alternatively, or in combination, the spatial error concealment technique proposed in Section 6.2 could also be used, probably providing better results than the present technique for some parts of the corrupted shape. In terms of the Dn metric corresponding to the shown concealed alpha planes, the values are, as expected, lower than for the previous case of a video packet loss rate of 20%: 4.11% for error pattern 1, 1.70% for error pattern 2, 1.78% for error pattern 3, and 2.32% for error pattern 4. As for the corresponding $PSNR$ values, they are 27.74 dB for pattern 1, 24.19 dB for pattern 2, 24.19 dB for pattern 3 and, finally, 26.25 dB for error pattern 4.

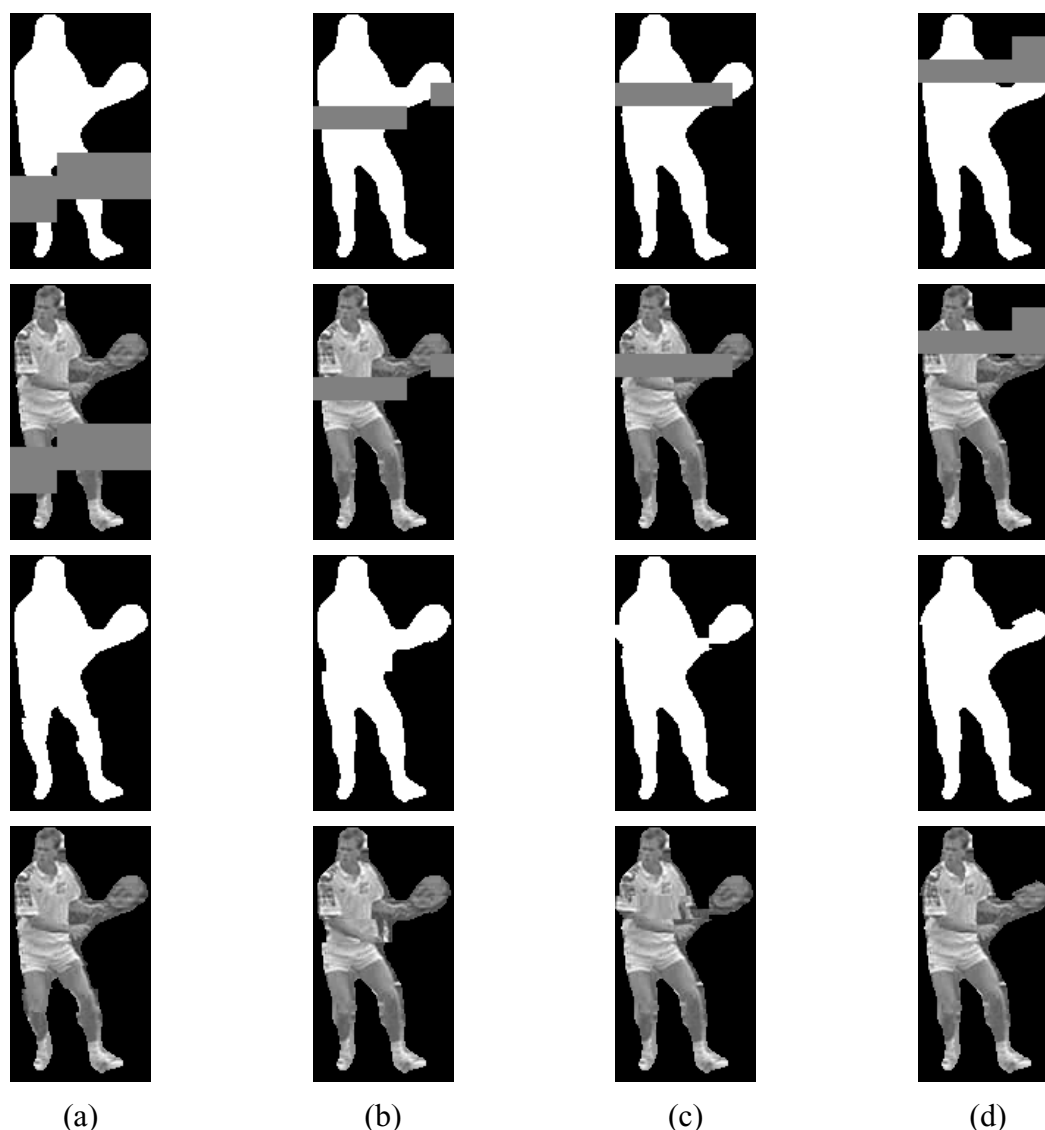


Figure 6.40 – Examples of corrupted and corresponding concealed alpha planes and textures for the Stefan video object with a video packet loss rate of 10% – (a) Error pattern 1; (b) Error pattern 2; (c) Error pattern 3; (d) Error pattern 4

6.3.2.2 Results with a Shape Intra Refreshment Period of 3

In the next set of results all the encoding parameters that were used previously are maintained except for the shape data refreshment rate. Now, instead of refreshing the shape data of the test sequences at each time instant, the refreshment is applied every three time instants. With this alpha refreshment rate, the error-free decoded texture qualities presented in Table 6.6 are obtained. When compared to the values obtained in Table 6.3, these values show some improvements, which are higher for the Akiyo video object and progressively lower for the Bream and Stefan video objects. This is easily explainable by considering the nature of these video object sequences. In Akiyo, the shape hardly changes in consecutive time instants and, therefore, by reducing the shape intra refreshment rate many bits will be saved which can be used to improve the texture quality (i.e., 0.46 dB). As for the Bream video object, the shape changes slightly more in consecutive time instants and, therefore, not as many bits will be saved when the shape intra refreshment rate is reduced, but they are still enough to slightly improve the texture quality (i.e., 0.06 dB). Finally, for the Stefan video object, the shape

changes a lot in consecutive time instants and hardly any bits are saved by reducing the amount of shape intra refreshment, which translates in the fact that the texture quality remains unchanged.

Table 6.6 – Error-free PSNR obtained for the chosen encoding parameters

	Akiyo	Bream	Stefan
Error-free PSNR [dB]	34.31	30.07	30.46

As can be seen in Table 6.7, the concealment results have considerably degraded when compared to those obtained with a shape refreshment period of 1 and shown in Table 6.4. The reason for this is the temporal error propagation, which can be very severe in the shape data, due to the way it is coded. As explained in Chapter 4, in order for the decoder to be able to actually decode the prediction error of an inter coded shape block, it is necessary that the past shape data on which it depends be uncorrupted. This may cause some very serious error propagation problems because, when the prediction error cannot be decoded, this typically means that the decoder will lose synchronization and, thus, discard all the data up to the next resynchronization point in the bitstream. In fact, this is much more severe than for the texture, where the actual decoding of the prediction error for an (inter coded) texture block does not depend on the past texture used as prediction; the past texture only affects the subsequent reconstruction of the texture blocks when it is added to the prediction error. Thus, in the case of the texture, past errors will only propagate to the texture blocks that use corrupted texture for prediction. With the shape, however, past errors can propagate to shape blocks other than the ones that explicitly depend on the past corrupted shape data.

In terms of texture quality, whose results are included in Table 6.8, the same type of behavior occurs with respect to the results shown in Table 6.5. At first, this may sound strange because of what was said about the error propagation being much less severe in the texture data. However, in the previous paragraph, the error propagation from the shape data to the texture data was not considered. After all, the correct decoding of the texture data depends on the correctness of the shape because the shape is used to indicate which blocks have associated texture data to be decoded. This means that the severe propagation of errors associated with the shape data will end up affecting also the texture data.

In this section, no visual results are included because the aspect of individual VOPs is very similar to aspect shown in Section 6.3.2.1. The main difference, when a shape intra refreshment period of 3 is used, is that more VOPs in the whole sequence are corrupted and have to be concealed, thus explaining the degradation of the numerical results (which are temporal averages).

Table 6.7 – Dn values for the tested video object sequences

Video packet loss rate	Dn [%] ($Dn_{low}/Dn_{avg}/Dn_{high}$)								
	Akiyo			Bream			Stefan		
1%	0.01	0.74	7.94	0.01	0.19	0.44	0.13	0.72	1.45
5%	0.14	2.62	10.50	0.67	1.94	7.80	2.19	4.03	7.44
10%	0.41	5.95	22.60	1.27	6.05	13.16	5.29	8.56	14.95
20%	3.84	17.86	31.83	6.37	16.51	31.46	13.54	20.26	33.79

Table 6.8 – PSNR values for the tested video object sequences

Video packet loss rate	PSNR [dB] (PSNR _{low} /PSNR _{avg} /PSNR _{high})								
	Akiyo			Bream			Stefan		
1%	28.14	32.02	33.73	26.02	28.18	29.75	25.94	28.19	29.80
5%	22.76	26.70	29.32	19.89	23.58	25.45	20.58	23.10	25.11
10%	17.25	23.15	25.92	17.12	20.30	22.78	18.69	20.36	22.06
20%	16.18	18.61	21.73	13.87	16.27	18.54	16.12	17.53	18.49

The results above help to emphasize the importance of the shape intra refreshment technique proposed in Chapter 5 because, as was seen here, by reducing the shape refreshment rate of a given object, its shape quality as well as texture quality will decay considerably. This way, it is important that objects that are more relevant to the user and harder to conceal have their shape refreshment periods reduced, possibly at the expense of others less important, that are also easier to conceal. For instance, the video objects used in the present chapter correspond to objects that would very likely be the most relevant to the user in a scene. Thus, it is possible to see the degradation that can be avoided by increasing the shape refreshment rate of these objects at the expense of others that would correspond to less important objects or to the background.

A direct comparison of the technique proposed here and the one proposed in [Salama02] (previously described in Chapter 3) is very difficult because in [Salama02] no numerical results were provided. However, for very simple cases where the shape motion can be described in terms of global motion alone, it can be expected that the technique proposed here be slightly inferior to the one proposed in [Salama02], where global motion parameters are computed at the encoder using uncorrupted video objects. However, the technique proposed here eliminates the very limiting disadvantage of having the decoder rely on specific data produced by the encoder, which is a rather unrealistic scenario at least in the context of applications using coding standards (in fact it would only work for encoder-decoder combinations from the same manufacturer unless the auxiliary stream with global motion data would be standardized). For other more realistic cases, where the shape motion can not be described in terms of global motion alone, the technique proposed here will clearly outperform the one proposed in [Salama02]. The disastrous effects of trying to conceal this type of shape data with global motion compensation alone were illustrated in Figure 6.22.

6.4 Combination of the Spatial and Temporal Concealment Techniques

The two shape error concealment techniques proposed in this chapter have their own advantages and disadvantages and cannot be used with acceptable results for all possible situations. This is understandable since the concealment techniques have to “create” data based on available information and some reasonable assumptions; if the corrupted data is rather different from the available information and does not follow the used assumptions, it is more than natural that the concealment “creation” be not that accurate. Still, it is better to have some, not so good, “created” data than nothing.

For instance, in order for the proposed spatial concealment technique to perform well, the contour smoothness assumption on which it is based has to be verified. This, however, is

typically verified only for relatively small sections of the video object contour; therefore, to guarantee that contours do not have to be interpolated over large areas, small video packets should be used (and the encoder may decide to selectively do this). If very large video packets are used, this could mean that a large percentage of the alpha plane (e.g., 50%) is going to be lost due to errors and, in this situation, the spatial contour interpolation technique will very likely be unable to recover the whole contour with acceptable quality.

As for the proposed temporal concealment technique, its performance greatly depends on how well-behaved the video object sequence is in terms of motion. If the video object changes in consecutive time instants can be accurately described with a global motion model, then the concealment results will be very good. However, this is not always the case and in many situations the global motion model is not able to perfectly describe the object motion; even so, in most of these cases, good results will still be achieved due to the introduced local motion refinement. But, in other more complex cases, such as alpha planes that suffer a lot of deformation along time, even the global motion compensation followed by the local motion refinement may be unable to achieve acceptable concealment results and, therefore, the spatial concealment technique would probably be much better, considering of course that an adequate video packet size is used.

Of course, if too much of the alpha plane has been corrupted, there is not much hope for either one of the proposed techniques to recover the corrupted alpha plane with acceptable quality unless the shape is rather stable and a simple repetition of the previous shape would work well. This happens because, typically, the proposed spatial concealment technique is not able to accurately interpolate contours over very large areas and the proposed temporal concealment technique is not able to accurately estimate the global motion parameters to be later used for (global) motion compensation when only a very small percentage of the alpha plane remains uncorrupted. However, in most cases, the amount of uncorrupted data in a corrupted alpha plane is enough for either one of the two proposed techniques to be used, notably if adequate parts of the shape are considered. Therefore, it would be very interesting and very useful to be able to combine the two proposed techniques into a single technique that could overcome the drawbacks of the two tools and provide the advantages of both.

6.4.1 Proposal for an Adaptive Spatio-Temporal Shape Concealment Method

Based on what has been said in the previous section, it becomes clear that it would be interesting to have an adaptive concealment method, at the decoder, to decide when the two proposed shape concealment techniques should be used in order to achieve the best concealment performance. Without this dynamic method, one of the two proposed techniques would have to be chosen at the beginning of the communication, which can have its share of problems if the video characteristics change along the way (e.g., the alpha plane of a given video object can start by having a very smooth contour and be easily concealable with the spatial concealment technique and, later on, become spatially very complex but with a well-behaved motion). On the other hand, if a method to dynamically select one of the two proposed techniques is available at the decoder, it would be possible to perform concealment in a much more flexible and, hence, effective way. For instance, the following situations can be better dealt with:

- **Video objects with different characteristics** – Since the various video objects in the scene do not necessarily have similar characteristics, this means that different concealment techniques can be applied to different objects.

- **Time-varying characteristics of objects** – Additionally, since the characteristics of the video objects also change in time, the used shape concealment techniques can be adapted accordingly.
- **Space-varying characteristics of objects** – In fact, even within a given video object at a given time instant, different techniques can be used for the different parts of the video object, which may have different spatial and temporal variation characteristics.

Therefore, by designing a scheme that adaptively selects one of the two proposed shape concealment techniques, it should be possible to obtain the advantages of both while compensating for their disadvantages and, thus, improve the achieved shape quality.

Since the proposed techniques in this chapter correspond to object-level shape concealment techniques, they deal independently with the concealment of each object in the scene, not exploiting the relationship between the various video objects. Therefore, in the scheme proposed here the same approach is also followed, meaning that the various objects in the scene are treated in an independent fashion. Additionally, since each of the techniques above is able to independently conceal the various corrupted areas of the video object, the same is also done here. Thus, for each video object in the scene, the proposed combined scheme adaptively chooses at each time instant the most adequate technique to conceal each corrupted part of the video object.

However, before actually selecting the most adequate technique for each corrupted part, the decoder should first decide if the adaptive spatio-temporal concealment method should be used at all. For this, after the decoder has detected and localized all the errors in a given alpha plane, the first step should be to determine if any of the corrupted blocks affects the contour of the video object (i.e., corresponds to a border block). If not, concealment can be easily and effectively applied by simply filling the corrupted blocks with the values of the surrounding shapels. If, on the other hand, the contour has been affected, the decoder should determine if it is possible to use any of the two proposed concealment techniques. This basically depends on the percentage of the contour that has been corrupted. Since the contour is not known beforehand, it is not possible to know what percentage of it has been corrupted and, therefore, an alternative metric could be to consider the percentage of alpha plane blocks (from the bounding box) that have been corrupted. If the percentage is above a given threshold, it is considered that with either one of the proposed techniques the concealment results will be unacceptable and something else very simple should be tried, such as simply replacing the corrupted alpha plane with the alpha plane from the previous time instant. If, on the other hand, the threshold is not exceeded, then the adaptive spatio-temporal concealment method should be used to select which one of the two proposed concealment techniques should be used for each corrupted area. Here, a threshold of 50% will be adopted: this value has been determined based on experiments performed for the three tested video object sequences, used in the previous sections of this chapter (i.e., Akiyo, Bream and Stefan). However, if the used video content is to be significantly different from the one used in this chapter, this threshold value may have to be revised, in order to guarantee that the concealment techniques are not being used in a too difficult situation, possibly leading to less good results.

This way, the block diagram for the proposed adaptive spatio-temporal shape concealment technique is presented in Figure 6.41; the input is a corrupted alpha plane with several lost blocks, typically arranged in bursts (as was already seen). In order to be concealed, the corrupted alpha plane will go through two different consecutive steps. At the end of the concealment process, the output is a fully concealed alpha plane.

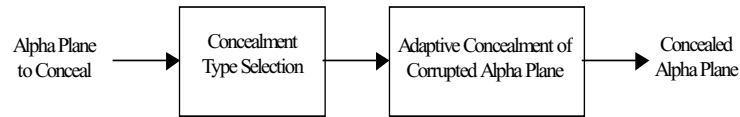


Figure 6.41 – Proposed adaptive spatio-temporal shape concealment process

Each block in Figure 6.41 corresponds to one of the two steps proposed for the spatio-temporal shape concealment process, which are:

- **Concealment type selection** – This module is responsible for determining in the corrupted alpha plane which corrupted areas are to be concealed with which type of concealment technique (i.e., spatial or temporal);
- **Adaptive concealment of corrupted alpha plane** – Finally, this module is in charge of the concealment itself for the various corrupted areas in the alpha plane; the concealment is done based on the decisions of the previous module.

The two steps above correspond to clearly separate problems and, therefore, shall be dealt with in the following sections.

6.4.1.1 Concealment Type Selection

After it has been decided that the proposed techniques should be used to conceal the corrupted alpha plane, the decoder has to consider the various independent corrupted areas of the alpha plane and decide, for each one, which one of the two shape concealment techniques should be used. In order to define an algorithm to make this selection, the characteristics of the two proposed techniques should be considered. While the spatial concealment technique is only able to produce accurate concealment results when the corrupted segment of the contour to be concealed is relatively short, the temporal concealment has no such limitation and can produce accurate results independently of the length of corrupted contour segment, as long as the object is relatively well-behaved in terms of motion. In fact, for the cases where the corrupted contour segment is short, the spatial concealment technique is typically able to achieve comparable (or even better) results than the temporal technique. And, therefore, for such cases, the spatial concealment technique can be used independently of how well-behaved the object is in terms of motion. On the other hand, when the corrupted contour segment is rather long, the results obtained with the spatial concealment are expected to be worse than the ones obtained with the temporal concealment. This is especially true if the motion is well-behaved but, even if it is not, the results will still be acceptable due to the local motion correction. Still for the case where the corrupted contour segment is rather long, when the motion is so complex that the temporal concealment technique is not able to produce good results, it is also unlikely that the spatial concealment technique will be able to do so.

Thus, the decision of which concealment technique will be applied to a given corrupted area can be based on the *length of the longest contour segment* that has been lost inside the corrupted area in question and has to be concealed. This length can then be compared to a length threshold to decide which one of the two proposed techniques will be used. The problem with this is that this length cannot be determined since the contour has been lost in the corrupted area. However, it can be estimated by considering the factors on which it depends. For this, the following two factors were identified:

- **Distance between coupled contour endings** – The first factor on which the length of the corrupted contour segment depends is clearly the distance that separates the two corresponding contour endings. This is clearly illustrated in Figure 6.42, where the two corrupted alpha planes have corrupted areas with the same size but correspond to very different lengths of corrupted contour segments. For corrupted areas where more than two contour endings exist, the decoder has first to couple the available contour endings for each corrupted segment according to the procedure defined in Section 6.2.1.2 and then decide for which pair the contour endings are farther apart.

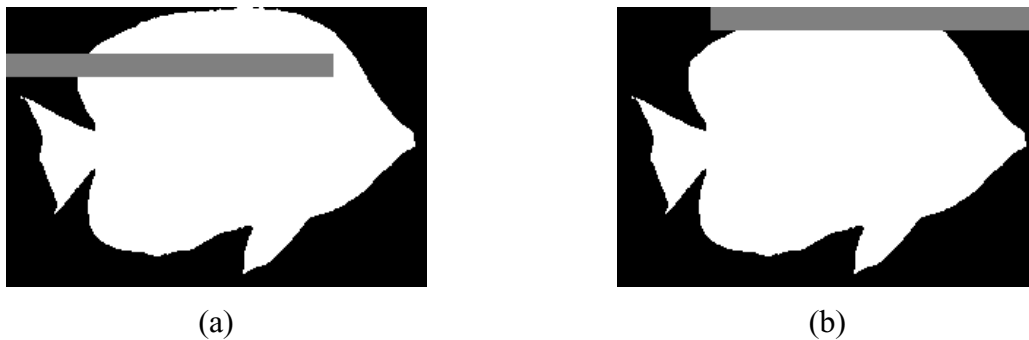


Figure 6.42 – Examples of corrupted alpha planes where the corrupted areas have the same size, but the length of the corrupted contour segment is very different – (a) Short corrupted contour segment; (b) Long corrupted contour segment

- **Size of the corrupted area** – However, the length of the corrupted contour segment does not depend only on the distance that separates the corresponding contour endings. In addition, it also depends on the size itself of the corrupted area. This happens because large corrupted areas may hide additional important information, which cannot be simply inferred from the contour endings alone. For instance, in Figure 6.43, three different corrupted alpha planes are shown with one corrupted area. In all three, the distance that separates the coupled contour endings is similar. However, the length of the corrupted contour segment inside the corrupted area is very different. As can be easily understood, for the cases with larger corrupted areas, contour interpolation will very unlikely yield a good result.

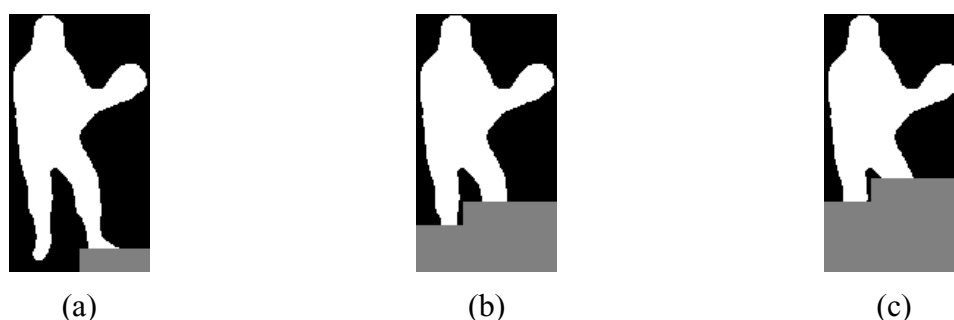


Figure 6.43 – Examples of corrupted alpha planes where the maximum distance separating the coupled contour endings are similar, but the size of the corrupted areas is very different – (a) Small corrupted area; (b) Large corrupted area; (c) Larger corrupted area

Therefore, based on these two factors that influence the usability of the spatial concealment technique, for a given corrupted area, the following metric S can be defined:

$$S = d_{\max} \times A_{\text{corrupted}} \quad (6.33)$$

where d_{\max} is the (Euclidean) distance, measured in shapels, separating the coupled contour end points farthest apart for the corrupted area in question and $A_{\text{corrupted}}$ is the size of the corrupted area in shape blocks. This way, to decide which shape concealment technique should be used for a given corrupted area, the decoder can compare S with a threshold S_{th} according to the following rule:

$$\begin{cases} \text{if } S \leq S_{th}, & \text{use the proposed spatial shape concealment} \\ \text{if } S > S_{th}, & \text{use the proposed temporal shape concealment} \end{cases} \quad (6.34)$$

As for the threshold S_{th} , it has to be determined according to the type of content that is going to be used. After all, the definition of what is a short corrupted contour segment and what is a small corrupted area are intrinsically related to the video content and its concealment difficulty.

6.4.1.2 Adaptive Concealment of Corrupted Alpha Plane

Finally, after the decision of which shape concealment technique (spatial or temporal) should be used has been taken, for each corrupted area, it is time for the concealment itself to be applied. This is simply a matter of proceeding as explained in the previous sections devoted to spatial and temporal concealment techniques. This way, if it was decided that a given corrupted area in the alpha plane is to be concealed with the spatial technique, then the concealment process described in Section 6.2.1 is followed. If on the other hand, the corrupted area is to be concealed with the temporal concealment technique, then the concealment process described in Section 6.3.1 is followed.

Before proceeding to the next section, it should be noted that, by carefully choosing how to divide the video object data in several video packets, the encoder can positively influence the concealment decisions (and results) at the decoder, especially if it knows about the shape concealment selection method used at the decoder (but not necessarily). For instance, if the encoder knows that a given part of the alpha plane, if corrupted, should be concealed with the spatial technique, it should use small video packets for that area in order for small corrupted areas to appear at the decoder, which will then use spatial concealment. On the other hand, if it knows that a given area can, if corrupted, be easily concealed with the temporal concealment technique, it can use large video packets instead (and thus save the overhead associated with video packets). This way, by carefully analyzing the video objects being encoded, the encoder can thus bias the decoder to make a given concealment decision since the size of the packets will largely determine the size of the corrupted areas.

6.4.2 Performance Evaluation

To illustrate the performance of the proposed adaptive spatio-temporal concealment method, showing the advantages of using it instead of just one of the previous two techniques proposed in this chapter, two video object sequences will be used in the following. The first one is the Stefan video object sequence and it was chosen because it is the one, among the three tested in previous sections of this chapter, where the two proposed techniques alone have the most difficulty in performing the concealment. For the other two tested video

objects (i.e., Akiyo and Bream), either one of the two proposed concealment techniques alone provides already very good results and, therefore, the improvements with the adaptive spatio-temporal concealment technique are not so significant. In addition to Stefan, the First Man video object from the Hall Monitor sequence will also be used here (see Annex A) because it also corresponds to a video object where the use of the adaptive spatio-temporal concealment technique can lead to significant improvements.

As explained in Section 6.4.1.1, before applying the adaptive spatio-temporal concealment technique to a given type of video content, the S_{th} threshold has to be determined. To do this for the type of content here considered, a reference d_{max} value corresponding to the distance of a shape block diagonally traversed is considered (i.e., 23 shapels), while the reference $A_{corrupted}$ value is considered to be 6 shape blocks; these values have been determined experimentally based on experiments performed for video object sequences with characteristics similar to the objects used in this section. Then, by using Equation (6.33), an S_{th} value of 138 was obtained, whose adequacy may easily be confirmed by the reader when the results are presented below.

The first video object sequence on which the proposed adaptive spatio-temporal concealment technique was tried is the Stefan video object, of which a few examples are shown in Figure 6.44. These selected examples adequately reflect the obtained results and, therefore, should be able to help the reader to learn about the performance of the method. In Figure 6.44 (a), the uncorrupted original alpha plane of a given decoded VOP from the Stefan video object is shown (corresponding to VOP 2 in the original 300 VOP sequence). In the remainder of Figure 6.44, four different corrupted versions of the alpha plane in Figure 6.44 (a) are shown. Below each one of these corrupted alpha planes, three concealed alpha planes are included, obtained by using the proposed spatial concealment technique, the proposed temporal concealment technique and the adaptive spatio-temporal concealment method that allows to adaptively select between the two simpler concealment methods. In Figure 6.44 (b) and (c), when the adaptive method is used, Stefan's feet are concealed with the temporal concealment technique and the rest of the alpha plane is concealed with the spatial concealment method. In Figure 6.44 (d), also for the adaptive method, Stefan's head is concealed with the temporal concealment technique, while the rest of the corrupted alpha plane is concealed with the spatial concealment technique. Finally, in Figure 6.44 (e), Stefan's head, as well as his legs, are concealed with the spatial technique; the rest is concealed with the temporal concealment technique. As can be seen from these examples, by using the adaptive spatio-temporal concealment technique, the subjective impact is generally improved. The only exception to this is Figure 6.44 (e), where the corrupted area corresponding to Stefan's head has been concealed with the spatial interpolation technique; better results would have been obtained with the temporal concealment technique, which has been used on the rest of the corrupted areas in that alpha plane. This less adequate decision could be changed by decreasing S_{th} but this lower value would create many more problems for other corrupted areas and, therefore, it was decided to not change the threshold. As for the Dn values associated with these concealed alpha planes, they are shown in Table 6.9 and correctly reflect what happens in terms of subjective impact. As can be seen, in all but one case, the Dn value obtained when the adaptive spatio-temporal concealment technique is used is lower than for either one of the two individual techniques. The only exception happens for error pattern 4, where the Dn value achieved with the temporal concealment technique is lower than the one obtained with the adaptive spatio-temporal concealment technique.

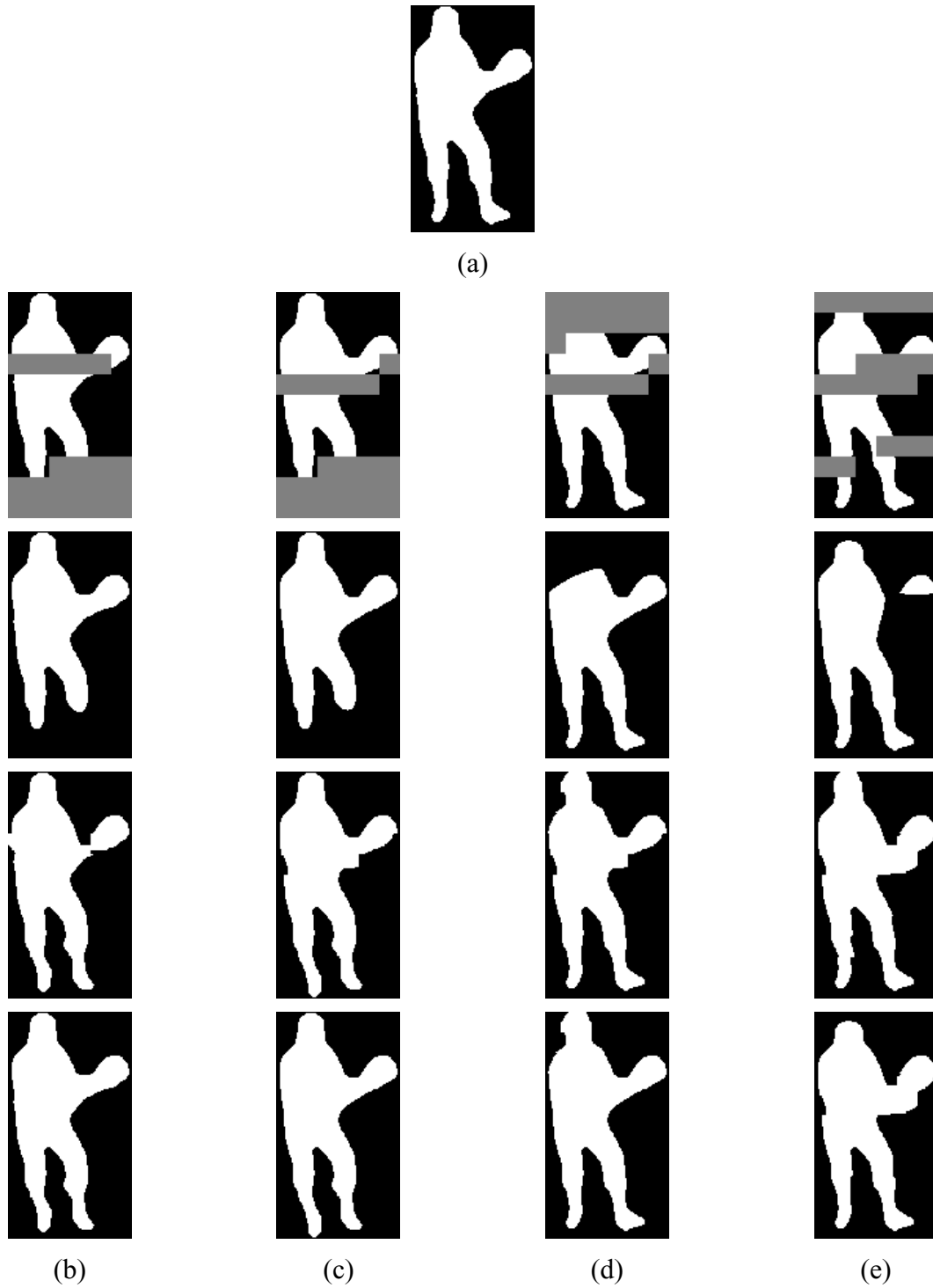


Figure 6.44 – Examples of corrupted and (three) corresponding concealed alpha planes for the Stefan video object, obtained with the proposed spatial concealment technique, the proposed temporal concealment technique and the adaptive spatio-temporal concealment technique – (a) Uncorrupted original alpha plane; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4

6. Shape Error Concealment for Object-based Video Coding

Table 6.9 – D_n values for the various concealed alpha planes shown in Figure 6.44, obtained when the various shape concealment techniques proposed in this chapter are used

Concealment technique	D_n [%]			
	Error pattern 1	Error pattern 2	Error pattern 3	Error pattern 4
Spatial concealment only	9.52	9.77	12.63	9.87
Temporal concealment only	9.84	10.56	4.30	9.01
Adaptive spatio-temporal concealment method	8.67	9.27	2.99	9.56

The second video object sequence on which the proposed adaptive spatio-temporal concealment technique was tested is the First Man video object, of which a few illustrative examples are shown in Figure 6.45. In Figure 6.45 (a), the uncorrupted original alpha plane of a given decoded VOP from the First Man video object is shown (corresponding to VOP 33¹¹ in the original 300 VOP sequence). The remainder of Figure 6.45 is organized as Figure 6.44. In Figure 6.45 (b), when the adaptive spatio-temporal concealment method is used, the First Man's head is concealed with the temporal concealment technique, while the foot is concealed with the spatial concealment method. In Figure 6.45 (c), also for the adaptive method, all the corrupted areas in the alpha plane are concealed with the temporal concealment technique. In Figure 6.45 (d), only the First Man's briefcase is concealed with the temporal concealment technique, while the rest of the corrupted alpha plane is concealed with the spatial concealment technique. Finally, in Figure 6.45 (e), only the First Man's upper torso is concealed with the temporal concealment technique; the rest is concealed with the spatial concealment technique. As can be seen from these examples, by using the adaptive spatio-temporal concealment technique, the subjective impact is generally improved. The only exception to this is Figure 6.45 (c), where all the corrupted areas in Stefan have been concealed with the temporal concealment technique and, therefore, the subjective impact is exactly the same as when only the temporal concealment technique is used. This, however, does not represent a wrong decision by the decoder because, after all, the temporal concealment technique was the best choice in this case for all the corrupted areas. As for the D_n values associated with these concealed alpha planes, they are shown in Table 6.10 and correctly reflect what happens in terms of subjective impact. As can be seen, in all but one case, the D_n value obtained when the adaptive spatio-temporal concealment technique is used is lower than for either one of the two individual techniques. The only exception happens for error pattern 2, where the D_n value achieved with the temporal concealment technique is identical to the one obtained with the adaptive spatio-temporal concealment technique, which is understandable since all the corrupted areas have been concealed with temporal concealment technique.

¹¹ This VOP was used because the first few VOPs in the video object sequence are empty.

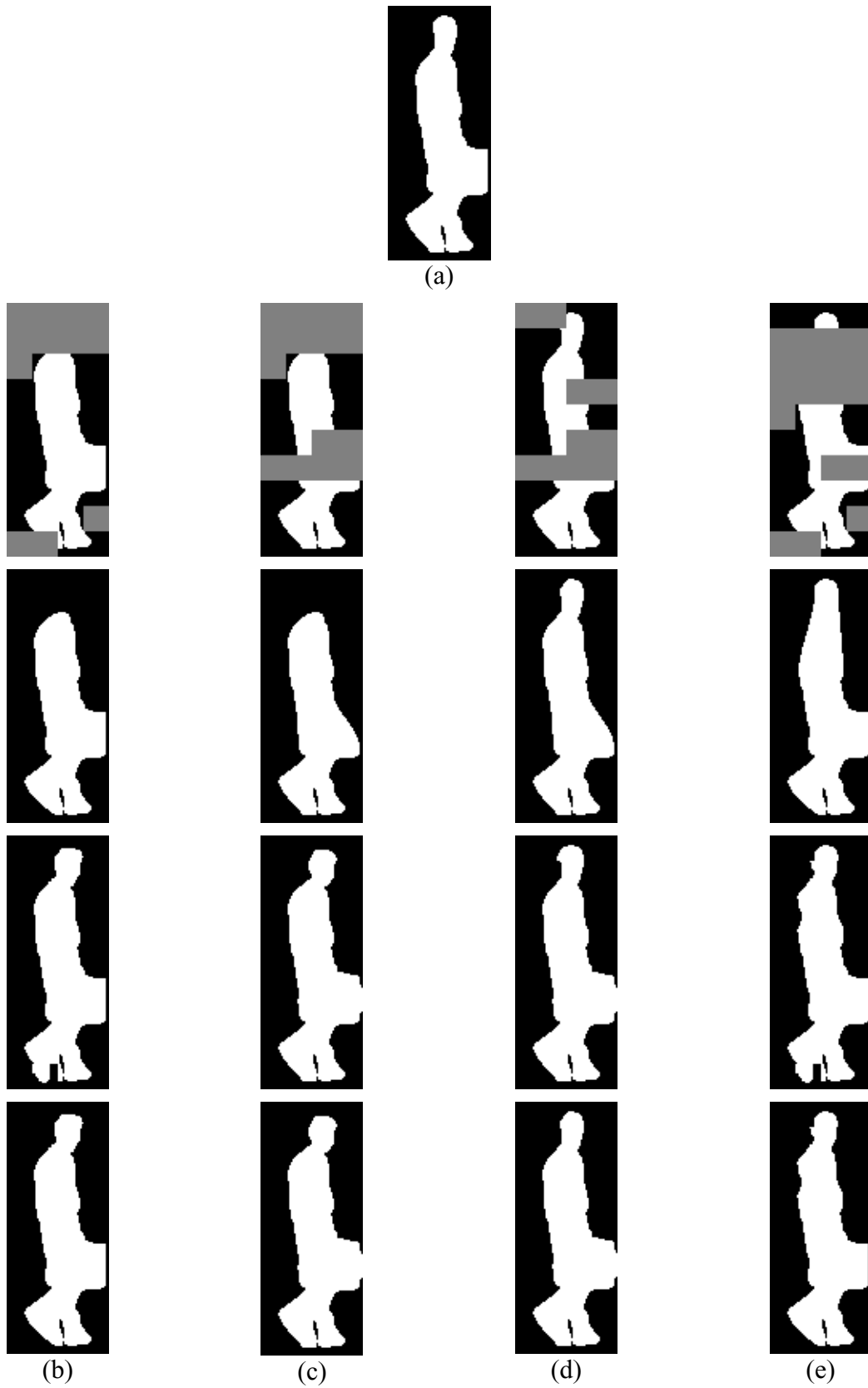


Figure 6.45 – Examples of corrupted and (three) corresponding concealed alpha planes for the First Man video object from the Hall Monitor sequence, obtained with the proposed spatial concealment technique, the proposed temporal concealment technique and the adaptive spatio-temporal concealment technique – (a) Uncorrupted original alpha plane; (b) Error pattern 1; (c) Error pattern 2; (d) Error pattern 3; (e) Error pattern 4

Table 6.10 – D_n values for the various concealed alpha planes shown in Figure 6.45, obtained when the various shape concealment techniques proposed in this chapter are used

Concealment technique	D_n [%]			
	Error pattern 1	Error pattern 2	Error pattern 3	Error pattern 4
Spatial concealment only	7.78	10.60	3.12	5.35
Temporal concealment only	6.48	6.30	2.28	5.92
Adaptive spatio-temporal concealment method	4.17	6.30	2.13	3.23

Finally, from these two examples, it can be concluded that by using the adaptive spatio-temporal concealment technique, the subjective impact, as well as the D_n values, are generally improved. This just confirms that investing in the adaptive spatio-temporal concealment method was worthwhile as expected since the drawbacks of the elementary concealment techniques are compensated and the benefits accumulated.

6.5 Final Remarks

In this chapter, two techniques were proposed to conceal shape errors in binary alpha planes or in the binary supports of gray scale alpha shapes corresponding to the objects in object-based video coding systems, such as the MPEG-4 standard.

The first technique, spatial shape concealment, assumes that most contour segments are rather smooth and uses Bézier curves to interpolate those that were broken due to channel errors. Results were presented, showing the capabilities of the technique to recover lost shape data with rather small differences with respect to the original, even for rather critical error conditions. However, if the basic smoothness assumption is not respected, some stronger artifacts may appear.

As for the second proposed technique, temporal shape concealment, it is based on a combination of global motion and local motion compensation. First, it starts by assuming that the alpha plane changes occurring in consecutive time instants can be described by a global motion model and simply tries to conceal the corrupted alpha plane blocks by using the corresponding blocks in the previous global motion compensated alpha plane. However, since not all alpha plane changes can be perfectly described by global motion, an additional local motion refinement is then applied to deal with areas of the object that have some local motion. Like for the spatial concealment technique, results have been presented showing that this technique is able to recover lost shape data with rather small distortion, even for cases where local motion exists and the global motion model is not able to perfectly describe the shape motion. However, in other more complex cases, such as alpha planes that suffer a lot of deformation along time, some artifacts may appear.

Based on the obtained results for the two proposed techniques, a method to combine them into an even more effective shape concealment scheme was also proposed. By adaptively choosing one of the proposed concealment techniques (spatial or temporal) to be used to conceal a given corrupted area in an alpha plane, the already good results achieved with the proposed techniques alone have been further improved.

Finally, to end this chapter, it is important to emphasize the relevance of shape concealment techniques, not only to achieve an acceptable shape quality, but also because the decoded

texture quality achieved is highly dependent on the quality of the shape data. Additionally, it is also important not to forget that many of the object-based functionalities depend on the quality of the shape data. Therefore, for object-based video applications to be actually used in error-prone environments, robust shape error concealment techniques will have to be available.

Chapter 7

Achievements and Future Directions

7.1 Discussion of Achievements

After the emergence of frame-based digital video, some twenty years ago, the amount of digital video content in our everyday life increased very rapidly, mainly due to the fast deployment of digital processors and digital storage/transmission media. However, frame-based digital video is still “too much” like the old analog video, in the sense that truly content-based functionalities are still not available. After all, the user can only perform very basic operations on the video information as a whole (e.g., “play”, “stop”, “pause” or “fast-forward”) and typically cannot interact with the video content itself. This type of limited interactivity is no longer able to satisfy today’s users, since many have already grown accustomed to the interactivity of the Internet (even though digital video on the Internet is still mostly frame-based).

More recently, to address these new interactivity needs, a new video representation model has been introduced: the object-based model. In this model, video data is no longer seen as a sequence of frames or fields, but consists of several independent (semantically) relevant video objects that together build the *video scene*. This new representation approach allows, in addition to the advantages already provided by the digital frame-based representation, new and improved functionalities in terms of interactivity, coding efficiency and universal access since, for the first time, the content is independently represented, accessed and consumed.

In parallel to the development of audiovisual coding systems, the number of networks being used to deliver video data has also been rapidly increasing. However, as it is well known, some of these networks have very critical bandwidth and channel error characteristics, as is the case for mobile networks and packet networks, such as the Internet. Therefore, when designing audiovisual coding systems for them, these characteristics have absolutely to be taken into account, which justifies the increasing importance of video error resilience

techniques. In particular, a significant pressure is being made to develop techniques that can be used in the context of object-based representation systems (e.g., as those provided by the MPEG-4 standard [MPEG4-V]) because, without them, the efficient transmission of object-based video over error-prone environments with an acceptable quality may not be possible. Recognizing this need, the subject of this Thesis has been selected to be the development of improved error resilience tools in the context of object-based video coding. In particular, since both encoder and decoder have an important part on the error resilience of the whole system, it was decided that techniques for both should be proposed. Thus, three main objectives were defined for the Thesis. The first one was to define a set of metrics to evaluate, at the encoder, the error resilience needs of each video object. The second objective, still at the encoder side, was to define an algorithm based on the previous set of metrics to decide the adequate amount of error resilience that should be introduced in each one of the generated individual bitstreams. Finally, the third objective of the Thesis was to propose effective shape concealment techniques to be used at the decoder.

Given the above subject, the first concern of Chapter 1, has been to precisely define the meaning of some terms that are important in this context. This is especially the case of “error resilience”, which is many times confused with “error concealment” in the literature. Afterwards, the objectives of the Thesis have been clearly defined, in addition to how they were addressed in terms of contributions. Finally, this chapter ended with a description of the organization of the remainder of the Thesis in terms of chapters.

Chapter 2 was devoted to the organization of the error resilience problem in video coding systems and the classification of the various possible solutions. This was done for both frame-based and object-based video and it was done independently of any specific (standard or not) video coding solution, even if some specific examples were given. The very first aspect to stand out from this chapter is the importance of methods that can be used to evaluate the performance of error resilience techniques. After all, in order to progress, it is necessary to be able to evaluate and compare the performance of new error resilience techniques with the available ones. Therefore, at the beginning of Chapter 2 and before going into the error resilience techniques themselves, the methods that can be used to evaluate the performance of error resilience techniques were discussed. In addition, in Chapter 2, two other very important aspects were also recognized. The first one is the fact that the error resilience of a video communication system can be globally improved by treating it at many different places in the communication chain, both at the encoder or the decoder [Soares99]. This way, an optimal solution includes the treatment of error resilience at every step of the video communication chain, in a cooperative effort. This is clearly present in this Thesis, which is naturally divided in two parts: Chapters 4 and 5 propose error resilience solutions for the encoder side of the communication chain, while Chapter 6 deals with the decoder side. The second aspect worth noting is the fact that the difficulty of the error resilience problem does not depend only on the type of video coding system that is used, but also on many other aspects, such as the channel used for delivery, the considered applications, the required functionalities to be provided and the type of content that is being delivered [Soares99].

A review of the most relevant available error resilience techniques is provided in Chapter 3 for both frame-based and object-based video coding systems. The structure adopted for this chapter is very similar to the structure of Chapter 2 to help the reader better understand under which category a given error resilience technique is included. Chapter 3 clearly shows that object-based error resilience techniques are almost non-existent when compared to the amount of available frame-based error resilience techniques. This is easily explainable by considering the fact that object-based video coding systems are still very recent, while frame-based systems have been around for a much longer time. However, it is expected that in the

near future the number of error resilience techniques for object-based video coding systems will also become very numerous.

As mentioned above, Chapter 4 is the first chapter dealing with error resilience tools at the encoder. This chapter starts by recognizing that, in object-based video, the various objects present in a given video scene may have very different characteristics in terms of error resilience needs (e.g., some objects may be inherently very easy to conceal if subject to errors, while others may be much harder) and, therefore, may need different (error resilience) treatment. In the predictive coding systems commonly used today, an effective way to introduce error resilience in a bitstream is to perform periodic intra coding refreshment, in order to refresh the decoded video quality and avoid long error propagation, thus substantially improving the video quality. With this in mind, the contribution of Chapter 4 was a set of encoder metrics that can be used by the encoder to decide how much error resilience should be introduced in the coded elementary bitstreams, in the form of intra coding refreshment. Since the visible data in object-based video is basically the shape and texture of the various objects in the scene, these metrics were proposed for both the shape and texture data of video objects and were, respectively, named *shape refreshment need* and *texture refreshment need*. These metrics express the necessity of refreshing the shape and texture data of a given video object. The shape refreshment need metric is based on two other metrics, also defined in Chapter 4: the *shape error vulnerability*, which basically measures the probability of losing part of the shape data, and the *shape concealment difficulty*, which measures how hard it is to conceal the shape data in case it is corrupted. Similarly, the texture refreshment need metric is based on the *texture error vulnerability* and the *texture concealment difficulty*, also defined in Chapter 4. The presented results have shown that the proposed metrics are able to correctly express the necessity of refreshment for each object and, therefore, should be extremely useful in allowing the encoder to efficiently decide which video objects should have their shape and texture data intra coded at each time instant to improve the subjective impact at the decoder side. At the time of writing, these metrics were the only ones of their kind available for object-based error resilient video coding. The publications related to this work are [Soares01][Soares02][Soares02a][Soares03].

In Chapter 5, the work of the previous chapter was extended by proposing an adaptive shape and texture intra coding refreshment scheme to be used at the encoder. The objective of such intra coding refreshment scheme is to efficiently decide the amount of error resilience (from the total available) that should be included in each object elementary stream to make it sufficiently robust against errors. The proposed refreshment scheme is based on the metrics defined in the previous chapter and includes shape and texture refreshment solutions, which can be applied independently or simultaneously. This way, based on the values of the refreshment need metrics, at each time instant, the refreshment scheme decides which video objects present in the video scene should have their error resilience improved by increasing the amount of shape and/or texture intra refreshment, possibly at the expense of other objects that can tolerate to have their amount of shape and/or texture intra refreshment decreased, without a significant impact on their error resilience. As for the previous metrics, the proposed intra coding scheme is the only one available in the literature dealing with both shape and texture data. Additionally, since no other object-based intra refreshment techniques are available in the literature, a simpler object-based intra refreshment scheme was proposed to be used as a reference. The reference intra refreshment scheme also includes shape and texture refreshment techniques, which as the adaptive intra refreshment techniques can be used independently or simultaneously. Comparison results have shown that the performance of the adaptive shape refreshment technique was always equal or better than the reference shape refreshment one and that the performance of the adaptive texture refreshment technique

was also always better than the reference texture refreshment one. This conclusion applies both for the shape and texture techniques used independently or combined. Based on these results, it was concluded that the use of the proposed shape and texture adaptive intra refreshment techniques is worth considering in any encoder. This work, published in [Soares02b][Soares04b], concludes the contribution of this Thesis on error resilience techniques to be used at the encoder side of the communication chain. However, a major contribution on error resilience techniques for the decoder side was proposed in Chapter 6.

After the encoder has added an adequate amount of error resilience to the coded bitstreams and transmitted them, it is time for the decoder to go to work and start decoding them, while minimizing the negative impact of any channel errors that might have corrupted the bitstream. In order to do this, the decoder should have access to error concealment techniques (in addition to error detection and localization techniques) allowing it to improve the decoded video quality. While a lot of texture error concealment techniques exist, which have been developed for frame-based video coding systems, but can be easily adapted to work also for object-based video coding systems, almost no shape resilience techniques exist in the literature. Therefore, in Chapter 6, two different shape error concealment techniques were proposed. While the first one is a spatial concealment technique, in the sense that it takes advantage only of the information at the time instant in question, the second one is a temporal concealment scheme, which is able to also use the shape information available from previous time instants. Additionally, since each one of the proposed techniques has limitations inherent to the fact that they are either only spatial or temporal, another technique was proposed, which corresponds to a combination of both techniques and therefore can be considered to be a spatio-temporal technique. The first technique, spatial shape concealment, assumes that most contour segments are rather smooth and uses Bézier curves to interpolate those that were broken due to channel errors. Results were presented, showing the capabilities of the technique to recover lost shape data with rather small differences with respect to the original, even for rather critical error conditions. However, if the basic smoothness assumption is not respected, some stronger artifacts may appear. As for the second proposed technique, temporal shape concealment, it is based on a combination of global motion and local motion compensation. First, it starts by assuming that the alpha plane changes occurring in consecutive time instants can be described by a global motion model and simply tries to conceal the corrupted alpha plane blocks by using the corresponding blocks in the previous global motion compensated alpha plane. However, since not all alpha plane changes can be perfectly described by global motion, an additional local motion refinement is then applied to deal with areas of the object that have some local motion. Like for the spatial concealment technique, results have been presented showing that this technique is able to recover lost shape data with rather small distortion, even for cases where local motion exists and the global motion model is not able to perfectly describe the shape motion. However, in other more complex cases, such as alpha planes that suffer a lot of deformation along time, some artifacts may appear. Finally, in Chapter 6, the relevance of shape concealment techniques was recognized, not only to achieve an acceptable shape quality, but also because the decoded texture quality achieved is highly dependent on the quality of the shape data. Additionally, it is also very important not to forget that many of the object-based functionalities depend on the shape data. Therefore, for object-based video applications to be actually used in error-prone environments, robust shape error concealment techniques will have to be available. The publications related to this work are [Soares04][Soares04a][Soares04c][Soares04d][Soares04e].

In addition to the work just described above, as already mentioned in Chapter 1, some other work was done during this Thesis, which did not directly fit in the major objectives of the

Thesis. The publications related to this work are [Soares98][Soares98a][Soares99][Soares99a][Soares00].

7.2 Future Directions

This Thesis has been focused on the development of error resilience techniques for object-based video coding systems. In particular, the algorithmic development has been directed towards intra coding refreshment at the encoder and shape data concealment at the decoder. This section discusses some of the work items directly related to the topics developed in this Thesis, which deserve to be pursued in the future:

- **Video Object Quality Evaluation** – The first topic where a lot of research can still be done and many improvements achieved is clearly in the field of quality evaluation for object-based video. In this Thesis, to evaluate the quality of the proposed error resilience tools, two different approaches were used, depending on the number of decoded video objects whose quality had to be evaluated. To evaluate the quality of a single video object, as is commonly done in the literature, two different metrics were used: the Dn metric for shape and the object-based $PSNR$ metric for texture. However, these metrics have significant well-known limitations, in the sense that they are not capable of fully expressing the subjective impact of the decoded video object on the user and therefore cannot completely replace subjective tests. Nevertheless, they are still commonly used, mainly because they are simple to compute and the proposed alternatives are not convincing enough. For instance, a possible way to improve the Dn metric would be to take into account the distribution of the incorrect shapels in the video object alpha plane. After all, in terms of subjective impact, it is very different to have a lot of wrong shapels evenly distributed around the alpha plane of the object, than to have the same number of wrong shapels clustered together on one side of the alpha plane. As for the object-based $PSNR$ metric, it currently does not include the texture corresponding to wrong shapels. However, this texture can have a very strong influence on the subjective impact and, therefore, should also be taken into account. Besides this, it would be much better to be able to evaluate the quality of a given object with a single quality metric that takes into account both the shape and texture of the object.
- **Video Scene Quality Evaluation** – When a whole video scene, with various video objects, is being considered, the problem is even more complicated. In this Thesis, the solution adopted was to first evaluate the quality of every video object in the scene with the Dn and $PSNR$ metrics, as explained above, and then combine the obtained values into a scene Dn value and a scene $PSNR$ value. The final scene Dn and $PSNR$ values are simply weighted averages of the individual values, where the used weights correspond to the relevance of the respective video objects in the scene, computed as proposed in [Correia00]. However, the work in [Correia00] was made with the evaluation of segmentation quality in mind and, therefore, a specific study for error resilience would probably be justified. Additionally, in the future, other aspects should also be taken into account when considering the whole video scene, such as the semantic importance of each object and how the various objects relate to each other and fit in segmented and composed scenes.
- **Back-channeling Encoder Error Resilience Techniques** – In terms of error resilience encoding techniques, this Thesis only proposed forward techniques (i.e., intra coding refreshment) in the sense that the encoder receives no feedback from the

decoder. However, if a back-channel is available, new shape and texture error resilience schemes can be envisioned, since the decoder can use the back-channel to signal the encoder where errors occurred and in what type of data (shape or texture). Based on this information, the encoder can then adopt the most adequate strategy. For instance, if some part of the texture data has been corrupted, the encoder can just choose to intra refresh it at the next time instant. However, if some part of the shape data has been corrupted, the encoder should probably refresh the shape as well as the corresponding texture at the next time instant; after all, texture decoding depends on the correctness of the shape data. Therefore, it is believed that back-channeling error resilience techniques for object-based video are a worthwhile topic that still remains to be investigated.

- **Object-level Texture Error Concealment** – Another topic that deserves to be pursued is the definition of object-level texture error concealment techniques. After all, the main concern of this Thesis (i.e., in Chapter 6) was the definition of shape concealment techniques, but texture concealment is also absolutely necessary. This choice was made mainly because many of the existing frame-based texture concealment techniques could be extended to work for object-based video, while shape concealment techniques were almost non-existent. Therefore, it was a question of urgency. Now, that some more shape concealment techniques are available, it is believed that some research effort should be devoted to either extending the existent frame-based texture concealment techniques to work for object-based video or even develop new ones. After all, the texture data in object-based video has some specific needs and problems that need to be dealt with. For instance, when concealing corrupted texture near the borders of an object, a lot of problems may be encountered. In this case, the use of spatial concealment techniques can be difficult because the only texture that can be used for interpolations is the texture inside the object since there is no texture from the outside of the object from which to interpolate (at least when performing object-level error concealment). On the other hand, if temporal concealment techniques are used instead, problems may also arise if the object border has changed significantly in consecutive time instants; if this happens, it is not obvious how to infer the corrupted texture from the past.
- **Scene-level Error Concealment** – In this Thesis, the work on error concealment was limited to object-level concealment (i.e., concealment of single video objects). However, when complete video scenes are considered, and as was explained in Chapter 2, many new problems appear in terms of concealment. The problem of the error concealment is no longer a problem of simply replacing the lost shape and texture data with something that looks visually pleasing. For instance, the objects may have to fit together like the pieces in a jigsaw puzzle. In fact, video objects that have been independently concealed and have a visually pleasing impact on the user when considered on their own, may have a very negative subjective impact on the user when considered in the context of the scene. For instance, if the common borders of two objects do not match each other exactly, holes can appear between them, which have a very negative impact on user. Of the problems that remain to be solved after this Thesis, this is probably the most interesting one, mainly because it is such a new topic and not many ideas exist yet. However, some possible approaches can be thought of. For instance, when concealing neighboring objects that have complementary shapes, an interesting approach would be to use the shape of the (correctly decoded) neighboring objects to conceal the corrupted shape of the other fitting object in question. On the other hand, if dealing with small holes that have

appeared between objects, they could be simply filled in with texture interpolated from the texture of the surrounding objects.

- **Composition Information Error Resilience Techniques** – Still related to complete video scenes, there is another problem that was not dealt with in this Thesis, which is the error resilience of composition information. When transmitting a video scene in object-based video, the various objects are encoded as separate units and some additional information (i.e., the composition) is needed to compose the original video scene at the decoder. Due to the importance of this information, it is expected that some challenging problems will appear related to its resilience to channel errors. After all, errors in the composition data can lead to very different scenes than those expected.
- **Systems Layer Error Resilience Techniques** – Since, in object-based video coding systems, the various video objects present in a scene are sent as separate bitstreams, additional data is necessary for the synchronization and multiplexing of the different bitstreams, at the systems layer. Therefore, the error resilience of the synchronization and timing data should also be further investigated, as well as the resilience of the multiplexing data.

To finalize, it is important to say that in some fields the use of error resilience techniques is especially tricky and must be applied with care. This is clearly the case of the medical field. In this case, the use of error concealment techniques can be very dangerous since the decoder manipulates the decoded video in order to have a pleasing effect on the user. This is highly unacceptable here because the user (i.e., the medical doctor) does not want the video to have a pleasing effect, but to exactly reflect what is happening at the encoder side. For this case, other solutions have to be adopted to avoid such situations, such as error correcting codes at the systems layer. Another example of a field where error concealment has to be applied carefully (or not at all) is digital cinema; after all, the work of art should be reproduced exactly as the artist intended it to be.

However, in other not so specific fields, it is expected that the techniques proposed in this Thesis, as well as the ones suggested as future work and that still have to be developed, will be absolutely necessary in the near future. This expectation is based on the fact that the demand for applications that can clearly benefit from an object-based approach is increasing rapidly. In addition to this, today's users want to have access to those applications anywhere and at anytime, which means that the video content may have to be delivered over a multitude of networks, some of which with very severe channel error and bandwidth characteristics. The only way for those object-based applications to be delivered over error-prone networks is by using adequate object-based error resilience techniques. Examples of such applications are video personal communications over mobile networks or over the Internet, as well as the broadcast of multimedia content over these same two networks.

Annex A

Test Sequences

This Annex provides a brief characterization of the test sequences used throughout this Thesis. This is done to avoid repeating the description of a given test sequence each time it is used to illustrate some results. In addition, the reader can use this Annex as a reference every time he/she needs to remind some details about a given sequence without having to look for the exact location of the description in the Thesis. To help the reader locate a given test sequence within this Annex, the presentation will be done in alphabetical order.

A.1 Akiyo Sequence

The main characteristics of the Akiyo sequence are summarized in Table A.2.

Table A.1 – Main characteristics of the Akiyo sequence

Sequence name	Akiyo
Available resolutions	QCIF, CIF
Number of available video objects	2
Video object names	Background, Akiyo
Duration	10 s
Frame rate	30 fps
Video object generation method	Computer generated (Background) and Chroma-keying (Akiyo object)

The Akiyo sequence shows a typical news program with one anchorperson reading the news. Behind her, there is a typical static studio background. Some sample frames from this sequence are given in Figure A.1, with a temporal spacing of 50 frames. The two available alpha plane sequences correspond to the Background and Akiyo objects. Since the

Background object is a static rectangular object, its alpha plane is simply a rectangle where all the shapels are opaque. As for the Akiyo object, sample frames are given in Figure A.2, where white represents opaque shapels and black represents transparent shapels. This same convention shall be used for the following sequences.



Figure A.1 – Akiyo sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

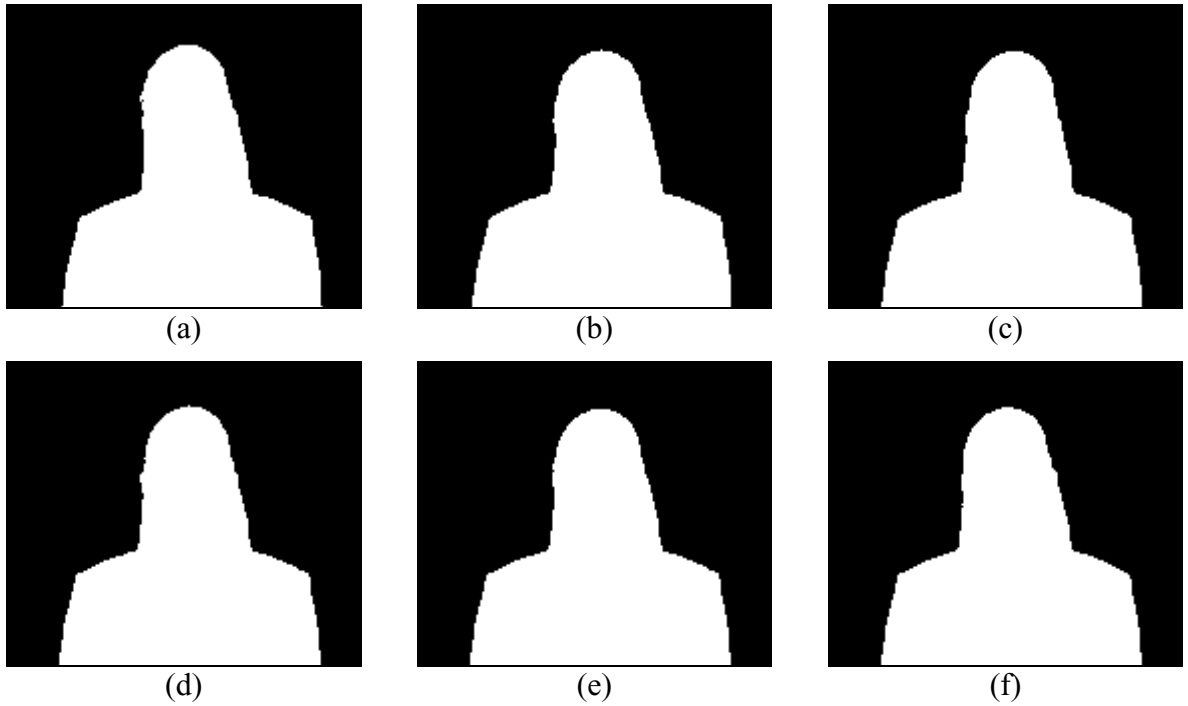


Figure A.2 – Akiyo sequence: Akiyo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.2 Children Sequence

The main characteristics of the Children sequence are summarized in Table A.2.

Table A.2 – Main characteristics of the Children sequence

Sequence name	Children
Available resolutions	QCIF, CIF, SIF
Number of available video objects	3
Video object names	Background, Kids, Logo
Duration	10 s
Frame rate	30 fps
Video object generation method	Computer generated (Background and Logo objects) and Chroma-keying (Kids object)

The Children sequence shows two kids throwing a ball at each other, over a static synthetic background. At a given point in time, a text overlay appears, flying in from the right. Later on, the text overlay goes to a deeper position in the scene and stops in the upper left corner, where it remains until the end of the sequence. Some sample frames from this sequence are given in Figure A.3, with a temporal spacing of 50 frames. The three available alpha plane sequences correspond to the Background, Kids and Logo objects. Since the Background object is a static rectangular object, its alpha plane is simply a rectangle where all the shapes are opaque. As for the remaining two objects, sample frames are given in Figure A.4 and Figure A.5.

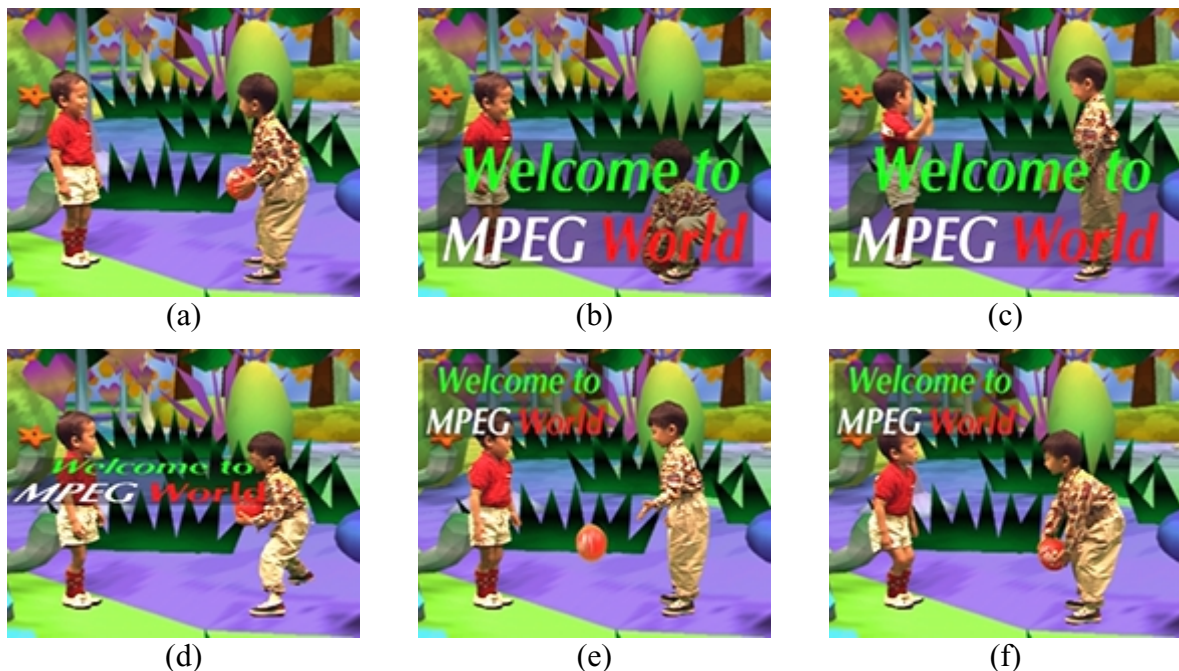


Figure A.3 – Children sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

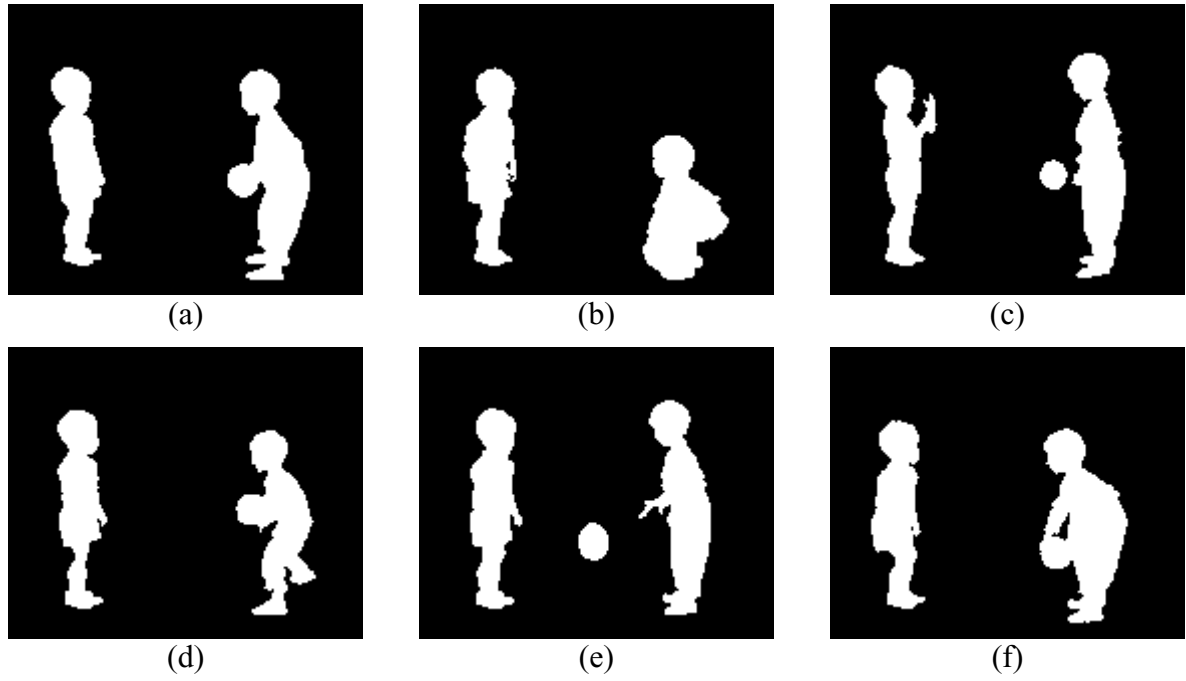


Figure A.4 – Children sequence: Kids alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

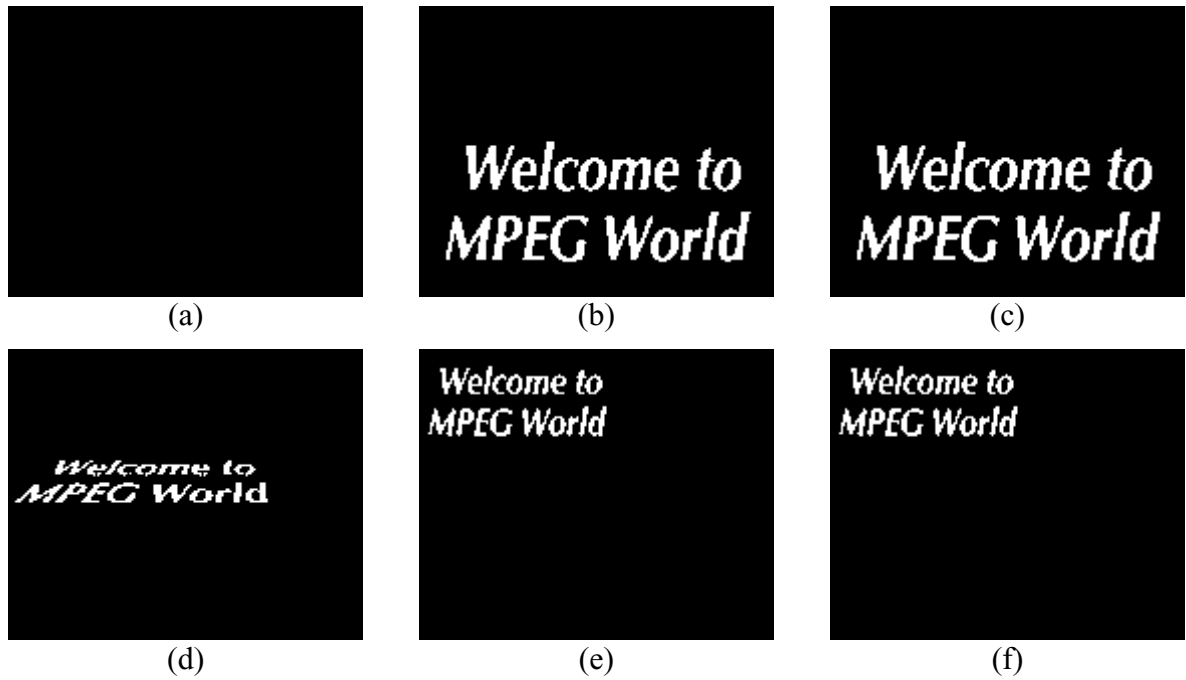


Figure A.5 – Children sequence: Logo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.3 Coastguard Sequence

The main characteristics of the Coastguard sequence are summarized in Table A.3.

Table A.3 – Main characteristics of the Coastguard sequence

Sequence name	Coastguard
Available resolutions	QCIF, CIF
Number of available video objects	4
Video object names	Water, Large Boat, Small Boat, Shore
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

In the Coastguard sequence, the camera is following a small boat moving to the left until a larger one comes into the scene from the right. At this point, there is a fast tilt up and the camera starts following the large boat moving to the right. Some sample frames from this sequence are given in Figure A.6, with a temporal spacing of 50 frames. The four available alpha plane sequences correspond to the Water, Large Boat, Small Boat and Shore objects, for which sample frames are given in Figure A.7, Figure A.8, Figure A.9 and Figure A.10, respectively.

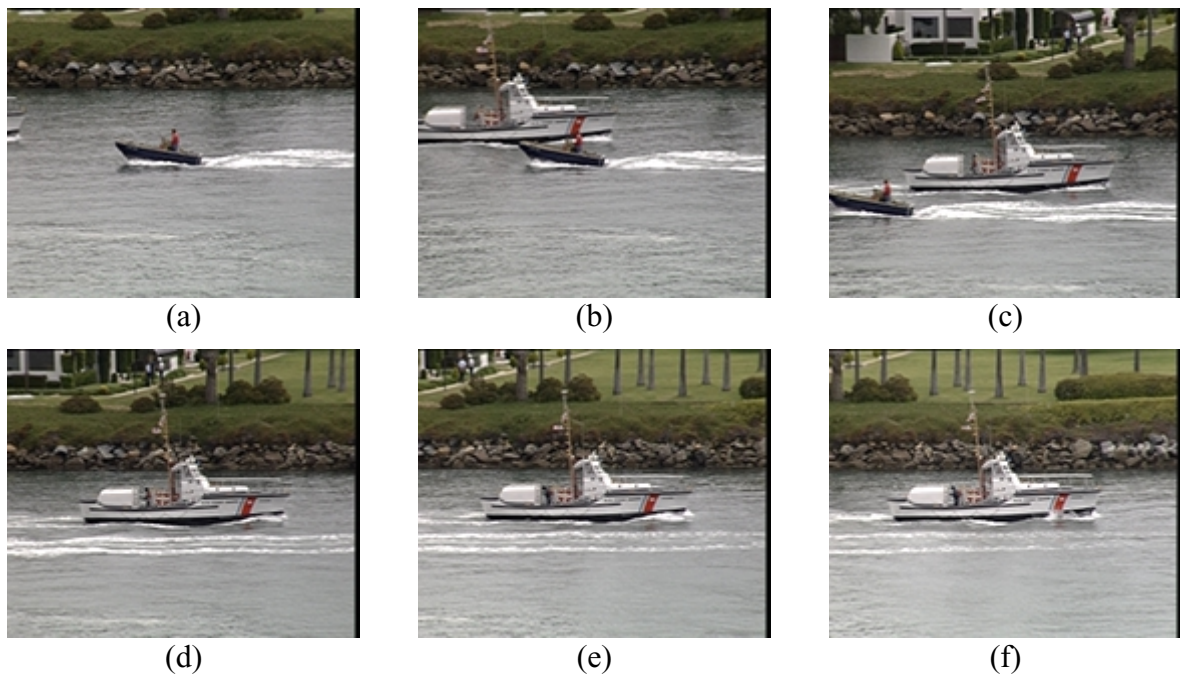


Figure A.6 – Coastguard sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

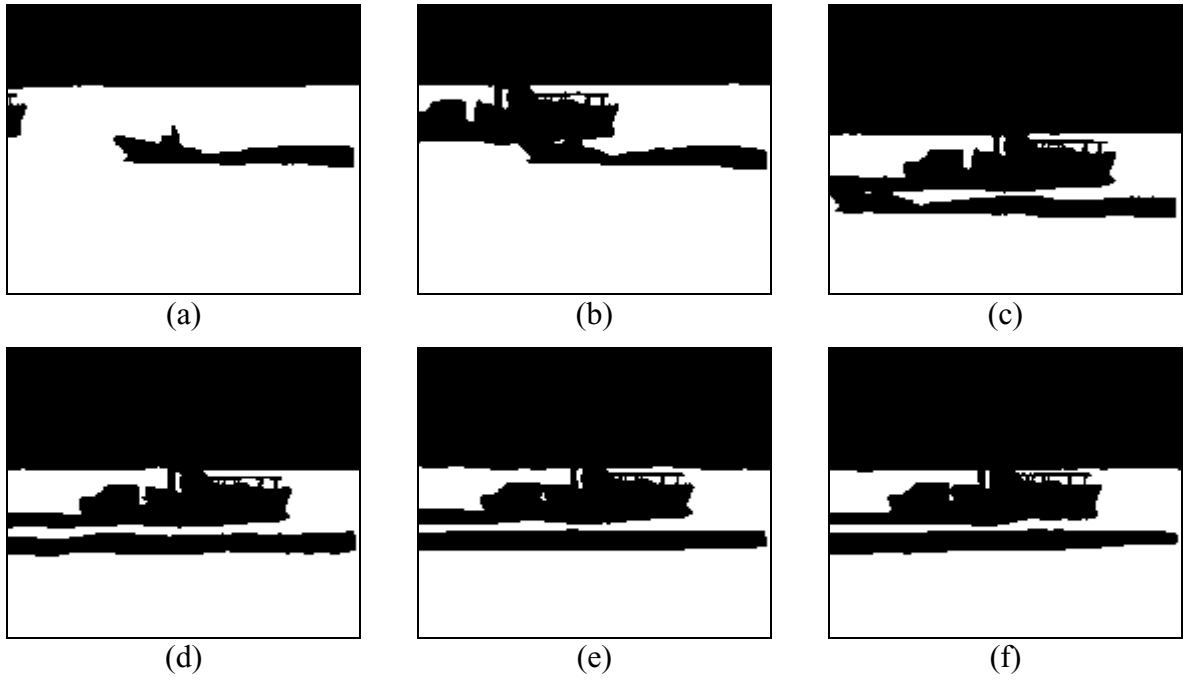


Figure A.7 – Coastguard sequence: Water alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

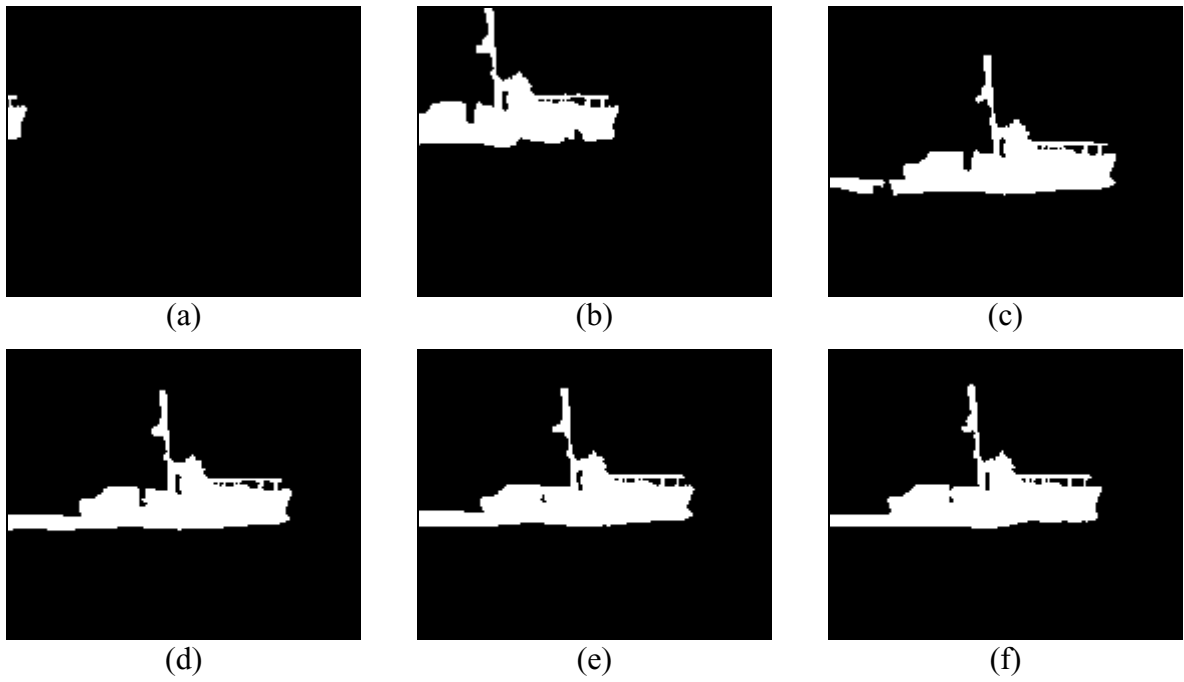


Figure A.8 – Coastguard sequence: Large Boat alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A. Test Sequences

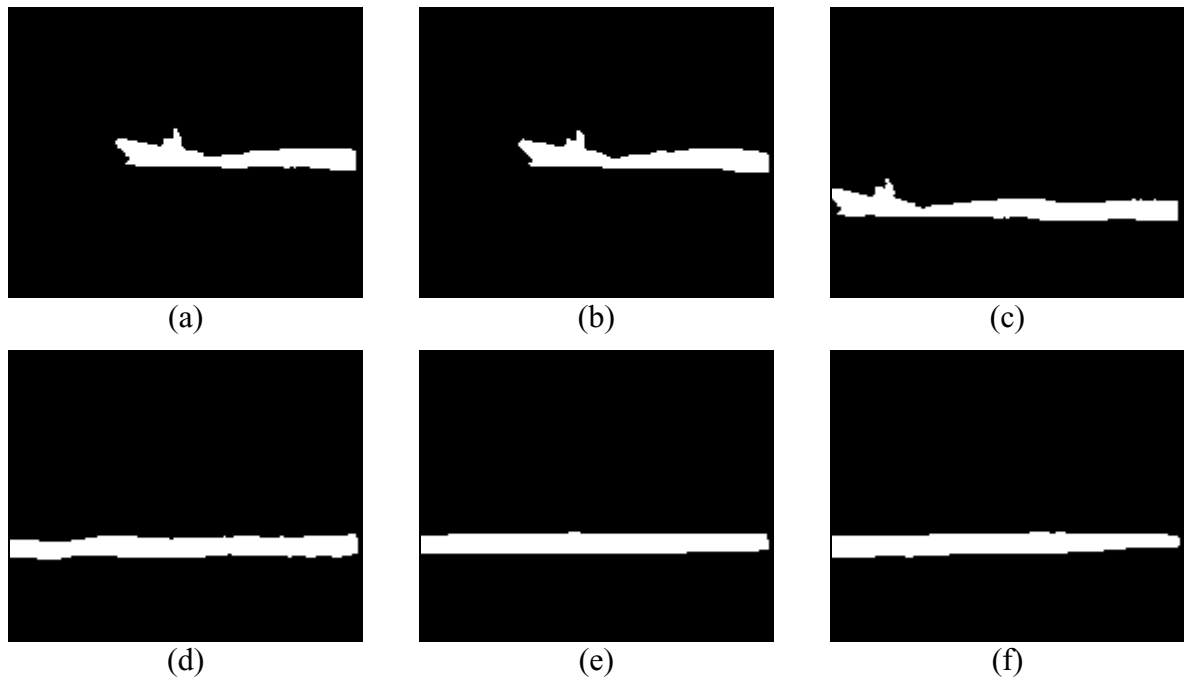


Figure A.9 – Coastguard sequence: Small Boat alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

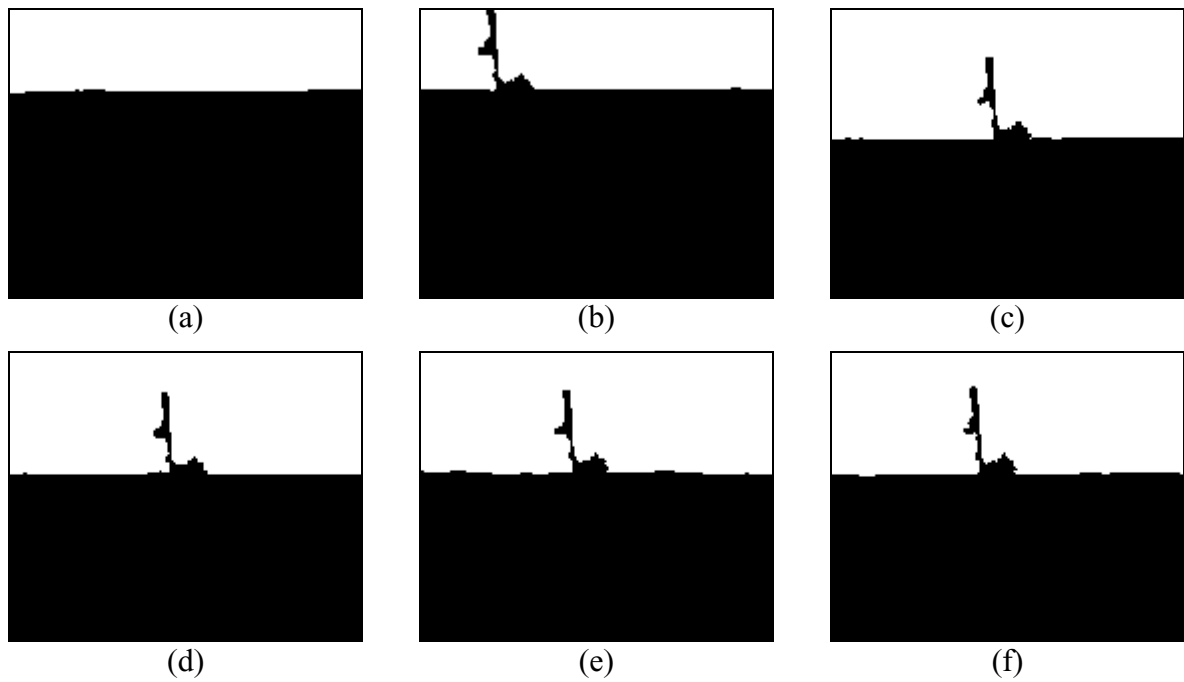


Figure A.10 – Coastguard sequence: Shore alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.4 Cyclamen Sequence

The main characteristics of the Cyclamen sequence are summarized in Table A.4.

Table A.4 – Main characteristics of the Cyclamen sequence

Sequence name	Cyclamen
Available resolutions	QCIF, CIF, SIF
Number of available video objects	1
Video object names	Cyclamen
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

The Cyclamen sequence corresponds to a long pan to the right of a plant with a flower until more plants and flowers appear, combined with a tilt down first and then a tilt up. Some sample frames from this sequence are given in Figure A.11, with a temporal spacing of 50 frames. The only available alpha plane sequence corresponds to the Cyclamen object, for which sample frames are given in Figure A.12.



Figure A.11 – Cyclamen sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

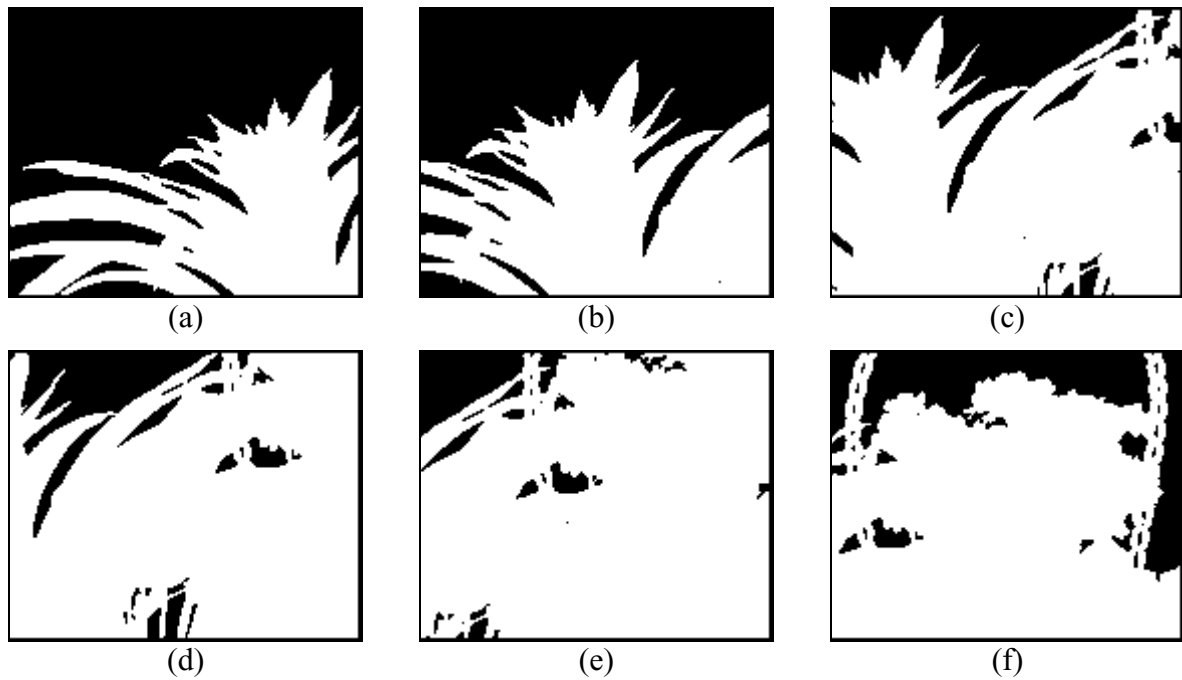


Figure A.12 – Cyclamen sequence: Cyclamen alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.5 Hall Monitor Sequence

The main characteristics of the Hall Monitor sequence are summarized in Table A.4.

Table A.5 – Main characteristics of the Hall Monitor sequence

Sequence name	Hall Monitor
Available resolutions	QCIF, CIF
Number of available video objects	3
Video object names	Background, First Man, Second Man
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

The Hall Monitor sequence is a typical surveillance sequence and it corresponds to a static surveillance camera in a corridor. First, a man appears and walks down the corridor (away from the camera). Meanwhile, another man appears at the end of the corridor and walks up to the camera. Some sample frames from this sequence are given in Figure A.13, with a temporal spacing of 50 frames. The three available alpha plane sequences correspond to the Background, First Man and Second Man objects, for which sample frames are given in Figure A.14, Figure A.15 and Figure A.16, respectively.

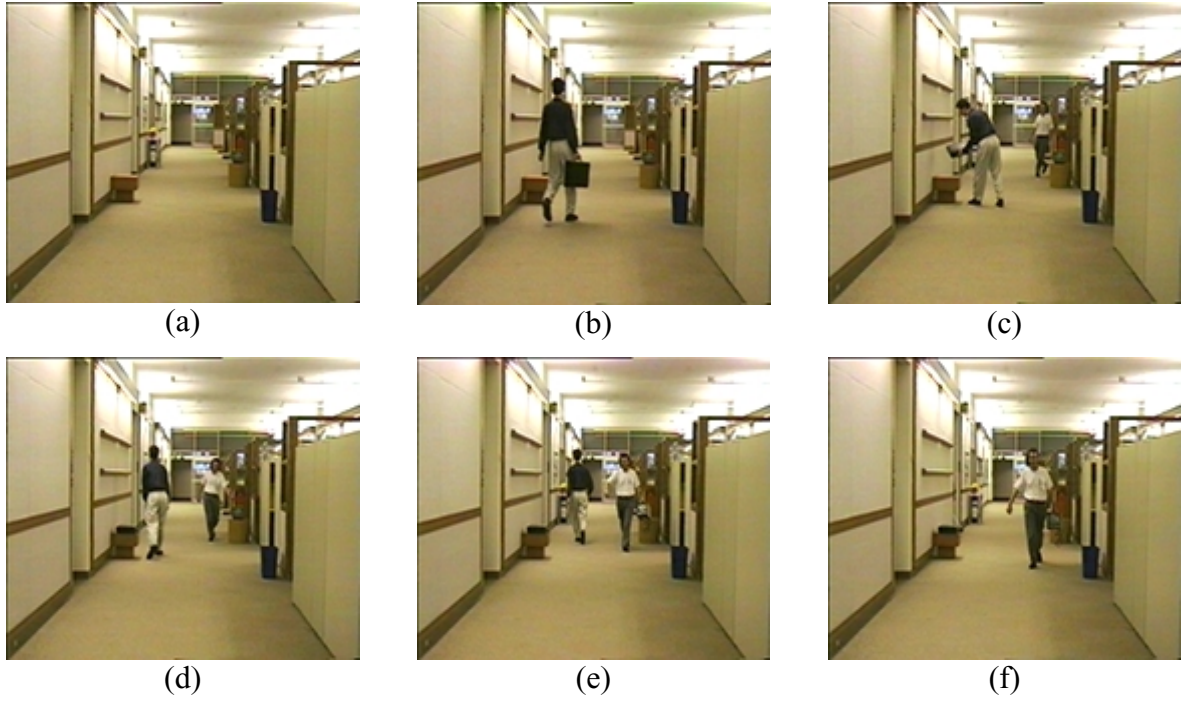


Figure A.13 – Hall Monitor sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249



Figure A.14 – Hall Monitor sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A. Test Sequences

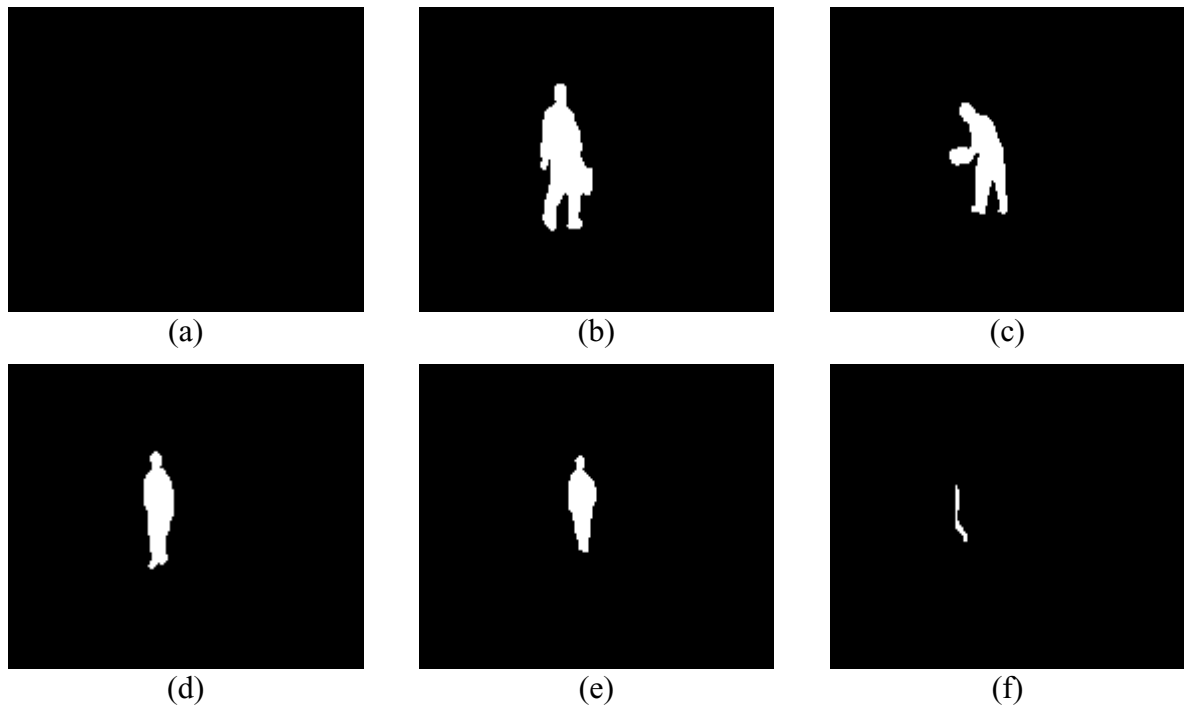


Figure A.15 – Hall Monitor sequence: First Man alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

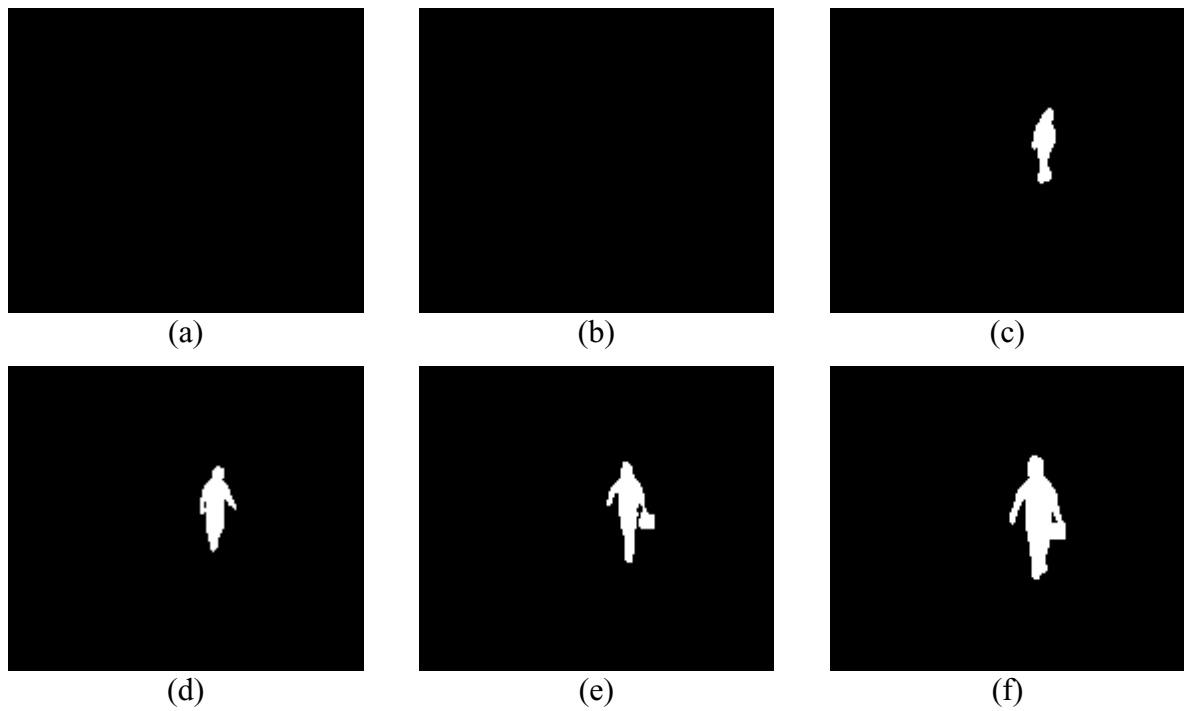


Figure A.16 – Hall Monitor sequence: Second Man alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.6 News Sequence

The main characteristics of the News sequence are summarized in Table A.6.

Table A.6 – Main characteristics of the News sequence

Sequence name	News
Available resolutions	QCIF, CIF
Number of available video objects	4
Video object names	Background, Dancers, Newscasters, Logo
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

The News sequence shows a typical news program with two anchorpersons reading the news. Behind them, in addition to the static background, there is a large monitor showing a video related to the news being read. In the lower left corner, a static text overlay is present. Some sample frames from this sequence are given in Figure A.17, with a temporal spacing of 50 frames. The four available alpha plane sequences correspond to the Background, Dancers, Newscasters and Logo objects, for which sample frames are given in Figure A.18, Figure A.19, Figure A.20 and Figure A.21, respectively.

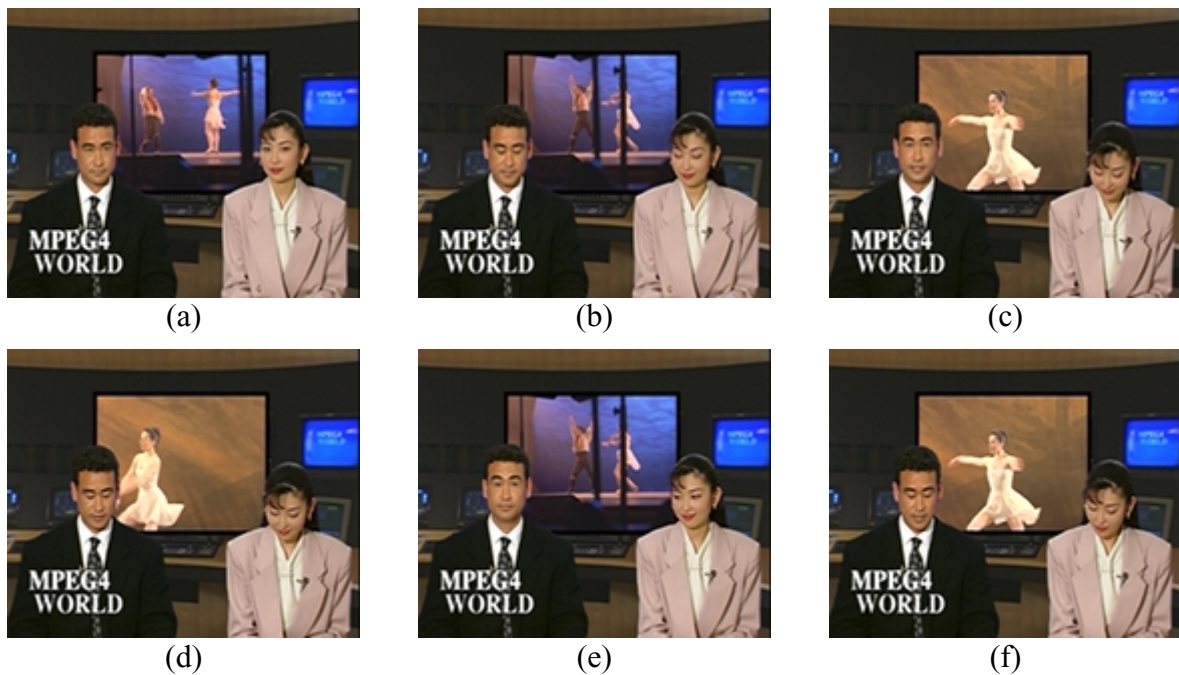


Figure A.17 – News sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A. Test Sequences

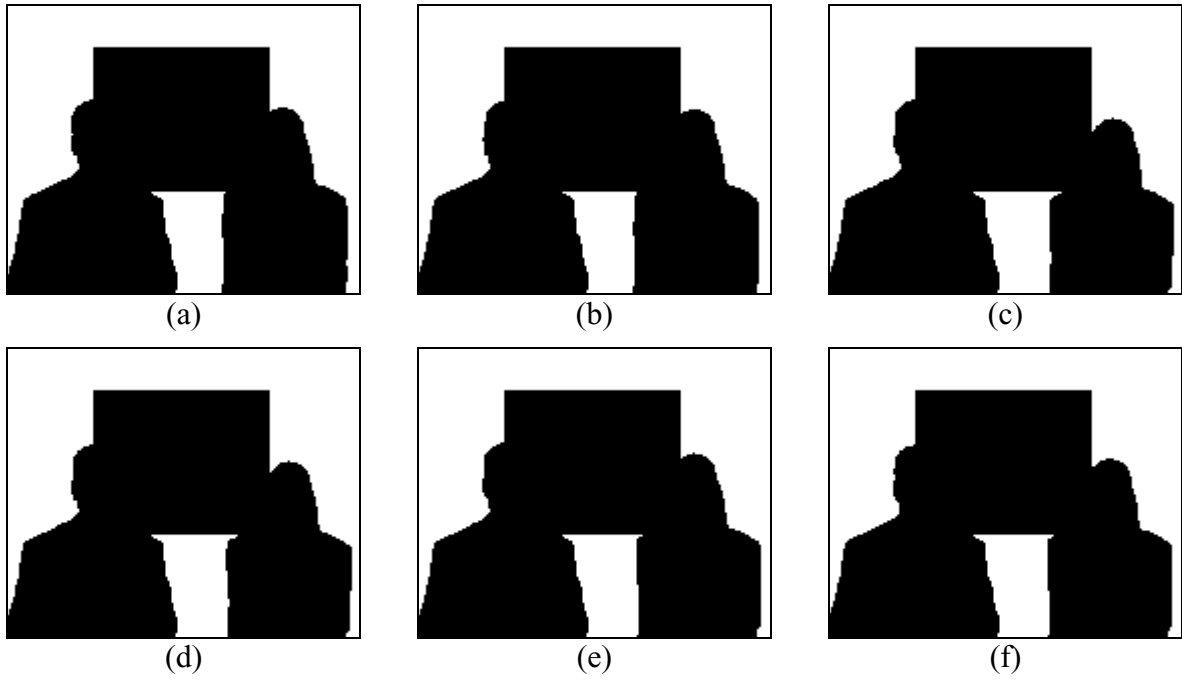


Figure A.18 – News sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

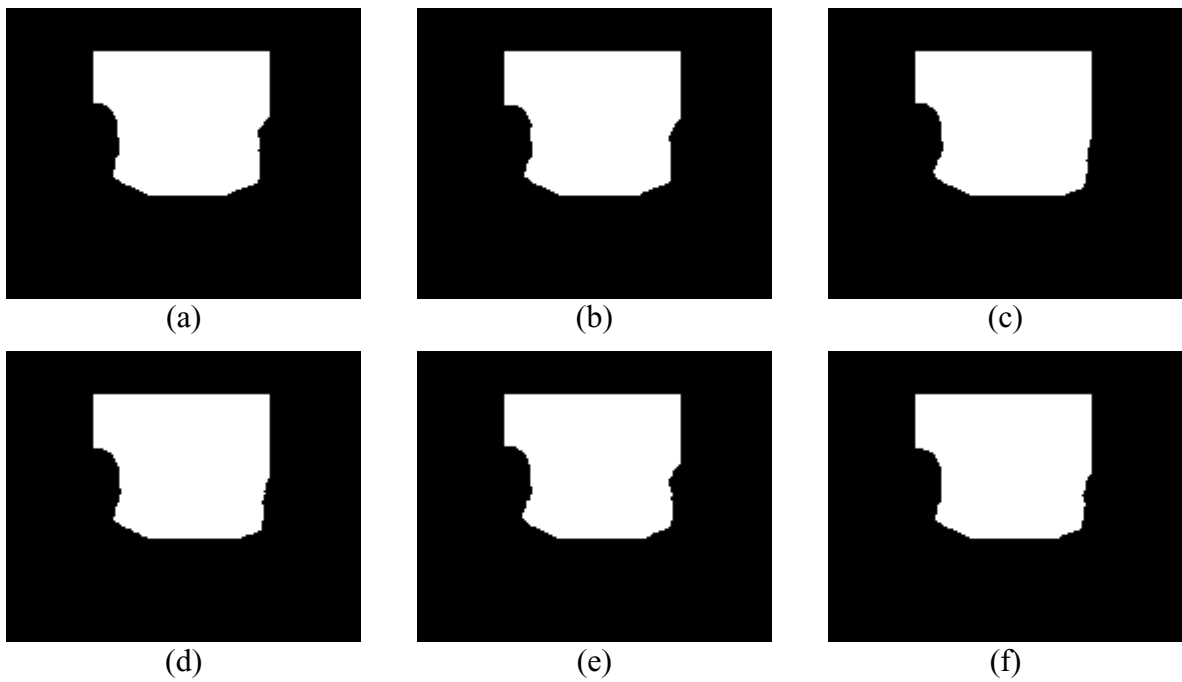


Figure A.19 – News sequence: Dancers alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

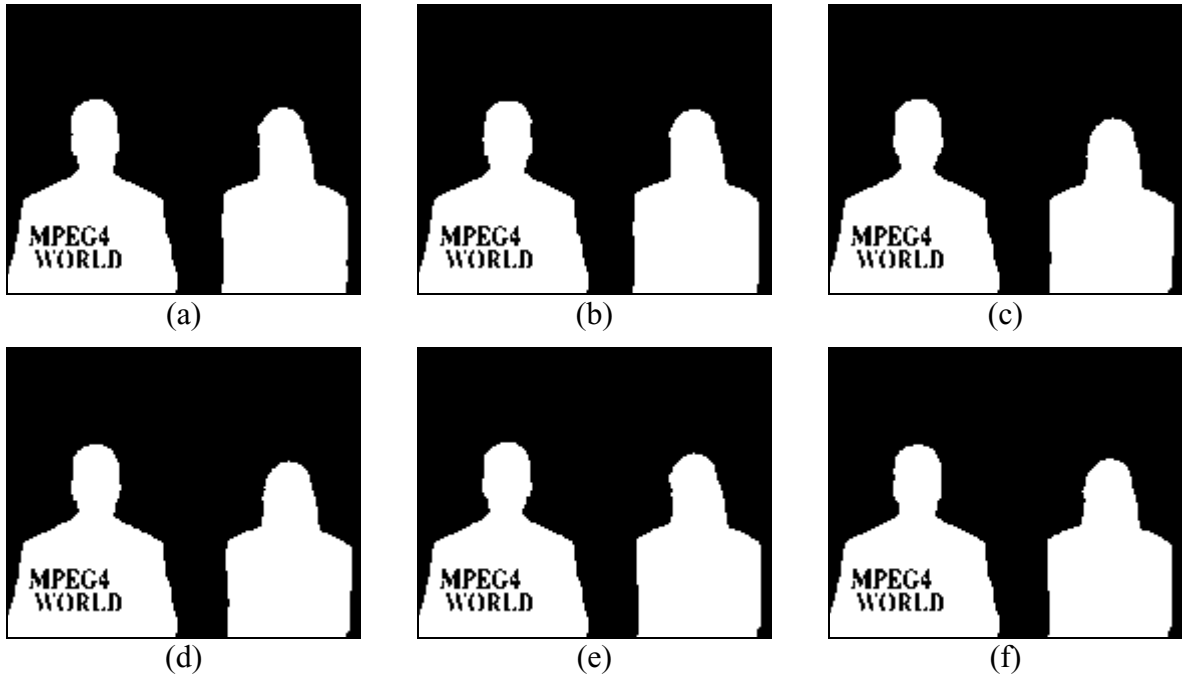


Figure A.20 – News sequence: Newscasters alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

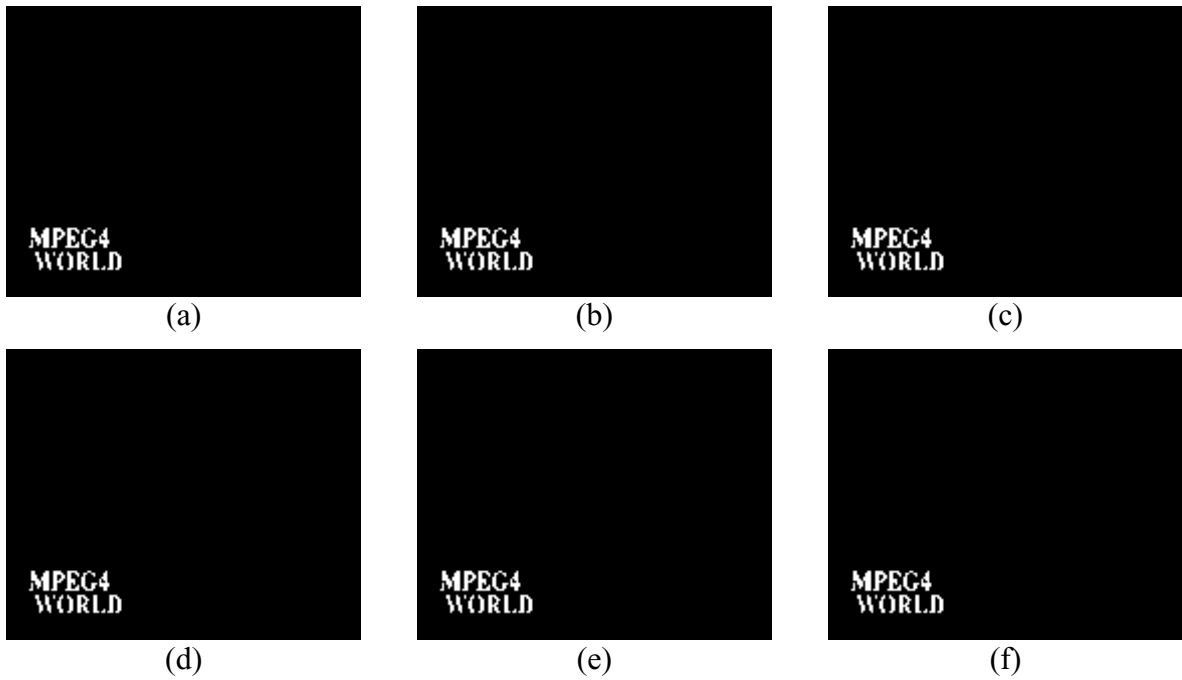


Figure A.21 – News sequence: Logo alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.7 Stefan Sequence

The main characteristics of the Stefan sequence are summarized in Table A.7.

Table A.7 – Main characteristics of the Stefan sequence

Sequence name	Stefan
Available resolutions	QCIF, CIF, SIF
Number of available video objects	2
Video object names	Background, Tennis Player
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

The Stefan sequence is the fastest sequence in the MPEG-4 test set and focuses on a tennis player moving in all directions; behind the tennis player is a crowd. Some sample frames from this sequence are given in Figure A.22, with a temporal spacing of 50 frames. The two available alpha plane sequences correspond to the Background and Tennis Player objects, for which sample frames are given in Figure A.23 and Figure A.24, respectively.

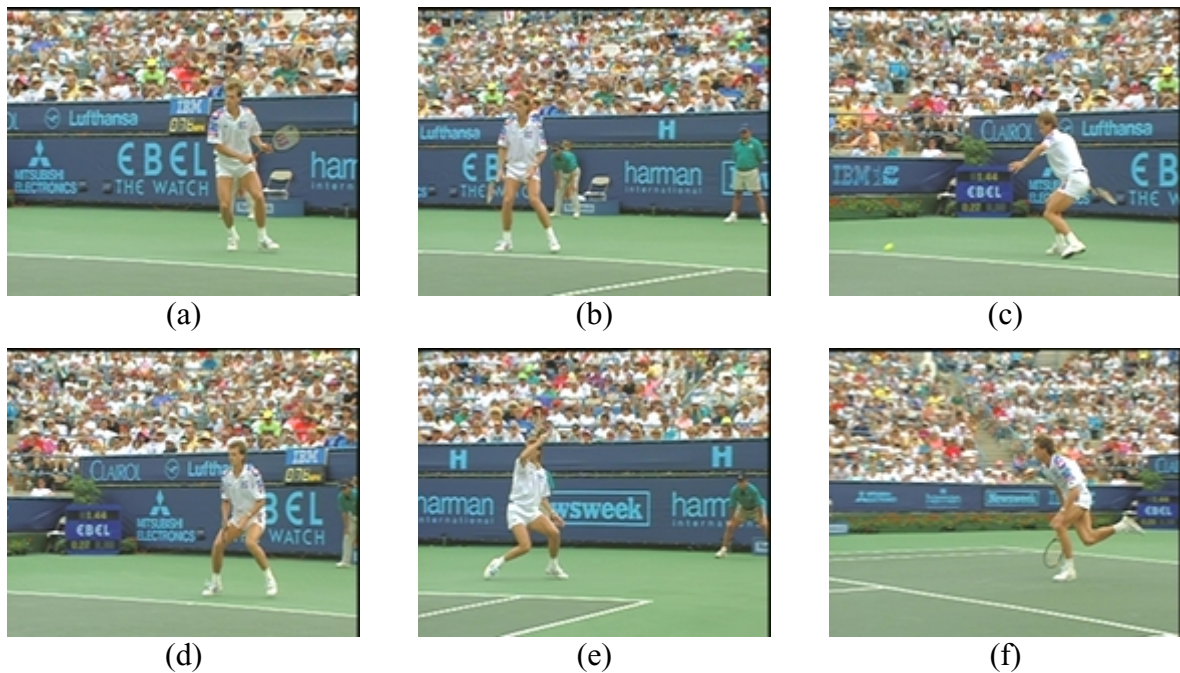


Figure A.22 – Stefan sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

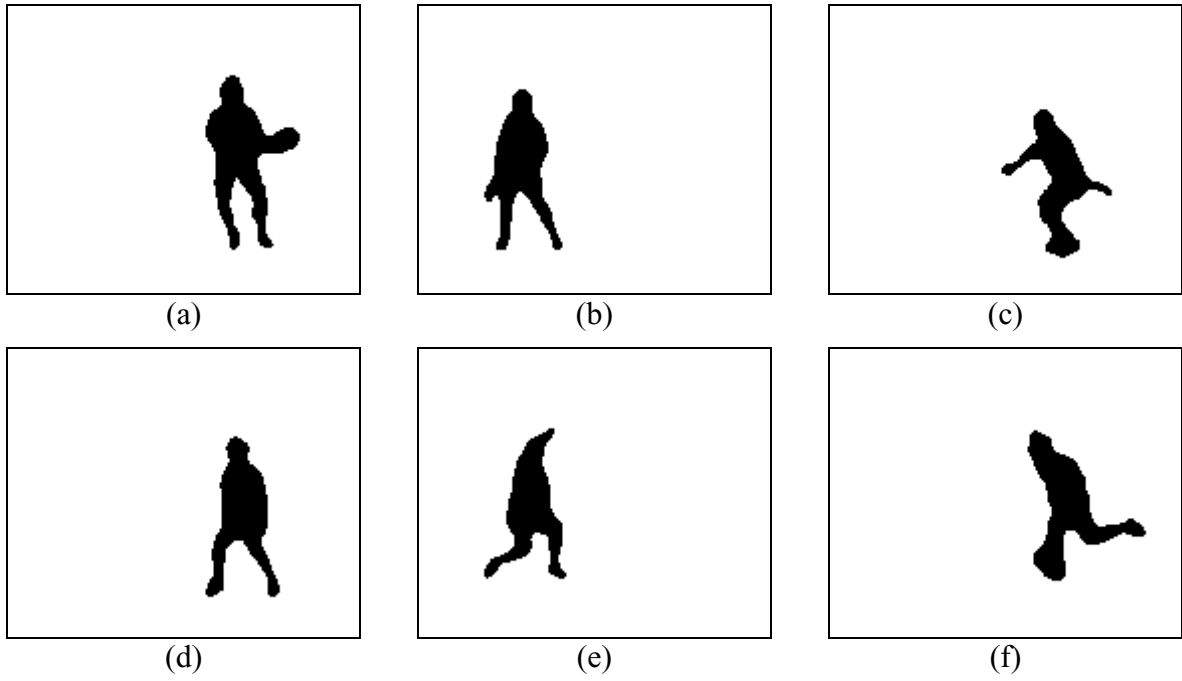


Figure A.23 – Stefan sequence: Background alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

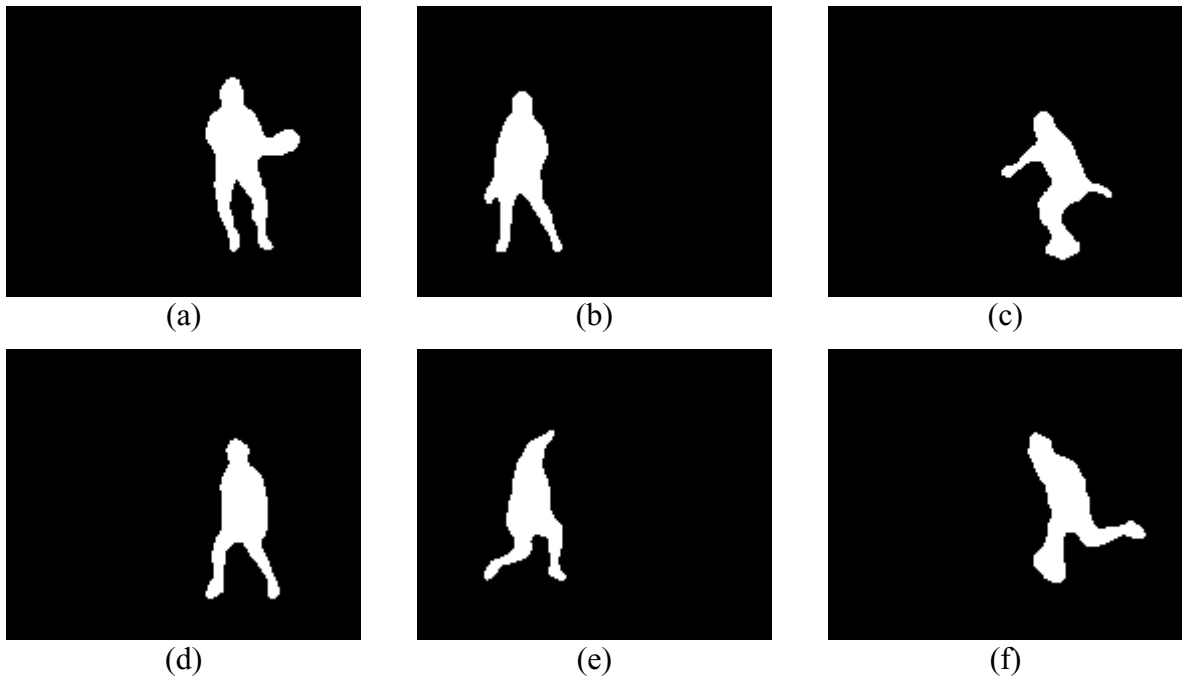


Figure A.24 – Stefan sequence: Tennis Player alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

A.8 Weather Sequence

The main characteristics of the Weather sequence are summarized in Table A.8.

Table A.8 – Main characteristics of the Weather sequence

Sequence name	Weather
Available resolutions	QCIF, CIF
Number of available video objects	2
Video object names	Weather Map, Weather Girl
Duration	10 s
Frame rate	30 fps
Video object generation method	Video segmentation

The Weather sequence is a very slow sequence with a girl presenting the weather forecast, in front of a static synthetic weather map. Some sample frames from this sequence are given in Figure A.25, with a temporal spacing of 50 frames. The two available alpha plane sequences correspond to the Weather Map and Weather Girl objects, for which sample frames are given in Figure A.26 and Figure A.27, respectively.

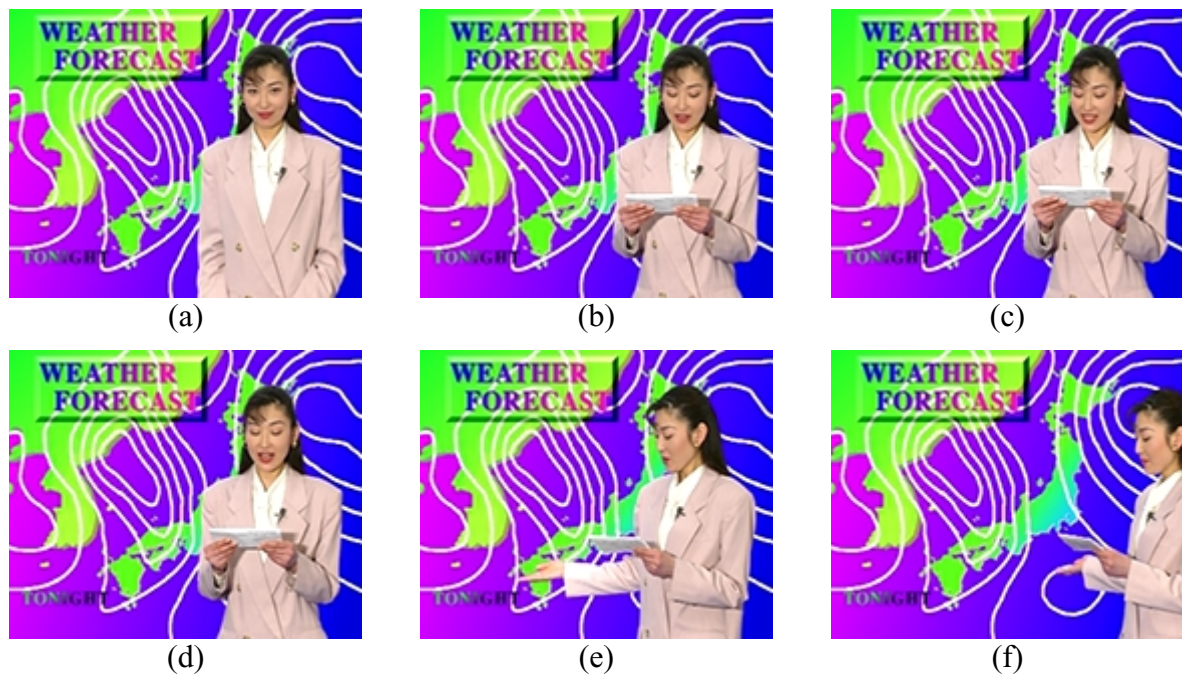


Figure A.25 – Weather sequence: (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

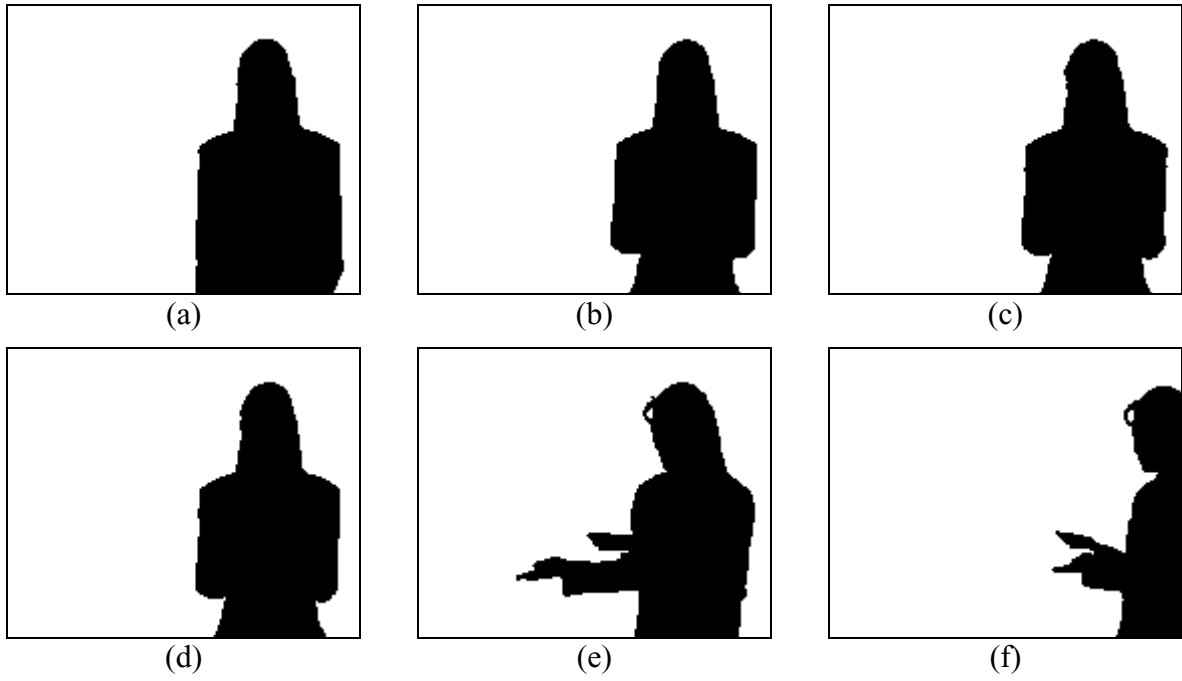


Figure A.26 – Weather sequence: Weather Map alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

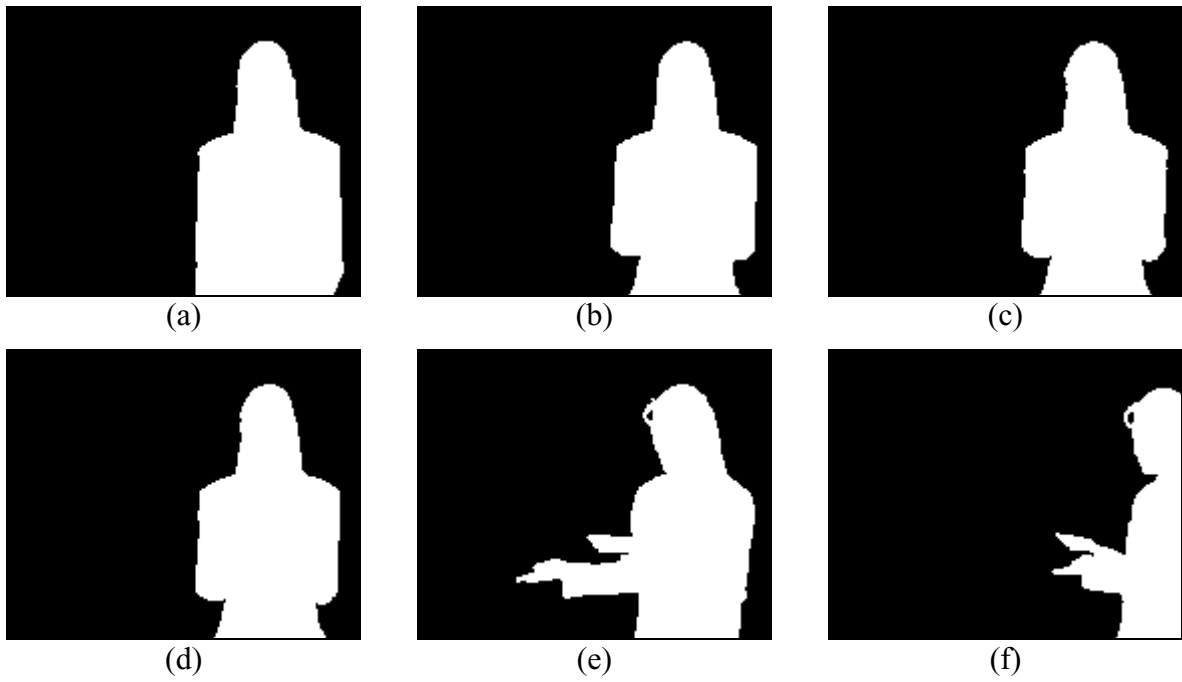


Figure A.27 – Weather sequence: Weather Girl alpha plane – (a) frame 0; (b) frame 49; (c) frame 99; (d) frame 149; (e) frame 199; (f) frame 249

References

- [Aign95] S. Aign, K. Fazel, "Temporal & Spatial Error Concealment Techniques for Hierarchical MPEG-2 Video Codec," *Proc. of the IEEE Global Telecommunications Conference (Globecom'95)*, pp. 1778-1783, Singapore, November 1995.
- [Aravind96] R. Aravind, M. R. Civanlar, A. R. Reibman, "Packet Loss Resilience of MPEG-2 Scalable Video Coding Algorithms," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 5, pp. 426-435, October 1996.
- [Atzori98] L. Atzori, F. G. B. De Natale, "Concealment of Visual Effects of Image Transmission Errors by a Sketch-Based Recovery Approach," *Proc. of the IEEE International Conference on Image Processing (ICIP'98)*, Vol. 3, pp. 482-486, Chicago, IL, USA, October 1998.
- [Ayanoglu95] E. Ayanoglu, P. Pancha, A. R. Reibman, "Video Transport in Wireless ATM," *Proc. of the IEEE International Conference on Image Processing (ICIP'95)*, Vol. 3, pp. 400-403, Washington, D.C., USA, October 1995.
- [Batllo97] J.-C. Batllo, V. A. Vaishampayan, "Asymptotic Performance of Multiple Description Transform Codes," *IEEE Transactions on Information Theory*, Vol. 43, No. 2, pp. 703-707, March 1997.
- [BBC] Web page of the British Broadcasting Company (BBC), <http://www.bbc.co.uk>.
- [Bierling88] M. Bierling, R. Thoma, "Motion Compensating Field Interpolation Using a Hierarchical Structure Displacement Estimator," *Signal Processing*, Vol. 11, No. 4, pp. 387-404, December 1988.
- [Bober99] M. Bober, "Proposal of a CSS-Based Shape Descriptor," ISO/IEC JTC1/SC29/WG11 P320, Lancaster, UK, February 1999.

- [Brady97] N. Brady, L. D. Soares, "Error Resilience of Arbitrarily Shaped VOs (CE E14)," ISO/IEC JTC1/SC29/WG11 M2370, Stockholm MPEG meeting, July 1997.
- [Brady99] N. Brady, "MPEG-4 Standardized Methods for the Compression of Arbitrarily Shaped Video Objects," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 8, December 1999.
- [BT.470] ITU-R Recommendation BT.470-4, "Television Systems," 1995.
- [BT.500] ITU-R Recommendation BT.500-7, "Methodology for the Subjective Assessment of the Quality of Television Pictures," 1995.
- [BT.601] ITU-R Recommendation BT.601-1, "Encoding Parameters of Digital Television for Studios," 1986.
- [Budagavi00] M. Budagavi, W. R. Heinzelman, J. Webb, R. Talluri, "Wireless MPEG-4 Video Communication on DSP Chips," *IEEE Signal Processing Magazine*, Vol. 17, No. 1, pp. 36-53, January 2000.
- [Carlson02] A. B. Carlson, P. B. Crilly, J. C. Rutledge, *Communication Systems: An Introduction to Signals and Noise in Electrical Communication*, 4th Ed., McGraw-Hill, 2002.
- [Carlsson88] S. Carlsson, "Sketch Based Coding of Grey Level Images," *Signal Processing*, Vol. 15, No. 1, pp. 57-83, July 1988.
- [Casio] Web page of the Casio Computer Company, <http://www.casio.com>.
- [Chen97] M.-J. Chen, L.-G. Chen, R.-M. Weng, "Error Concealment of Lost Motion Vectors With Overlapped Motion Compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 3, pp. 560-563, June 1997.
- [Cheng92] N.-T. Cheng, N. G. Kingsbury, "The ERPC: An Efficient Error-Resilient Technique for Encoding Positional Information or Sparse Data," *IEEE Transactions on Communications*, Vol. 40, No. 1, pp. 140-148, January 1992.
- [Clarion] Web page of Clarion, <http://www.clarion.com>.
- [Connectix] Web page of the Connectix Corporation, <http://www.connectix.com>.
- [Correia00] P. Correia, F. Pereira, "Estimation of Video Object's Relevance," *Proc. of the X European Signal Processing Conference (EUSIPCO'2000)*, pp. 925-928, Tampere, Finland, September 2000.
- [Correia02] P. Correia, "Video Analysis for Object-based Coding and Description," *Ph.D. Thesis*, Instituto Superior Técnico, Portugal, 2002.
- [CSE] Web page of the Stanford University's Computer Science Education, <http://www-cse.stanford.edu>.
- [Cumani91] A. Cumani, P. Grattoni, A. Guiducci, "An Edge-Based Description of Color Images," *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, Vol. 53, No. 4, pp. 313-323, July 1991.

References

- [D023] VideoLocus Inc., “AVC Real-Time SD Encoder Demo,” JVT-D023, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), Klagenfurt, Austria, July 2002.
- [ErrorGen] NTT DoCoMo, Error Generating Software, provided to the MPEG Error Resilience Ad Hoc Group Reflector on 31st October 1996.
- [Ferguson84] T. J. Ferguson, J. H. Rabinowitz, “Self-synchronizing Huffman Codes,” *IEEE Transactions on Information Theory*, Vol. IT-30, No. 4, pp. 687-693, July 1984.
- [Foley94] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips, *Introduction to Computer Graphics*, Addison-Wesley Publishing Company, Inc., 1994.
- [Frater98] M. R. Frater, W. S. Lee, M. Pickering, J. F. Arnold, “Error Concealment of Arbitrarily Shaped Video Objects,” *Proc. of the IEEE International Conference on Image Processing (ICIP’98)*, Vol. 3, pp. 507-511, Chicago, IL, USA, October 1998.
- [Gamal82] A. A. El Gamal, T. M. Cover, “Achievable Rates for Multiple Descriptions,” *IEEE Transactions on Information Theory*, Vol. IT-28, No. 6, pp. 851–857, November 1982.
- [Ghanbari89] M. Ghanbari, “Two-Layer Coding of Video Signals for VBR Networks,” *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 5, pp. 771–781, June 1989.
- [Ghanbari93] M. Ghanbari, V. Seferidis, “Cell-Loss Concealment in ATM Video Codecs,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 3, pp. 238-247, June 1993.
- [Ghanbari96] M. Ghanbari, “Postprocessing of Late Cells for Packet Video,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 6, pp. 669-678, December 1996.
- [Gilbert60] E. N. Gilbert, “Capacity of a Burst-Noise Channel,” *The Bell System Technical Journal*, Vol. 39, No. 9, pp. 1253-1265, September 1960.
- [Glassner94] A. S. Glassner, Ed., *Graphics Gems*, Academic Press, January 1994.
- [Graham67] D. N. Graham, “Image Transmission by Two-Dimensional Contour Coding,” *Proceedings of the IEEE*, Vol. 55, No. 3, pp. 336–346, March 1967.
- [Grattoni90] P. Grattoni, A. Guiducci, “Contour Coding for Image Description,” *Pattern Recognition Letters*, Vol. 11, No. 2, pp. 95–105, February 1990.
- [H.223/A] ITU-T Recommendation H.223/Annex A, “Multiplexing Protocol for Low Bitrate Mobile Multimedia Communication,” 1996.
- [H.261] ITU-T Recommendation H.261, “Video Codec for Audiovisual Services at p×64 kbps,” 1993.
- [H.263] ITU-T Recommendation H.263, “Video Coding for Low Bitrate Communication,” 1998.

- [Haralick92] R. M. Haralick, L. G. Shapiro, *Computer and Robot Vision*, Vol. 1, Addison-Wesley, 1992.
- [Haskell90] P. Haskell, D. Messerschmitt, "Reconstructing Lost Video Data in a Lapped Orthogonal Transform Based Coder," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'88)*, pp. 1985-1988, Albuquerque, NM, USA, April 1990.
- [Haskell92] P. Haskell, D. Messerschmitt, "Resynchronization of Motion Compensated Video Affected by ATM Cell Loss," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'92)*, Vol. 3, pp. 545-548, San Francisco, CA, USA, March 1992.
- [Haskell97] B. G. Haskell, A. Puri, A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman & Hall, 1997.
- [Heinzelman99] W. R. Heinzelman, M. Budagavi, R. Talluri, "Unequal Error Protection of MPEG-4 Compressed Video," *Proc. of the IEEE International Conference on Image Processing (ICIP'99)*, Vol. 2, pp. 530-534, Kobe, Japan, October 1999.
- [Hemami94] S. S. Hemami, R. M. Gray, "Image Reconstruction Using Vector Quantized Linear Interpolation," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'94)*, Vol. V, pp. 629-632, Adelaide, Australia, May 1994.
- [Hemami95] S. S. Hemami, T. H.-Y. Meng, "Transform Coded Image Reconstruction Exploiting Interblock Correlation," *IEEE Transactions on Image Processing*, Vol. 4, No. 7, pp. 1023-1027, July 1995.
- [Hemami96] S. S. Hemami, "Reconstruction-Optimized Lapped Orthogonal Transforms for Robust Image Transmission," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 2, pp. 168-181, April 1996.
- [Hemami97] S. S. Hemami, R. M. Gray, "Subband-Coded Image Reconstruction for Lossy Packet Networks," *IEEE Transactions on Image Processing*, Vol. 6, No. 4, pp. 523-539, April 1997.
- [Hong99] M.-C. Hong, H. Schwab, L. P. Kondi, A. K. Katsaggelos, "Error Concealment Algorithms for Compressed Video," *Signal Processing: Image Communication*, Vol. 14, Nos. 6-8, pp. 473-492, May 1999.
- [Horne93] C. Horne, A. R. Reibman, "Adaptation to Cell Loss in a 2-Layer Video Codec for ATM Networks," *Proc. of the Picture Coding Symposium (PCS'93)*, Lausanne, Switzerland, March 1993.
- [iWL] Web page of iWatchLive.Com, <http://www.iwatchlive.com>.
- [Katsaggelos98] A. K. Katsaggelos, N. P. Galatsanos, Eds., *Signal Recovery Techniques for Image and Video Compression and Transmission*, Kluwer Academic Publishers, 1998.

References

- [Kawahara96] T. Kawahara, S. Adachi, "Video Transmission Technology With Effective Error Protection and Tough Synchronization for Wireless Channels," *Proc. of the IEEE International Conference on Image Processing (ICIP'96)*, Vol. 2, pp. 101–104, Lausanne, Switzerland, November 1996.
- [Khansari95] M. Khansari, M. Vetterli, "Layered Transmission of Signals over Power-Constrained Wireless Channels," *Proc. of the IEEE International Conference on Image Processing (ICIP'95)*, Vol. 3, pp. 380–383, Washington, D.C., USA, October 1995.
- [Khansari96] M. Khansari, A. Jalali, E. Dubois, P. Mermelstein, "Low Bit-Rate Video Transmission over Fading Channels for Wireless Microcellular Systems," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 1, pp. 1–11, February 1996.
- [Kieu94] L. H. Kieu, K. N. Ngan, "Cell-Loss Concealment Techniques for Layered Video Codecs in an ATM Network," *IEEE Transactions on Image Processing*, Vol. 3, No. 5, pp. 666–677, September 1994.
- [Kim00] W. Kim, Y. Kim, "A Region-based Shape Descriptor using Zernike Moments," *Signal Processing: Image Communication*, Vol. 16, Nos. 1-2, pp. 95-102, September 2000.
- [Kim99] W. Kim, Y. Kim, "A Rotation Invariant Geometric Shape Descriptor using Zernike Moments," ISO/IEC JTC1/SC29/WG11 P687, Lancaster, UK, February 1999.
- [Kim99a] W. Kim, Y. Kim, "A New Region-based Shape Descriptor: The ART (Angular Radial Transform) Descriptor," ISO/IEC JTC1/SC29/WG11 M5472, Maui MPEG meeting, December 1999.
- [Kwok93] W. Kwok, H. Sun, "Multi-Directional Interpolation for Spatial Error Concealment," *IEEE Transactions on Consumer Electronics*, Vol. 39, No. 3, pp. 455-460, August 1993.
- [Lam92] W.-M. Lam, A. R. Reibman, "Self-Synchronizing Variable-Length Codes for Image Transmission," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'92)*, Vol. 3, pp. 477–480, San Francisco, CA, USA, March 1992.
- [Lam93] W.-M. Lam, A. R. Reibman, B. Liu, "Recovery of Lost or Erroneously Received Motion Vectors," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'93)*, Vol. 5, pp. 417-420, Minneapolis, MN, USA, April 1993.
- [Lee95] X. Lee, Y.-Q. Zhang, A. Leon-Garcia, "Information Loss Recovery for Block-based Image Coding Techniques – A Fuzzy Logic Approach," *IEEE Transactions on Image Processing*, Vol. 4, No. 3, pp. 259-273, March 1995.
- [LeGall91] D. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No. 4, pp. 46-58, April 1991.

- [Lei91] S.-M. Lei, M.-T. Sun, "An Entropy Coding System for Digital HDTV Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 1, No. 1, pp. 147-155, March 1991.
- [Li02] X. Li, A. K. Katsaggelos, G. M. Schuster, "A Recursive Shape Error Concealment Algorithm," *Proc. of the IEEE International Conference on Image Processing (ICIP'2002)*, Vol. 1, pp. 177-180, Rochester, NY, USA, September 2002.
- [Li02a] X. Li, G. M. Schuster, A. K. Katsaggelos, "Curve-fitting Algorithms for Shape Error Concealment," *Proc. of the Nordic Signal Processing Symposium (NORSIG'2002)*, pp. 4-7, Hurtigruten, Norway, October 2002.
- [Liao00] J. Y. Liao, J. D. Villasenor, "Adaptive Intra Block Update for Robust Transmission of H.263," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pp. 30-35, February 2000.
- [Liao96] J. Y. Liao, J. D. Villasenor, "Adaptive Intra Update for Video Coding over Noisy Channels," *Proc. of the IEEE International Conference on Image Processing (ICIP'96)*, Vol. 3, pp. 763-766, Lausanne, Switzerland, September 1996.
- [Malvar89] H. S. Malvar, D. H. Staelin, "The LOT: Transform Coding Without Blocking Effects," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 4, pp. 553-559, April 1989.
- [Manjunath02] B. S. Manjunath, P. Salembier, T. Sikora, Eds., *Introduction to MPEG-7*, Wiley, 2002.
- [Marr82] D. Marr, *Vision*, W. H. Freeman & Company, New York, 1982.
- [Maxted85] J. C. Maxted, J. P. Robinson, "Error Recovery for Variable Length Codes," *IEEE Transactions on Information Theory*, Vol. IT-31, No. 6, pp. 794-801, November 1985.
- [Mish95] F. C. Mish, Ed., *Merriam-Webster's Collegiate Dictionary*, 10th Ed., Merriam-Webster, Incorporated, Springfield, MA, 1995.
- [Modestino79] J. W. Modestino, D. G. Daut, "Combined Source-Channel Coding of Images," *IEEE Transactions on Communications*, Vol. COM-27, No. 11, pp. 1644-1659, Nov. 1979.
- [Modestino81] J. W. Modestino, D. G. Daut, A. L. Vickers, "Combined Source-Channel Coding of Images Using the Block Cosine Transform," *IEEE Transactions on Communications*, Vol. COM-29, No. 9, pp. 1261-1274, Sept. 1981.
- [Mokhtarian96] F. Mokhtarian, S. Abbasi, J. Kittler, "Efficient and Robust Retrieval by Shape Content through Curvature Scale Space," *Proc. of the International Workshop on Image Databases and Multimedia Search*, pp. 35-42, Amsterdam, The Netherlands, August 1996.
- [Mokhtarian96a] F. Mokhtarian, S. Abbasi, J. Kittler, "Robust and Efficient Shape Indexing through Curvature Scale Space," *Proc. of the British Machine Vision Conference*, pp. 53-62, Edinburgh, UK, September 1996.

References

- [Mokhtarian99] F. Mokhtarian, S. Abbasi, "Shape-Based Indexing using Curvature Scale Space with Affine Curvature," *Proc. of the European Workshop on Content-Based Multimedia Indexing (CBMI'99)*, pp. 255-262, Toulouse, France, October 1999.
- [MPEG1] ISO/IEC 11172, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbps," 1993.
- [MPEG2] ISO/IEC 13818, "Information Technology – Generic Coding of Moving Pictures and Associated Audio Information," 1996.
- [MPEG4] ISO/IEC 14496, "Information Technology – Coding of Audio-Visual Objects," 1999.
- [MPEG4-S] ISO/IEC 14496-1, "Information Technology – Coding of Audio-Visual Objects, Part 1: Systems," 1999.
- [MPEG4-SW] ISO/IEC 14496-5, "Information Technology – Coding of Audio-Visual Objects, Part 5: Reference Software," 2000.
- [MPEG4-V] ISO/IEC 14496-2, "Information Technology – Coding of Audio-Visual Objects, Part 2: Visual," 1999.
- [MPEG7-SW] MPEG, "Text of ISO/IEC FCD 15938-6 Information Technology - Multimedia Content Description Interface - Part 6: Reference Software," ISO/IEC JTC1/SC29/WG11 N4006, Singapore MPEG meeting, March 2001.
- [MPEG7-V] MPEG, "Text of ISO/IEC FCD 15938-3 Information Technology - Multimedia Content Description Interface - Part 3: Visual," ISO/IEC JTC1/SC29/WG11 N4003, Singapore MPEG meeting, March 2001.
- [Mukawa84] N. Mukawa, H. Kuroda, T. Matuoka, "An Interframe Coding System for Video Teleconferencing Signal Transmission at a 1.5 Mbit/s Rate," *IEEE Transactions on Communications*, Vol. COM-32, No. 3, pp. 280-287, March 1984.
- [N1471] Video Subgroup, "Core Experiments on MPEG-4 Video Shape Coding," ISO/IEC JTC1/SC29/WG11 N1471, Seville MPEG meeting, February 1997.
- [N1584] MPEG Ad Hoc Group on Shape Coding, "Description of Core Experiments on MPEG-4 Video Shape Coding," ISO/IEC JTC1/SC29/WG11 N1584, Seville MPEG meeting, February 1997.
- [N1586] MPEG Ad Hoc Group on Error Resilience, "Description of Error Resilient Core Experiments," ISO/IEC JTC1/SC29/WG11 N1586, Seville MPEG meeting, February 1997.
- [N1646] MPEG Ad Hoc Group on Error Resilience, "Description of Error Resilient Core Experiments," ISO/IEC JTC1/WG11/SC29 N1646, Bristol MPEG meeting, April 1997.
- [N2061] Test and Video Subgroups, "Workplan for Formal Verification Tests on Video Error Resilience," ISO/IEC JTC1/SC29/WG11 N2061, San Jose MPEG meeting, February 1998.

- [N2165] Test and Video Subgroups, “Revised Work Plan for Formal Verification Tests on Video Error Resilience,” ISO/IEC JTC1/SC29/WG11 N2165, Tokyo MPEG meeting, March 1998.
- [N2604] Test Subgroup, “Report of the Formal Verification Tests on MPEG-4 Video Error Resilience,” ISO/IEC JTC1/SC29/WG11 N2604, Rome MPEG meeting, December 1998.
- [Neumann71] P. G. Neumann, “Self-synchronizing Sequential Coding with Low Redundancy,” *The Bell System Technical Journal*, Vol. 50, No. 3, pp. 951-981, March 1971.
- [News14] Web page of the cable channel News 14 Carolina, <http://news14.com>.
- [Nokia] Web page of the Nokia Corporation, <http://www.nokia.com>.
- [Nomura91] M. Nomura, T. Fujii, N. Ohta, “Layered Coding for ATM Based Video Distribution Systems,” *Signal Processing: Image Communication*, Vol. 3, No. 4, pp. 301–311, September 1991.
- [Orchard97] M. T. Orchard, Y. Wang, V. Vaishampayan, A. R. Reibman, “Reduced Rate-Distortion Analysis of Multiple Description Coding Using Pairwise Correlating Transforms,” *Proc. of the IEEE International Conference on Image Processing (ICIP’97)*, Vol. 1, pp. 608–611, Santa Barbara, CA, USA, October 1997.
- [Ozarow80] L. Ozarow, “On a Source Coding Problem with Two Channels and Three Receivers,” *The Bell System Technical Journal*, Vol. 59, No. 10, pp. 1909–1921, December 1980.
- [P.910] ITU-T Recommendation P.910, “Subjective Video Quality Assessment Methods for Multimedia Applications,” 1996.
- [Papoulis91] A. Papoulis, *Probability, Random Variables and Stochastic Processes*, 3rd Ed., McGraw-Hill, 1991.
- [Park97] J. W. Park, J. W. Kim, S. U. Lee, “DCT Coefficients Recovery-Based Error Concealment Technique and Its Application to the MPEG-2 Bit Stream Error,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 6, pp. 845-854, December 1997.
- [Pennebaker92] W. B. Pennebaker, J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand, New York, 1992.
- [Pereira02] F. Pereira, T. Ebrahimi, Eds., *The MPEG-4 Book*, Prentice-Hall, 2002.
- [Pereira97] F. Pereira, T. Alpert, “MPEG-4 Video Subjective Test Procedures and Results,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 1, pp. 32-51, February 1997.
- [Philips] Web page of the Royal Philips Electronics Research Division, <http://www.research.philips.com>.
- [Press92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C – The Art of Scientific Computing*, 2nd Ed., Cambridge University Press, 1992.

References

- [Rabiner89] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286, February 1989.
- [Ramchandran93] K. Ramchandran, A. Ortega, K. M. Uz, M. Vetterli, "Multiresolution Broadcast for Digital HDTV Using Joint Source/Channel Coding," *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 1, pp. 6-23, January 1993.
- [Redl95] S. M. Redl, M. K. Weber, M. W. Oliphant, *An Introduction to GSM*, Artech House, Boston, MA, 1995.
- [Redmill96] D. W. Redmill, N. G. Kingsbury, "The EREC: An Error-Resilient Technique for Coding Variable-Length Blocks of Data," *IEEE Transactions on Image Processing*, Vol. 5, No. 4, pp. 565-574, April 1996.
- [Reimers01] U. Reimers, *Digital Video Broadcasting: The International Standard for Digital Television*, Springer, 2001.
- [Riley96] M. J. Riley, I. E. G. Richardson, *Digital Video Communications*, Artech House, 1996.
- [Rosiles95] G. Gonzalez-Rosiles, S. D. Cabrera, S. W. Wu, "Recovery With Lost Subband Data in Overcomplete Image Coding," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'95)*, Vol. II, pp. 1476-1479, Detroit, MI, USA, May 1995.
- [Salama02] P. Salama, C. Huang, "Error Concealment for Shape Coding," *Proc. of the IEEE International Conference on Image Processing (ICIP'2002)*, Vol. 2, pp. 701-704, Rochester, NY, USA, September 2002.
- [Schafer03] R. Schäfer, T. Wiegand, H. Schwarz, "The Emerging H.264/AVC Standard," *EBU Technical Review*, January 2003.
- [Schuster03] G. M. Schuster, X. Li, A. K. Katsaggelos, "Spline Based Boundary Loss Concealment," *Proc. of the IEEE International Conference on Image Processing (ICIP'2003)*, Vol. 2, pp. 671-674, Barcelona, Spain, September 2003.
- [Schuster04] G. M. Schuster, X. Li, A. K. Katsaggelos, "Shape Error Concealment Using Hermite Splines," *IEEE Transactions on Image Processing*, Vol. 13, No. 6, June 2004.
- [Sequeira97] M. M. de Sequeira, D. Cortez, "Partitions: A Taxonomy of Types and Representations and an Overview of Coding Techniques," *Signal Processing: Image Communication*, Vol. 10, Nos. 1-3, pp. 5-19, July 1997.
- [Servetto00] S. D. Servetto, K. Ramchandran, V. A. Vaishampayan, K. Nahrstedt, "Multiple Description Wavelet Based Image Coding," *IEEE Transactions on Image Processing*, Vol. 9, No. 5, pp. 813-826, May 2000.

- [Sezan82] M. I. Sezan, H. Stark, "Image Restoration by the Method of Convex Projections: Part 2 – Applications and Numerical Results," *IEEE Transactions on Medical Imaging*, Vol. MI-1, No. 2, pp. 95-101, October 1982.
- [Shirani00] S. Shirani, B. Erol, F. Kossentini, "A Concealment Method for Shape Information in MPEG-4 Coded Video Sequences," *IEEE Transactions on Multimedia*, Vol. 2, No. 3, pp. 185-190, September 2000.
- [Soares00] L. D. Soares, S. Adachi, F. Pereira, "Influence of Encoder Parameters on the Decoded Video Quality for MPEG-4 over W-CDMA Mobile Networks," *Proc. of the IEEE International Conference on Image Processing (ICIP'2000)*, Vol. 2, pp. 148-151, Vancouver, Canada, September 2000.
- [Soares01] L. D. Soares, F. Pereira, "MPEG-7 Based Spatial Shape Concealment Difficulty", *Proc. of the 3rd Conference on Telecommunications (ConfTele'2001)*, pp. 433-437, Figueira da Foz, Portugal, April 2001.
- [Soares02] L. D. Soares, F. Pereira, "Texture Refreshment Need Metric for Resilient Object-Based Video Coding," *Proc. of the IEEE International Conference on Multimedia and Expo (ICME'2002)*, Lausanne, Switzerland, August 2002.
- [Soares02a] L. D. Soares, F. Pereira, "Shape Refreshment Need Metric for Object-Based Resilient Video Coding," *Proc. of the IEEE International Conference on Image Processing (ICIP'2002)*, Vol. 1, pp. 173-176, Rochester, NY, USA, September 2002.
- [Soares02b] L. D. Soares, F. Pereira, "Adaptive Shape-Texture Intra Coding Refreshment for Error Resilient Object-Based Video," *Proc. of the IEEE Workshop on Multimedia Signal Processing (MMSP'2002)*, St. Thomas, USVI, December 2002.
- [Soares03] L. D. Soares, F. Pereira, "Refreshment Need Metrics for Improved Shape and Texture Object-Based Resilient Video Coding," *IEEE Transactions on Image Processing*, Vol. 12, No. 3, pp. 328-340, March 2003.
- [Soares04] L. D. Soares, F. Pereira, "Temporal Shape Error Concealment by Global Motion Compensation," *Proc. of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'2004)*, Lisbon, Portugal, April 2004.
- [Soares04a] L. D. Soares, F. Pereira, "Spatial Shape Error Concealment for Object-based Image and Video Coding," *IEEE Transactions on Image Processing*, Vol. 13, No. 4, April 2004.
- [Soares04b] L. D. Soares, F. Pereira, "Adaptive Shape and Texture Intra Refreshment Schemes for Improved Error Resilience in Object-based Video Coding," *IEEE Transactions on Image Processing*, Vol. 13, No. 5, May 2004.
- [Soares04c] L. D. Soares, F. Pereira, "Temporal Shape Error Concealment for Object-based Video", *submitted to the International Symposium on Image/Video Communications over Fixed and Mobile Networks (ISICV'2004)*, Brest, France, July 2004.

References

- [Soares04d] L. D. Soares, F. Pereira, "Motion-based Shape Error Concealment for Object-based Video," *submitted to the IEEE International Conference on Image Processing (ICIP'2004)*, Singapore, October 2004.
- [Soares04e] L. D. Soares, F. Pereira, "Temporal Shape Error Concealment by Global Motion Compensation with Local Refinement," *submitted to IEEE Transactions on Image Processing*.
- [Soares98] L. D. Soares, F. Pereira, "MPEG-4: A Flexible Coding Standard for the Emerging Mobile Multimedia Applications", *Proc. of the International Symposium on Personal, Indoor and, Mobile Radio Communications (PIMRC'98)*, Vol. 3, pp. 1335-1339, Boston, MA, USA, September 1998.
- [Soares98a] L. D. Soares, F. Pereira, "An Alternative to the MPEG-4 Object-Based Error Resilient Video Syntax," *Proc. of the IEEE International Conference on Image Processing (ICIP'98)*, Vol. 3, pp. 467-471, Chicago, IL, USA, October 1998.
- [Soares99] L. D. Soares, F. Pereira, "Error Resilience and Concealment Performance for MPEG-4 Frame-Based Video Coding", *Signal Processing: Image Communication*, Vol. 14, Nos. 6-8, pp. 447-472, May 1999.
- [Soares99a] S. Adachi, L. D. Soares, "A Study on Optimal Parameters for Error Resilience Tools of MPEG-4 Video", *Proc. of the Picture Coding Symposium of Japan 1999 (PCSJ'99)*, pp. 55-56, Japan, September 1999 (in Japanese).
- [SQUID] Web page of the Shape Queries Using Image Databases (SQUID) system, <http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html>
- [Sun92] H. Sun, K. Challapali, J. Zdepski, "Error Concealment in Digital Simulcast AD-HDTV Decoder," *IEEE Transactions on Consumer Electronics*, Vol. 38, No. 3, pp. 108-117, August 1992.
- [Sun95] H. Sun, W. Kwok, "Concealment of Damaged Block Transform Coded Images Using Projections onto Convex Sets," *IEEE Transactions on Image Processing*, Vol. 4, No. 4, pp. 470-477, April 1995.
- [Swann96] R. Swann, N. Kingsbury, "Transcoding of MPEG-II for Enhanced Resilience to Transmission Errors," *Proc. of the IEEE International Conference on Image Processing (ICIP'96)*, Vol. 2, pp. 813-816, Lausanne, Switzerland, November 1996.
- [Takishima95] Y. Takishima, M. Wada, H. Murakami, "Reversible Variable Length Codes," *IEEE Transactions on Communications*, Vol. 43, Nos. 2/3/4, pp. 158-162, February/March/April 1995.
- [Talluri97] R. Talluri, I. Moccagatta, Y. Nag, "E8 – Core Experiment on Error Concealment by Data Partitioning," ISO/IEC JTC1/SC29/WG11 M1622, Seville MPEG meeting, February 1997.
- [Taylor98] J. Taylor, *DVD Demystified*, McGraw-Hill, 1998.
- [TCDesigns] Web page of TC Designs, <http://www.tcchang.com>.

- [Tom91] A. S. Tom, C. L. Yeh, F. Chu, "Packet Video for Cell Loss Protection Using Deinterleaving and Scrambling," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'91)*, pp. 2857–2860, Toronto, Canada, May 1991.
- [TransBus] Web page of the French association for safety in public transportation TransBus, <http://www.transbus.org>.
- [Tsai97] I.-W. Tsai, C.-L. Huang, "Hybrid Cell Loss Concealment Methods for MPEG-II-Based Packet Video," *Signal Processing: Image Communication*, Vol. 9, No. 2, pp. 99-124, January 1997.
- [Tzou89] K. H. Tzou, "Post Filtering for Cell Loss Concealment in Packet Video," *Proc. of the SPIE Conference on Visual Communication and Image Processing (VCIP'89)*, pp. 1620-1627, Philadelphia, PA, USA, November 1989.
- [UNICODE] The Unicode Consortium, *The Unicode Standard, Version 3.0*, Addison-Wesley, 2000.
- [Vaishampayan93] V. A. Vaishampayan, "Design of Multiple Description Scalar Quantizers," *IEEE Transactions on Information Theory*, Vol. 39, No. 3, pp. 821–834, May 1993.
- [Vaishampayan94] V. A. Vaishampayan, J. Domaszewicz, "Design of Entropy-Constrained Multiple-Description Scalar Quantizers," *IEEE Transactions on Information Theory*, Vol. 40, No. 1, pp. 245–250, January 1994.
- [Vaishampayan96] V. A. Vaishampayan, "Application of Multiple Description Codes to Image and Video Transmission over Lossy Networks," *Proc. of the 7th International Workshop on Packet Video*, pp. 55-60, Brisbane, Australia, March 1996.
- [VQEG] The Video Quality Experts Group (VQEG) Web Site, <http://www.vqeg.org>
- [VQEG00] VQEG, "Final Report from the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment," April 2000.
- [Wada89] W. Wada, "Selective Recovery of Video Packet Loss Using Error Concealment," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 5, pp. 807-814, June 1989.
- [Wang00] Y. Wang, S. Wenger, J. Wen, A. K. Katsaggelos, "Error Resilient Video Coding Techniques," *IEEE Signal Processing Magazine*, Vol. 17, No. 4, pp. 61-82, July 2000.
- [Wang01] Y. Wang, M. T. Orchard, V. Vaishampayan, A. R. Reibman, "Multiple Description Coding Using Pairwise Correlating Transforms," *IEEE Transactions on Image Processing*, Vol. 10, No. 3, pp. 351-366, March 2001.
- [Wang91] Y. Wang, V. Ramamoorthy, "Image Reconstruction from Spatial Subband Images and Its Applications in Packet Video Transmission," *Signal Processing*, Vol. 3, Nos. 2-3, pp.197-229, June 1991.

References

- [Wang93] Y. Wang, Q.-F. Zhu, L. Shaw, "Maximally Smooth Image Recovery in Transform Coding," *IEEE Transactions on Communications*, Vol. 41, No. 10, pp. 1544-1551, October 1993.
- [Wang97] Y. Wang, M. T. Orchard, A. R. Reibman, "Multiple Description Image Coding for Noisy Channels by Pairing Transform Coefficients," *Proc. of the IEEE Workshop on MultiMedia Signal Processing (MMSP'97)*, pp. 419-424, Princeton, NJ, USA, June 1997.
- [Wang98] Y. Wang, Q.-F. Zhu, "Error Control and Concealment for Video Communications: A Review," *Proceedings of the IEEE*, Vol. 86, No. 5, pp. 974-997, May 1998.
- [Watanabe97] T. Watanabe, Y. Kikuchi, "Error Resilience for Shape Coding (CE E14)," ISO/IEC JTC1/SC29/WG11 M2642, Fribourg MPEG meeting, October 1997.
- [Weather] Web page of The Weather Channel, <http://www.weather.com>.
- [Wen97] J. Wen, J. D. Villasenor, "A Class of Reversible Variable Length Codes for Robust Image and Video Coding," *Proc. of the IEEE International Conference on Image Processing (ICIP'97)*, Vol. 2, pp. 65-68, Santa Barbara, CA, USA, October 1997.
- [Wenger97] S. Wenger, "Video Redundancy Coding in H.263+," *Proc. of the International Workshop on Audio-Visual Services over Packet Networks (AVSPN'97)*, Aberdeen, UK, September 1997.
- [Wolf80] J. K. Wolf, A. Wyner, J. Ziv, "Source Coding for Multiple Descriptions," *The Bell System Technical Journal*, Vol. 59, No. 8, pp. 1417-1426, October 1980.
- [Wollborn98] M. Wollborn, R. Mech, "Refined Procedure for Objective Evaluation of Video Object Generation Algorithms," ISO/IEC JTC1/SC29/WG11 M3448, Tokyo MPEG meeting, March 1998.
- [Wong78] W. C. Wong, R. Steele, "Partial Correction of Transmission Errors in Walsh Transform Image Coding Without Recourse to Error Correction Coding," *Electronic Letters*, Vol. 14, No. 10, pp. 298-300, May 1978.
- [Yasuda77] H. Yasuda, H. Kuroda, H. Kawanishi, F. Kanaya, H. Hashimoto, "Transmitting 4 MHz TV Signals by Combinational Difference Coding," *IEEE Transactions on Communications*, Vol. COM-25, No. 5, pp. 508-516, May 1977.
- [Youla82] D. C. Youla, H. Webb, "Image Restoration by the Method of Convex Projections: Part 1 – Theory," *IEEE Transactions on Medical Imaging*, Vol. MI-1, No. 2, pp. 81-94, October 1982.
- [Yu98] G.-S. Yu, M. M.-K. Liu, M. W. Marcellin, "POCS-Based Error Concealment for Packet Video Using Multiframe Overlap Information," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 4, pp. 422-434, August 1998.
- [Zakhor93] A. Zakhor, F. Lari, "Edge Based 3-D Camera Motion Estimation with Application to Video Coding," *IEEE Transactions on Image Processing*, Vol. 2, No. 4, pp. 481-498, October 1993.

- [Zhang94] Y.-Q. Zhang, Y.-J. Liu, R. L. Pickholtz, “Layered Image Transmission over Cellular Radio Channels,” *IEEE Transactions on Vehicular Technology*, Vol. 43, No. 3, pp. 786–794, August 1994.
- [Zhu93] Q.-F. Zhu, Y. Wang, L. Shaw, “Coding and Cell-Loss Recovery in DCT-Based Packet Video,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 3, pp. 248–258, June 1993.
- [Zhu95] W. Zhu, Y. Wang, “A Comparison of Smoothness Measures for Error Concealment in Transform Coding,” *Proc. of the SPIE Conference on Visual Communication and Image Processing (VCIP'95)*, Vol. II, pp. 1205-1214, Taipei, Taiwan, May 1995.
- [Zhu96] Q.-F. Zhu, “Device and Method of Signal Loss Recovery for Realtime and/or Interactive Communications,” U.S. Patent 5550847, August 1996.