



UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Modelo Alternativo de Complexidade para a Norma MPEG-4 Visual

João Manuel Malveiro Valentim
(licenciado)

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

DOCUMENTO PROVISÓRIO

Agradecimentos

Em primeiro lugar quero agradecer ao Prof. Fernando Pereira pela sua dedicação, acompanhamento e incentivo ao longo do mestrado, estando sempre disponível para ajudar e para oferecer todas as condições necessárias para a sua realização.

A todos os elementos do Grupo de Imagem do Instituto de Telecomunicações pela sua camaradagem e ajuda nos momentos de menor inspiração: Carla Zibreira, João Ascenso, Paula Queluz, Paulo Correia e especialmente ao Luis Ducla Soares e Paulo Nunes que perderam muitas horas comigo de volta do código e a explicar-me detalhes da norma MPEG-4.

Aos meus pais e ao meu irmão, que sempre me apoiaram e motivaram ao longo dos dois anos de duração do mestrado.

A todos os meus amigos que estiveram ao meu lado e me ajudaram sempre que possível.

Por fim agradeço à Fundação para a Ciência e a Tecnologia (FCT) pela bolsa de mestrado que me foi concedida no âmbito do programa PRAXIS XXI.

Modelo Alternativo de Complexidade para a Norma MPEG-4 Visual

João Manuel Malveiro Valentim

Curso de Mestrado em: Engenharia Electrotécnica e de Computadores

Orientador: Prof. Doutor Fernando Manuel Bernardo Pereira

Provas concluídas em:

Resumo

A norma MPEG-4 é a primeira norma de representação audiovisual que modela uma cena como uma composição de objectos audiovisuais independentes. Foi desenvolvida com vista a ser usada num grande número de aplicações que requerem um leque muito abrangente de funcionalidades, tipos de dados, etc. Para garantir a interoperabilidade e limitar a complexidade, é necessário definir normativamente subconjuntos de ferramentas, denominados por perfis, que devem satisfazer as necessidades de certas classes de aplicações. Por sua vez, os níveis especificam restrições às ferramentas incluídas em cada perfil, através de limitações impostas a alguns parâmetros relevantes. Para garantir conformidade, a norma define um mecanismo de verificação de vídeo que deve ser implementado no codificador para controlar que os fluxos binários gerados respeitem os limites de complexidade correspondentes ao perfil@nível escolhido.

Esta tese propõe um modelo alternativo para o mecanismo que verifica a complexidade dos fluxos binários gerados pelo codificador, *Video Complexity Verifier* (VCV), baseado na atribuição, a cada macrobloco codificado, de um peso associado à complexidade de descodificação das várias classes de macroblocos usados na codificação de vídeo MPEG-4. Para melhor alcançar estes objectivos, foi também desenvolvida uma aplicação de edição, codificação e descodificação de vídeo MPEG-4, que oferece grande flexibilidade em termos da criação e composição de cenas de vídeo baseadas em objectos. As cenas criadas podem ser codificadas de acordo com um determinado perfil@nível, oferecendo assim a possibilidade de criar fluxos binários de vídeo MPEG-4 conformes.

Palavras Chave

MPEG-4, Perfis e Níveis, Conformidade, Mecanismo de Verificação de Vídeo, Complexidade de Descodificação, Modelo VCV

An Alternative Complexity Model for the MPEG-4 Visual Standard

João Manuel Malveiro Valentim

Abstract

MPEG-4 is the first audiovisual standard modelling a scene as a composition of independent audiovisual objects. This standard was developed to be used for a large number of applications requiring a wide range of functionalities, data types, etc. To ensure interoperability and limit complexity, subsets of tools called profiles have to be normatively defined, in order to satisfy the needs of some applications classes. On the other hand, levels specify restrictions to the tools included in each profile, by imposing limitations to some relevant parameters. To ensure conformance, the standard defines a video buffering verifier mechanism that must be implemented at the encoder to check that the generated bitstreams respect the complexity limits associated to the selected profile@level.

This thesis proposes an alternative model for the mechanism that verifies the complexity of the bitstreams generated by the encoder, Video Complexity Verifier (VCV), based on the attribution, to each coded macroblock, of a weight associated to the decoding complexity of the various macroblock classes used in MPEG-4 video coding. To reach this objective, an MPEG-4 video editing, coding and decoding application was also developed, offering great flexibility in the creation and composition of object-based video scenes. The scenes created may be coded according to a specified profile@level, offering the possibility to create MPEG-4 compliant video bitstreams.

Keywords

MPEG-4, Profiles and Levels, Conformance, Video Buffering Verifier, Decoding Complexity, VCV Model

Índice

ÍNDICE DE FIGURAS	viii
--------------------------	-------------

ÍNDICE DE TABELAS	xii
--------------------------	------------

LISTA DE ACRÓNIMOS	xiv
---------------------------	------------

CAPÍTULO 1 INTRODUÇÃO	1
------------------------------	----------

CAPÍTULO 2 A NORMA MPEG-4 VISUAL	7
---	----------

2.1	INTRODUÇÃO	7
2.2	FUNCIONALIDADES	10
2.3	ÁREAS DE APLICAÇÃO	14
2.4	ORGANIZAÇÃO DA NORMA MPEG-4	15
2.5	ESTRUTURA E SINTAXE DA NORMA MPEG-4 VÍDEO	16
2.6	FERRAMENTAS DE CODIFICAÇÃO DE VÍDEO	20
2.6.1	<i>Codificação de forma</i>	21
2.6.2	<i>Estimação e compensação de movimento</i>	22
2.6.3	<i>Codificação de textura</i>	23
2.6.4	<i>Sprites</i>	24
2.6.5	<i>Codificação de texturas estáticas</i>	25
2.6.6	<i>Resiliência a erros</i>	25
2.6.7	<i>Escalabilidade</i>	27
2.7	TIPOS DE OBJECTO, PERFIS E NÍVEIS	29
2.7.1	<i>Tipos de objecto visuais</i>	31
2.7.2	<i>Perfis visuais</i>	33
2.8	MECANISMO DE VERIFICAÇÃO DE VÍDEO	36
2.8.1	<i>Interacção entre VBV, VCV e VMV</i>	38
2.9	CONCLUSÕES	41

CAPÍTULO 3 VÍDEO MPEG-4: UMA APLICAÇÃO EM CONFORMIDADE	43
---	-----------

3.1	INTRODUÇÃO	43
3.2	EDITOR DE CENA	44
3.2.1	<i>Criação de uma nova cena</i>	45
3.2.2	<i>Criação de um novo objecto de vídeo</i>	46

3.2.3	<i>Adição de objectos de vídeo à cena</i>	48
3.2.4	<i>Edição da cena</i>	49
3.2.5	<i>Gravação de objectos de vídeo e cenas</i>	52
3.2.6	<i>Abertura de objectos de vídeo e cenas</i>	53
3.3	CODIFICADOR MPEG-4	53
3.3.1	<i>Controlo de débito</i>	58
3.3.2	<i>Perfis, níveis e tipos de objecto</i>	60
3.4	DESCODIFICADOR MPEG-4	63
3.5	VISUALIZAÇÃO DE CENAS	64
3.6	FICHEIROS GERADOS PELA APLICAÇÃO	64
3.7	OUTRAS FUNCIONALIDADES	66
3.7.1.1	<i>Directórios de defeito</i>	66
3.7.1.2	<i>Cor de um pixel</i>	66
3.7.1.3	<i>Manual da aplicação</i>	66
3.8	CONCLUSÕES	66

CAPÍTULO 4 OPTIMIZAÇÃO DO DESCODIFICADOR DE VÍDEO MPEG-4

69

4.1	INTRODUÇÃO	69
4.2	O CODEC DE VÍDEO MoMuSys/IST	70
4.3	MELHORIAS AO DESCODIFICADOR DE VÍDEO MoMuSys/IST	71
4.3.1	<i>Condições de teste</i>	71
4.3.2	<i>Desempenho do decodificador inicial</i>	72
4.3.2.1	<i>Descrição das principais funções do decodificador</i>	73
4.3.2.2	<i>Tempo gasto pelas principais funções de decodificação</i>	74
4.3.2.3	<i>Influência do perfil no desempenho do decodificador</i>	78
4.3.2.4	<i>Influência do nível no desempenho do decodificador</i>	79
4.3.3	<i>Optimização de funções seleccionadas</i>	81
4.3.4	<i>Sistema de gestão de memória</i>	84
4.3.5	<i>Desempenho do decodificador optimizado</i>	87
4.4	CONCLUSÕES	90

CAPÍTULO 5 ESTUDO DA COMPLEXIDADE DE DESCODIFICAÇÃO DOS MACROBLOCOS DE VÍDEO MPEG-4

91

5.1	INTRODUÇÃO	91
5.2	TIPOS DE MACROBLOCOS	92
5.2.1	<i>Macroblocos com informação de textura</i>	92
5.2.2	<i>Macroblocos com informação de forma</i>	93
5.2.3	<i>Tipos relevantes de macroblocos</i>	94
5.3	MODELOS DE COMPLEXIDADE DOS MACROBLOCOS	95
5.4	METODOLOGIA PARA A AVALIAÇÃO DA COMPLEXIDADE DE DESCODIFICAÇÃO DOS MACROBLOCOS	97
5.4.1	<i>Processo de decodificação</i>	98
5.4.1.1	<i>Decodificação de VOPs Intra</i>	101
5.4.1.2	<i>Decodificação de VOPs Inter</i>	101
5.4.2	<i>Medição do tempo de decodificação</i>	102
5.5	AVALIAÇÃO DA COMPLEXIDADE: RESULTADOS	103
5.5.1	<i>Avaliação da complexidade: objectos de vídeo rectangulares</i>	103
5.5.2	<i>Avaliação da complexidade: objectos de vídeo de forma arbitrária</i>	106

5.5.3	<i>Avaliação da complexidade: macroblocos transparentes</i>	110
5.5.4	<i>Comparação da complexidade para os vários tipos de macroblocos</i>	112
5.5.4.1	<i>Comparação entre os tipos de macroblocos para objectos rectangulares</i>	112
5.5.4.2	<i>Comparação entre os tipos de macroblocos para objectos com forma arbitrária</i>	114
5.5.5	<i>Determinação da complexidade relativa dos vários tipos de macroblocos de vídeo MPEG-4</i>	121
5.6	CONCLUSÕES	125

CAPÍTULO 6 MODELO ALTERNATIVO PARA O VCV **127**

6.1	INTRODUÇÃO	127
6.2	MODELO VCV ESPECIFICADO NA NORMA MPEG-4 VISUAL	127
6.2.1	<i>Parâmetros do modelo VCV</i>	128
6.2.2	<i>Dinâmica de ocupação do VCV</i>	129
6.2.3	<i>Restrições impostas pelo modelo VCV</i>	131
6.3	ABORDAGENS ALTERNATIVAS PARA O MODELO VCV	131
6.3.1	<i>Memória única com ritmo de descodificação único</i>	131
6.3.2	<i>Memória única com vários ritmos de descodificação</i>	132
6.3.3	<i>Várias memórias com vários ritmos de descodificação</i>	133
6.3.4	<i>Modelo VCV MPEG-4</i>	134
6.4	MODELO VCV IST: UMA SOLUÇÃO MAIS EFICIENTE	138
6.4.1	<i>Comparação entre os modelos VCV IST e VCV MPEG-4</i>	140
6.4.1.1	<i>Comparação entre os modelos VCV: cenas com um único objecto de vídeo rectangular</i>	141
6.4.1.2	<i>Comparação entre os modelos VCV: cenas com objectos de vídeo de forma arbitrária</i>	143
6.5	CONCLUSÕES	149

CAPÍTULO 7 CONCLUSÕES E TRABALHO FUTURO **151**

BIBLIOGRAFIA **157**

Índice de Figuras

<i>Figura 1 – Canal Bloomberg [5]: exemplo de conteúdo para o qual o modelo MPEG-4 pode trazer enormes benefícios em termos de eficiência e funcionalidades.</i>	<i>3</i>
<i>Figura 2 – Exemplo de uma cena MPEG-4 composta por vários objectos audiovisuais: um objecto de vídeo como fundo, um objecto de vídeo de forma arbitrária que contém o comentador, três outros objectos de vídeo de menor dimensão e com hiperligação no lado direito do ecrã e quatro objectos gráficos com hiperligação na parte de baixo do ecrã.</i>	<i>8</i>
<i>Figura 3 – Arquitectura de um sistema MPEG-4.</i>	<i>9</i>
<i>Figura 4 – Cenas compostas por objectos de origem natural e sintética.</i>	<i>12</i>
<i>Figura 5 – Estrutura hierárquica de um fluxo binário de vídeo MPEG-4.</i>	<i>17</i>
<i>Figura 6 – Modos de codificação de um VOP: I, P e B.</i>	<i>18</i>
<i>Figura 7 – Exemplo de uma cena com vários objectos.</i>	<i>19</i>
<i>Figura 8 – Árvore correspondente à descrição da cena na Figura 7.</i>	<i>20</i>
<i>Figura 9 – Codificação aritmética baseada no contexto (CAE) no modo Intra (a) e Inter (b). O shapel a ser codificado está marcado com um círculo e os shapels que contribuem para o cálculo do contexto estão marcados com uma cruz.</i>	<i>21</i>
<i>Figura 10 – Exemplo da utilização de padding para a estimação de movimento.</i>	<i>23</i>
<i>Figura 11 – Exemplo dos vários tipos de macroblocos existentes num VOP de forma arbitrária.</i>	<i>24</i>
<i>Figura 12 – Exemplo de uma sprite para a sequência Stefan.</i>	<i>25</i>
<i>Figura 13 – Descodificação utilizando códigos reversíveis de comprimento variável.</i>	<i>27</i>
<i>Figura 14 – Técnicas de resiliência a erros na norma MPEG-4 Visual.</i>	<i>27</i>
<i>Figura 15 – Escalabilidade temporal de vídeo na norma MPEG-4.</i>	<i>28</i>
<i>Figura 16 – Escalabilidade espacial de vídeo na norma MPEG-4.</i>	<i>29</i>
<i>Figura 17 – Relações existentes entre os tipos de objecto e os perfis de vídeo definidos na versão 1 da norma MPEG-4.</i>	<i>35</i>
<i>Figura 18 – Mecanismo de verificação de vídeo.</i>	<i>39</i>
<i>Figura 19 – Dinâmica da ocupação do VBV.</i>	<i>39</i>
<i>Figura 20 – Dinâmica de ocupação do VCV.</i>	<i>40</i>
<i>Figura 21 – Dinâmica da ocupação do VMV.</i>	<i>41</i>
<i>Figura 22 – Aspecto geral da aplicação.</i>	<i>45</i>
<i>Figura 23 – Parâmetros básicos na criação de uma nova cena.</i>	<i>46</i>
<i>Figura 24 – Criação de um novo objecto de vídeo.</i>	<i>47</i>
<i>Figura 25 – Menu do objecto de vídeo na árvore de objectos.</i>	<i>48</i>

<i>Figura 26 – Menu do objecto de vídeo na árvore de objectos depois de o objecto estar inserido na cena.</i>	49
<i>Figura 27 – Informação temporal para o objecto Akiyo.</i>	50
<i>Figura 28 – Menu do objecto de vídeo na cena.</i>	50
<i>Figura 29 – Definição das propriedades de composição e codificação de um objecto de vídeo.</i>	51
<i>Figura 30 – Botões para controlo da visualização da cena.</i>	52
<i>Figura 31 – Definição dos parâmetros de codificação de um objecto de vídeo: a) estrutura temporal; b) dados definidos pelo utilizador, que podem ser incluídos no fluxo binário; c) resiliência a erros; d) movimento; e) quantificação; f) controlo de débito; g) forma; h) escalabilidade; i) estimação de complexidade; j) directório para a escrita do VOP codificado reconstruído pelo codificador.</i>	57
<i>Figura 32 – Controlo de débito ao nível do objecto.</i>	59
<i>Figura 33 – Controlo de débito ao nível da cena.</i>	60
<i>Figura 34 – Tipos de objecto de vídeo: a) objectos na cena; b) escolha do tipo de objecto.</i>	61
<i>Figura 35 – Escolha do perfil e nível para a cena.</i>	61
<i>Figura 36 – Gráfico com o enchimento das várias memórias do mecanismo de verificação de vídeo.</i>	62
<i>Figura 37 – Estatísticas da codificação dos objectos Akiyo e Stefan.</i>	63
<i>Figura 38 – Janela de diálogo para a descodificação de uma cena.</i>	64
<i>Figura 39 – Tutorial sobre a aplicação, disponível a página do Grupo de Imagem.</i>	67
<i>Figura 40 – Imagens das sequências de teste: a) Akiyo; b) News; c) Coastguard; d) Stefan.</i>	72
<i>Figura 41 – Alterações efectuadas para otimizar o acesso: a) imagens bidimensionais e b) imagens armazenadas em vectores.</i>	83
<i>Figura 42 – Funcionamento do vector de ponteiros para o sistema de gestão de memória. O vector de ponteiros está representado ao centro e cada vector aponta para um bloco de memória.</i>	86
<i>Figura 43 – VOP de forma arbitrária mostrando vários tipos de macroblocos.</i>	92
<i>Figura 44 – Estratégias de avaliação da complexidade de descodificação para vídeo MPEG-4.</i>	96
<i>Figura 45 – Fluxograma de descodificação para VOPs Intra.</i>	99
<i>Figura 46 – Fluxograma de descodificação para VOPs Inter.</i>	100
<i>Figura 47 – Tempo de descodificação dos macroblocos para a sequência Akiyo: rectangular, QCIF, Simple@L2, 128 kbit/s (número de macroblocos usados: Intra – 99, Inter – 11287, Inter4V – 274).</i>	104
<i>Figura 48 – Tempo de descodificação dos macroblocos para a sequência News: rectangular, CIF, Simple@L3, 384 kbit/s (número de macroblocos usados: Intra – 536, Inter – 35141, Inter4V – 6225).</i>	105

<i>Figura 49 – Tempo de descodificação dos macroblocos para a sequência News: rectangular, CIF, Core@L2, 2000 kbit/s (número de macroblocos usados: Intra – 538, Inter – 89926, Inter4V – 5968).</i>	105
<i>Figura 50 – Tempo de descodificação para vários tipos de codificação da forma: Coastguard, QCIF, Core@L1, 384 kbit/s, textura em modo Intra (número de macroblocos usados: Opaco – 62, NoUpdate – 4, IntraCAE – 63).</i>	107
<i>Figura 51 – Tempo de descodificação para vários tipos de codificação da forma: Coastguard, QCIF, Core@L1, 384 kbit/s, textura em modo Inter (número de macroblocos usados: Opaco – 290, NoUpdate – 28182, IntraCAE – 634, InterCAE – 5247).</i>	107
<i>Figura 52 – Tempo de descodificação para vários tipos de codificação da forma: Coastguard, QCIF, Core@L1, 384 kbit/s, textura em modo Inter4V (número de macroblocos usados: Opaco – 38, NoUpdate – 2329, IntraCAE – 138, InterCAE – 1207).</i>	108
<i>Figura 53 – Tempo de descodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Intra (número de macroblocos usados: Opaco – 356, NoUpdate – 485, IntraCAE – 264).</i>	108
<i>Figura 54 – Tempo de descodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Inter (número de macroblocos usados: Opaco – 1175, NoUpdate – 74314, IntraCAE – 5334, InterCAE – 2364).</i>	109
<i>Figura 55 – Tempo de descodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Inter4V (número de macroblocos usados: Opaco – 1061, NoUpdate – 25207, IntraCAE – 6056, InterCAE – 2351).</i>	109
<i>Figura 56 – Histograma para o tempo de descodificação de macroblocos transparentes: Coastguard, QCIF, Core@L1, 384 kbit/s (número de macroblocos usados: 7673).</i>	111
<i>Figura 57 – Histograma para o tempo de descodificação de macroblocos transparentes: Stefan, CIF, Core@L2, 2000 kbit/s (número de macroblocos usados: 10500).</i>	111
<i>Figura 58 – Tempo de descodificação para os macroblocos (só com informação de textura) em sequências rectangulares (número de macroblocos usados: Intra – 61653, Inter – 252982, Inter4V – 30317).</i>	113
<i>Figura 59 – Histograma do tempo de descodificação dos macroblocos Skipped para objectos rectangulares (número de macroblocos usados: 114660).</i>	113
<i>Figura 60 – Tempo de descodificação para os macroblocos opacos em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra – 19272, Inter – 1544, Inter4V – 1391).</i>	114
<i>Figura 61 – Tempo de descodificação para os macroblocos com codificação de forma NoUpdate em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra – 500, Inter – 73256, Inter4V – 29913).</i>	115
<i>Figura 62 – Tempo de descodificação para os macroblocos com codificação de forma IntraCAE e InterCAE em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra+IntraCAE – 6997, Inter+IntraCAE – 8583, Inter4V+IntraCAE – 12131, Inter+InterCAE – 16251, Inter4V+InterCAE – 11284).</i>	116
<i>Figura 63 – Histograma do tempo de descodificação dos macroblocos Skipped para forma opaca (número de macroblocos usados: 184).</i>	117
<i>Figura 64 – Histograma do tempo de descodificação dos macroblocos Skipped para forma NoUpdate (número de macroblocos usados: 10914).</i>	118

<i>Figura 65 – Histograma do tempo de descodificação dos macroblocos Skipped para forma IntraCAE (número de macroblocos usados: 201).</i>	118
<i>Figura 66 – Histograma do tempo de descodificação dos macroblocos Skipped para forma InterCAE (número de macroblocos usados: 763).</i>	119
<i>Figura 67 – Dinâmica da ocupação da memória VCV.</i>	130
<i>Figura 68 – Sequência Container: objecto composto principalmente por macroblocos transparentes (82%).</i>	136
<i>Figura 69 – Sequência News: os macroblocos dentro dos rectângulos são contados três vezes para o VCV e VMV, duas como transparentes e uma como opacos ou de fronteira.</i>	136
<i>Figura 70 – Sequência News: a) número de macroblocos; b) ocupação das memórias.</i>	137
<i>Figura 71 – Sequência Coastguard: a) número de macroblocos; b) ocupação das memórias.</i>	137
<i>Figura 72 – Evolução da ocupação das várias memórias: sequência Akiyo, rectangular, QCIF a 30 tramas/s, para Simple@L1 a 64 kbit/s.</i>	142
<i>Figura 73 – Evolução da ocupação das várias memórias: sequência Stefan, rectangular, CIF a 30 tramas/s para Simple@L3 a 384 kbit/s.</i>	142
<i>Figura 74 – Evolução da ocupação das várias memórias: sequência Coastguard, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.</i>	143
<i>Figura 75 – Sequência Akiyo_and_Coastguard, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.</i>	144
<i>Figura 76 – Evolução da ocupação das várias memórias: sequência Akiyo_and_Coastguard, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.</i>	144
<i>Figura 77 – Sequência Children_and_Flag, 3 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s, com as bounding boxes representadas para cada objecto.</i>	145
<i>Figura 78 – Evolução da ocupação das várias memórias: sequência Children_and_Flag, 3 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.</i>	146
<i>Figura 79 – Evolução da ocupação das várias memórias: sequência News, 4 VOs, QCIF a 15 tramas/s para Core@L1 a 384 kbit/s.</i>	146
<i>Figura 80 – Evolução da ocupação várias memórias: sequência News, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.</i>	147
<i>Figura 81 – Evolução da ocupação das várias memórias: sequência News, 4 VOs, CIF a 30 tramas/s para Core@L2 a 2000 kbit/s.</i>	148
<i>Figura 82 – Evolução do número de macroblocos: sequência News, 4 VOs, CIF a 30 tramas/s para Core@L2 a 2000 kbit/s.</i>	148

Índice de Tabelas

<i>Tabela 1 – Ferramentas utilizadas para os vários tipos de objectos visuais na versão 1 da norma MPEG-4.</i>	32
<i>Tabela 2 – Tipos de objecto suportados pelos perfis visuais definidos na versão 1 da norma MPEG-4.</i>	34
<i>Tabela 3 – Tipos de objecto visuais e respectivo número máximo de objectos para cada perfil@nível definido na versão 1 da norma MPEG-4.</i>	36
<i>Tabela 4 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Akiyo para Simple@L2.</i>	75
<i>Tabela 5 – Percentagem de tempo gasto para as principais funções de descodificação: sequência News para Simple@L3.</i>	75
<i>Tabela 6 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Coastguard para Core@L1.</i>	76
<i>Tabela 7 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Stefan para Core@L2.</i>	76
<i>Tabela 8 – Média e variância da percentagem de tempo gasto nas principais funções de descodificação das sequências de teste.</i>	77
<i>Tabela 9 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Akiyo para Simple@L2 e Core@L1 a 128 kbit/s.</i>	78
<i>Tabela 10 – Percentagem de tempo gasto para as principais funções de descodificação: sequência News para Simple@L3 e Core@L2 a 384 kbit/s.</i>	79
<i>Tabela 11 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Akiyo para Simple@L2 e Simple@L3 a 128 e 384 kbit/s.</i>	80
<i>Tabela 12 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Coastguard para Core@L1 e Core@L2 a 384 e 2000 kbit/s.</i>	81
<i>Tabela 13 – Tipos de blocos implementados no sistema de gestão de memória.</i>	85
<i>Tabela 14 – Percentagem de tempo e tempo total gasto para as principais funções de descodificação: sequência Akiyo para Simple@L2.</i>	87
<i>Tabela 15 – Percentagem de tempo e tempo total gasto para as principais funções de descodificação: sequência News para Simple@L3.</i>	88
<i>Tabela 16 – Percentagem de tempo e tempo total gasto para as principais funções de descodificação: sequência Coastguard para Core@L1.</i>	88
<i>Tabela 17 – Percentagem de tempo e tempo total gasto para as principais funções de descodificação: sequência Stefan para Core@L2.</i>	89
<i>Tabela 18 – Ganhos entre os decodificadores original e optimizado em termos de tempo total de descodificação.</i>	89
<i>Tabela 19 – Tipos de macroblocos a estudar: a) em termos de informação de textura; b) em termos de informação de forma.</i>	94

Tabela 21 – Valores estatísticos para o tempo de descodificação dos vários tipos de macroblocos para as sequências utilizadas: média, desvio padrão, moda e mediana. ____ 120

Tabela 23 – Tempos de descodificação médio e máximo para os vários tipos de macroblocos e respectivas relações de tempos. _____ 124

Tabela 25 – Pesos relativos de complexidade atribuídos às várias classes de macroblocos. _____ 125

Tabela 27 – Dimensão das memórias VMV e VCV/B-VCV e respectivos ritmos de descodificação/esvaziamento para cada perfil@nível visual. _____ 135

Tabela 28 – Pesos relativos de complexidade atribuídos aos vários tipos de macroblocos. 139

Lista de Acrónimos

- 3GPP – *Third Generation Partnership Project*
- ACTS – *Advanced Communications Technologies and Services*
- ATSC – *Advanced Television Systems Committee*
- BAB – *Binary Alpha Block*
- BIFS – *Binary Format for Scenes*
- B-VCV – *Boundary macroblocks Video Complexity Verifier*
- CAE – *Context-based Arithmetic Encoding*
- CDi – *Compact Disc Interactive*
- CIF – *Common Intermediate Format*
- DAB – *Digital Audio Broadcasting*
- DCT – *Discrete Cosine Transform*
- DMIF – *Delivery Multimedia Integration Framework*
- DVB – *Digital Video Broadcasting*
- DVD – *Digital Versatile Disc*
- DWT – *Discrete Wavelet Transform*
- FGS – *Fine Grain Scalability*
- FIFO – *First In First Out*
- GOV – *Group Of Video object planes*
- IDCT – *Inverse Discrete Cosine Transform*
- IEC – *International Electrotechnical Commission*
- IPMP – *Intellectual Property Management & Protection*
- ISO – *International Organization for Standardization*
- IST – *Instituto Superior Técnico*
- JPEG – *Joint Pictures Experts Group*
- MB – *Macrobloco*
- MoMuSys – *Mobile Multimedia Systems*
- MPEG – *Moving Pictures Experts Group*
- MVDS – *Motion Vector Difference for Shape*
- NTSC – *National Television System Committee*

OCI – *Object Content Information*
PAL – *Phase Alternating Line*
PSNR – *Power to Signal Noise Ratio*
QCIF – *Quarter-CIF*
RAM – *Random Access Memory*
RGB – *Red, Green, Blue*
RVLC – *Reversible Variable Length Coding*
SECAM – *SEquentiel Couleur A Memoire*
SIF – *Source Intermediate Format*
UMTS – *Universal Mobile Telecommunications System*
URL – *Uniform Resource Locator*
VBV – *Video rate Buffer Verifier*
VCV – *Video Complexity Verifier*
VHS – *Video Home System*
VLC – *Variable Length Coding*
VMV – *Video reference Memory Verifier*
VO – *Video Object*
VOL – *Video Object Layer*
VOP – *Video Object Plane*
VRML – *Virtual Reality Modeling Language*
VS – *Visual object Sequence*
WAP – *Wireless Application Protocol*

Capítulo 1

Introdução

Vivemos na era digital. Na verdade, a tremenda evolução na tecnologia digital que se verificou nos últimos anos tem vindo a modificar a forma como todo o tipo de informação e, em especial, a informação visual, é criada, manipulada e consumida. Uma das áreas que conheceu maior desenvolvimento foi, precisamente, a das tecnologias relacionadas com informação audiovisual. Hoje em dia, a produção de conteúdo audiovisual é uma tarefa que é cada vez mais fácil e banalizada. Esta é uma área com grande interesse, cujo crescimento é estimulado pelo aparecimento, e vendas em crescendo, de máquinas fotográficas digitais que armazenam as imagens directamente no formato JPEG¹ e de câmaras digitais, surgidas mais recentemente, que gravam vídeo no formato MPEG-1². A disponibilização crescente destes dispositivos, bem como a consequente baixa dos preços, representa um grande passo para a aceitação das tecnologias de aquisição e distribuição digital de informação audiovisual por parte do mercado de consumo e já não só por parte dos profissionais. Neste novo contexto, qualquer pessoa é um potencial produtor de conteúdo que pode ser facilmente criado, editado, distribuído e publicado, nomeadamente usando a Internet. Existem pois muitas áreas em que o vídeo digital está a substituir o vídeo analógico, depois de o mesmo ter já acontecido com a música. Exemplo disso é a recente introdução da televisão digital em vários países, e muitos outros se seguem, ou a substituição progressiva das cassetes de vídeo analógicas, por exemplo VHS, pelo *Digital Versatile Disc* (DVD) como meio privilegiado para gravar filmes.

As normas MPEG-1 [1] e MPEG-2 [2] foram, em termos de tecnologia de representação audiovisual, as soluções chave que permitiram a divulgação e aceitação dos novos meios de produção e difusão de conteúdo audiovisual. Nas aplicações existentes, implementadas com base nestas normas, o vídeo digital oferece as mesmas funcionalidades do vídeo analógico, ou seja, o conteúdo é representado segundo o mesmo modelo de dados – uma sequência periódica de tramas rectangulares – mas agora usando um formato digital e já não analógico.

¹ Joint Pictures Experts Group

² Moving Pictures Experts Group

A representação digital traz, como é óbvio, grandes benefícios, já que se obtém normalmente maior eficiência em termos de banda, para além de qualidade e fiabilidade acrescidas, mas as formas de consumo do conteúdo pelos utentes mantiveram-se fundamentalmente as mesmas por não mudar o modelo de dados usado na representação. Contudo, a crescente difusão das tecnologias digitais, nomeadamente via Internet, tem vindo a pôr cada vez mais à vista as limitações do modelo de vídeo baseado nas tramas, mostrando que novas funcionalidades podem ser oferecidas aos utentes, nomeadamente em termos de acesso e manipulação da informação, se um novo modelo de dados for adoptado. A norma MPEG-4 [3] veio disponibilizar e normalizar as tecnologias de representação audiovisual que permitem alcançar estas novas funcionalidades.

Com a grande expansão da Internet, surgiu uma nova forma de lidar com a informação audiovisual, nomeadamente permitindo uma maior interacção com os conteúdos, já não só ao nível da sequenciação temporal como antes, por exemplo *pause*, *play*, *fast reverse*, mas também ao nível dos vários componentes informativos, por exemplo fotografias, texto, gráficos, botões com hiperligações, o que demonstrou que o paradigma da televisão tradicional já não chega para satisfazer as expectativas criadas em termos de consumo de informação audiovisual. Os utilizadores querem agora lidar com o áudio e o vídeo, em qualquer contexto, da mesma forma que lidam com o texto e gráficos na Internet e ainda poder ter uma relação mais directa com os vários elementos duma cena audiovisual, por exemplo reutilizando um dos elementos da cena ou alterando a sua posição na cena [4].

Na verdade, todas as normas internacionais de representação de vídeo criadas até hoje, nomeadamente as normas analógicas PAL, NTSC e SECAM e as digitais H.261, H.263, MPEG-1 e MPEG-2, modelam a informação de vídeo em termos de tramas, ou seja, uma sequência de vídeo corresponde a um série de imagens rectangulares que se sucedem a uma frequência constante. Os métodos de codificação de vídeo baseados no modelo de tramas utilizam técnicas que exploram as características estatísticas da informação visual dentro de uma trama e entre tramas. Este tipo de modelo dá pouca importância ao conteúdo da cena e à sua semântica, apresentando por isso limitações ao nível das funcionalidades que oferece, já que o utilizador permanece um consumidor/espectador passivo cuja única interacção com o conteúdo se limita a alterar o som ou o brilho da imagem ou a usar os modos de gravação *play*, *pause*, *fast forward* e *fast reverse*. Para além disso, as técnicas de compressão baseadas neste modelo parecem ter chegado a um ponto de saturação, não sendo previsível que melhorias significativas aconteçam nesta área nos tempos mais próximos.

Com a intenção de criar uma nova norma de representação audiovisual que utilizasse um novo modelo de dados, oferecendo mais funcionalidades aos utentes, o grupo ISO/MPEG (*Moving Pictures Experts Group*) lançou, em 1993, um novo projecto, denominado “Codificação de Objectos Audiovisuais”, geralmente conhecido por MPEG-4 [4]. De facto, a norma MPEG-4 é a primeira norma internacional de representação audiovisual que modela uma cena como uma composição de objectos audiovisuais independentes, com determinadas características no espaço e no tempo. Esta nova abordagem da representação audiovisual está associada a uma maior compreensão/interpretação do conteúdo da imagem, que se assemelha mais ao modo como o ser humano adquire, representa e interpreta a informação audiovisual. O conceito de composição da cena por objectos audiovisuais vai ao encontro das expectativas dos utilizadores, cada vez mais habituados a um certo nível de interactividade ao nível dos conteúdos, por exemplo através da Internet. Actualmente, há já muitas aplicações em que o

utilizador não fica satisfeito só por ver, desejando também “mexer”, por exemplo aplicações educativas, jogos, comércio electrónico. O modelo de representação baseado em objectos traz novas funcionalidades ao nível da interacção baseada em conteúdo e manipulação desse conteúdo, por exemplo adicionar ou remover objectos da cena, mudar a posição na cena dos objectos, alterar as suas propriedades em termos de dimensão, forma, textura, associar hiperligações a objectos, etc. Inerente a este modelo está um aumento significativo da eficiência de codificação, nomeadamente para determinados conteúdos como os que se mostram na Figura 1, já que cada objecto pode ser codificado de um modo independente e adequado às suas características específicas. No caso da Figura 1, retirada do canal *Bloomberg* [5], hoje disponível na rede de televisão por cabo, a norma MPEG-4 permite que as zonas de texto sejam consideradas objectos textuais e logo codificadas como tal e não como vídeo natural, como acontecia com as normas anteriores; esta adequação da tecnologia de codificação ao tipo de dados em questão traz aumentos significativos de eficiência e de qualidade. Para além disso, com a norma MPEG-4, o utilizador poderá facilmente controlar o número, posição e características dos vários objectos textuais e de vídeo presentes na cena, personalizando a sua cena em termos de conteúdo (objectos) e *layout*.



Figura 1 – Canal Bloomberg [5]: exemplo de conteúdo para o qual o modelo MPEG-4 pode trazer enormes benefícios em termos de eficiência e funcionalidades.

Por outro lado, a informação audiovisual é hoje transmitida num número cada vez maior de tipos de redes diferentes. A única informação visual que era transmitida até há bem pouco tempo era o sinal de televisão. Com a massificação da Internet, os utilizadores querem ter acesso a áudio e a vídeo, o que implica a necessidade de oferecer conteúdos audiovisuais com qualidade aceitável, a baixos débitos binários, na Internet. Por outro lado, a mobilidade das comunicações é um dado adquirido, como o comprova a explosão do número de telefones móveis, cada vez mais sofisticados em termos de funcionalidades. Num futuro próximo, as comunicações móveis não estarão limitadas à voz e aos dados, mas irão abrir-se a outros tipos de dados, nomeadamente dados multimédia (os primeiros passos foram já dados com a Internet móvel, o WAP). Atento à importância destes dois ambientes – Internet e redes móveis – o grupo MPEG incluiu na norma MPEG-4 ferramentas de resiliência a erros e o suporte de débitos binários muito baixos, sempre num contexto de representação baseada em objectos e logo continuando a oferecer novas funcionalidades.

Com vista a explorar as vantagens da convergência digital, a norma MPEG-4 foi desenvolvida com vista a ser usada num grande número de aplicações que requerem um leve

muito abrangente de funcionalidades, tipos de dados, débitos binários, etc. Esta abrangência em termos de aplicações e funcionalidades justifica o elevado número de ferramentas que a norma inclui. Pensar em garantir a interoperabilidade entre terminais MPEG-4 através da implementação em todos eles de todas as ferramentas especificadas é algo impraticável, uma vez que implicaria necessariamente uma complexidade de implementação muito grande, provavelmente insuportável para algumas aplicações. Assim, a norma é construída como uma “caixa de ferramentas” – *toolbox approach* – e não como um bloco indivisível, permitindo criar soluções incluindo apenas parte das ferramentas desenvolvidas, ajustadas (as soluções) a determinadas classes de aplicações. Para garantir a interoperabilidade, é necessário que estes subconjuntos de ferramentas sejam normativamente definidos, surgindo assim os **perfis** que consistem em subconjuntos de ferramentas que o MPEG agrupou para satisfazer as necessidades de certas classes de aplicações. Assim, cada fabricante/operador/utilizador poderá escolher o perfil que mais se adequa às suas necessidades, limitando a complexidade dos dispositivos, sem perder significativamente interoperabilidade. Contudo, a especificação de perfis não é suficiente para limitar a complexidade, pois ainda assim poderia existir um (ou mais) objecto visual com um débito binário arbitrário que excedesse as capacidades computacionais do descodificador (em termos de memória ou descodificação em tempo real). Surgem assim os **níveis** que especificam, para cada perfil, as restrições impostas às ferramentas definidas pelos perfis, através de limitações impostas a alguns parâmetros relevantes, por exemplo o débito binário. Esta estratégia combinada perfil-nível é fundamental para garantir a interoperabilidade entre terminais MPEG-4 e limitar a complexidade/custo, possibilitando assim que mais classes de aplicações adoptem a norma MPEG-4. A combinação perfil@nível define um ponto de conformidade, ou seja, define regras que garantem que os fluxos binários gerados por um codificador que trabalhe com um dado perfil@nível podem ser correctamente descodificados por um descodificador conforme com essa mesma combinação.

Para poder garantir a conformidade dos fluxos binários de vídeo a uma dada combinação perfil@nível, a norma MPEG-4 define o **mecanismo de verificação de vídeo** [3]. O mecanismo de verificação de vídeo é implementado no codificador e consiste em três memórias virtuais que acumulam os bits gerados através do *Video Buffering Verifier* (VBV), o número de macroblocos codificados através do *Video Complexity Verifier* (VCV) e a memória através do *Video Memory Verifier* (VMV). O codificador deve verificar se os fluxos binários gerados estão de acordo com as características de um dado perfil@nível, controlando o enchimento e esvaziamento das memórias referidas. O tema principal desta tese está precisamente associado ao mecanismo de verificação de vídeo que estabelece as regras que garantem a conformidade dos fluxos binários, mais exactamente com o modelo VCV que é utilizado para verificar se a capacidade computacional, medida em macroblocos por segundo, necessária no descodificador para descodificar os fluxos binários gerados pelo codificador não excede os valores especificados para a combinação perfil@nível escolhida.

O principal objectivo desta tese é a definição de um novo modelo VCV, mais eficiente do que o especificado pela norma MPEG-4, no sentido de que leva em conta as diferentes complexidades de descodificação associadas aos vários tipos de macroblocos de vídeo possíveis. O modelo VCV actualmente especificado na norma MPEG-4 apenas distingue entre os macroblocos de fronteira (com contorno) e os restantes macroblocos de vídeo, não diferenciando a complexidade inerente aos vários tipos de macroblocos que não de fronteira.

Este facto leva a que macroblocos com vários graus de complexidade de descodificação sejam tratados do mesmo modo pelo modelo, por exemplo macroblocos transparentes e macroblocos opacos. O novo modelo VCV, proposto nesta tese, explora as diferenças de complexidade existentes entre os vários tipos de macroblocos, permitindo que a complexidade estimada se aproxime mais da complexidade real dos objectos codificados, utilizando de um modo mais eficiente os recursos do descodificador. Com o novo modelo, fluxos binários que seriam considerados não conformes para um dado perfil@nível, usando o modelo MPEG-4, passarão agora a ser conformes, isto mantendo os mesmos recursos de descodificação.

O trabalho desenvolvido no contexto desta tese pode ser dividido em duas partes principais:

- **Aplicação de criação, codificação e descodificação de cenas MPEG-4 [6]** – A aplicação desenvolvida permite a criação de cenas compostas por vários objectos de vídeo através de uma interface potente, flexível e amigável. Estas cenas podem ter quaisquer características, por exemplo em termos de número de objectos, dimensão, forma, textura, movimento, posição, etc, permitindo assim testar as vantagens do novo modelo VCV implementado com maior credibilidade, ou seja usando cenas com características críticas em termos de teste. Os objectos de vídeo podem ser codificados em conformidade com a norma MPEG-4, segundo um determinado perfil e nível, podendo o utilizador especificar todos os parâmetros de codificação. A descodificação dos fluxos binários resulta em objectos de vídeo que podem ser acedidos de um modo independente, na cena descodificada. Os objectos utilizados para a criação de novas cenas são obtidos usando uma ferramenta de segmentação desenvolvida no Grupo de Imagem do Instituto de Telecomunicações, onde o autor desta tese esteve integrado. Tanto quanto se sabe, esta é a primeira aplicação a nível mundial que permite a codificação e descodificação de cenas de vídeo utilizando perfis e níveis visuais MPEG-4 e respeitando o mecanismo de verificação de vídeo. Esta aplicação foi apresentada ao grupo MPEG [6] e foi, até hoje, disponibilizada a mais de 20 companhias e universidades em todo o mundo que a pediram com vista a poder demonstrar, de um forma amigável, as capacidades da nova norma MPEG-4 Visual [3].
- **Novo modelo de verificação da complexidade de vídeo (VCV)** – O modelo proposto nesta tese pretende eliminar algumas das desvantagens do modelo actualmente especificado pela norma MPEG-4 Visual [3]. O modelo de verificação da complexidade de vídeo desenvolvido pretende substituir o modelo implementado na norma MPEG-4, que apenas distingue os macroblocos de fronteira dos restantes, em termos de complexidade. O novo modelo melhora substancialmente o modelo VCV da norma MPEG-4, pois explora de um modo mais preciso as diferenças de complexidade existentes entre os vários tipos de macroblocos que constituem os objectos de vídeo, não sobreavaliando a complexidade de alguns tipos de macroblocos. Torna-se assim possível ao codificador modelar de um modo mais preciso a complexidade de uma cena de vídeo e explorar de uma maneira mais eficiente a capacidade computacional do descodificador. Por outras palavras, o codificador passa a poder gerar fluxos binários, para um dado perfil@nível, que não seriam aceites pelo modelo VCV actual devido a uma sobreavaliação da

complexidade real mas que serão ainda assim correctamente descodificados por um descodificador com a capacidade correspondente ao perfil@nível escolhido.

O texto da presente tese está organizado em sete capítulos.

Este capítulo descreve a motivação e contexto da tese, os seus objectivos e o trabalho que foi desenvolvido.

O **capítulo 2** descreve a norma MPEG-4 Visual, com especial incidência nos aspectos de codificação de vídeo e de conformidade, nomeadamente os perfis, níveis e mecanismo de verificação de vídeo, devido à sua importância para esta tese. Neste capítulo, descrevem-se ainda as funcionalidades, organização, estrutura, sintaxe e ferramentas de codificação de vídeo da norma MPEG-4.

O **capítulo 3** descreve em pormenor a aplicação desenvolvida. Esta aplicação é uma ferramenta potente de composição e codificação, oferecendo grande flexibilidade em termos de criação e composição de conteúdo de vídeo baseado em objectos. Todos os fluxos binários gerados estão em conformidade com a norma MPEG-4 Visual e permitem o estudo dos vários aspectos da codificação de vídeo segundo a norma MPEG-4 Visual.

O **capítulo 4** descreve as alterações efectuadas ao descodificador de vídeo MoMuSys/IST com os objectivos de o tornar mais rápido e permitir a obtenção de uma estimativa credível da complexidade relativa das várias classes de macroblocos de vídeo usadas na norma MPEG-4 Visual [3].

O **capítulo 5** é dedicado ao estudo da complexidade relativa dos vários tipos de macroblocos usados na codificação de vídeo. Para isso foi utilizado o descodificador optimizado com vista a determinar quais as classes de macroblocos mais relevantes em termos de complexidade e quais os seus pesos relativos.

O **capítulo 6** descreve o modelo alternativo proposto para o VCV, baseado na atribuição de pesos diferentes às várias classes de macroblocos usados na codificação de vídeo, e relacionados com a sua complexidade efectiva. São apresentados resultados que mostram as vantagens do modelo desenvolvido relativamente ao modelo especificado pela norma MPEG-4 Visual [3].

O **capítulo 7** conclui a tese com algumas considerações sobre o trabalho desenvolvido e propostas para trabalho futuro, relacionadas com o tema estudado.

Capítulo 2

A Norma MPEG-4 Visual

2.1 Introdução

A norma MPEG-4 é a primeira norma de codificação audiovisual que representa o conteúdo como um conjunto de objectos audiovisuais que compõem a cena em questão e que apresentam um determinado comportamento no espaço e no tempo. Para melhor se entender o conceito de objecto audiovisual, imagine-se, por exemplo, a transmissão da volta a Portugal em bicicleta usando este modelo de representação (Figura 2). Em fundo, podem ver-se os ciclistas que estão destacados na frente da corrida, enquanto que outras janelas mais pequenas no lado direito do ecrã mostram a “cabeça” e a “cauda” do pelotão ou outras imagens relevantes. No canto superior esquerdo está a face do comentador que relata a corrida. Na parte inferior do ecrã podem ainda ver-se as diferenças de tempo entre os vários grupos de ciclistas. A barra na parte inferior do ecrã contém ainda hiperligações que levam a pôr em fundo a imagem correspondente ao *icon* que se seleccionar. Nesta cena existem vários objectos visuais: as várias imagens da corrida, a face segmentada do comentador e a janela gráfica com hiperligações e que inclui as várias diferenças de tempo. O objecto áudio seleccionado e que é sobreposto à voz do comentador é aquele correspondente à sequência de vídeo no fundo, ou seja o áudio alterar-se-á se no fundo passar a estar outra sequência de vídeo. Para apresentar ao espectador uma cena composta por objectos é ainda necessário enviar informação de composição que determina como se devem integrar os vários objectos na cena (pelo menos até o espectador alterar algo). A grande novidade na norma MPEG-4 é que o utilizador pode aceder independentemente a cada um dos objectos, alterando a sua disposição no ecrã, o seu número e tamanho, escolhendo qual das vistas deseja ver melhor numa janela aumentada ou até “sufar” até à página Web do principal fugitivo através de um simples *click* sobre a sua imagem. Para além disso, o espectador poderá ainda ver a classificação geral ou quem lidera as classificações das camisolas através de um menu lateral que é apenas mais um objecto visual colocado onde e quando o espectador entender.



Figura 2 – Exemplo de uma cena MPEG-4 composta por vários objectos audiovisuais: um objecto de vídeo como fundo, um objecto de vídeo de forma arbitrária que contém o comentador, três outros objectos de vídeo de menor dimensão e com hiperligação no lado direito do ecrã e quatro objectos gráficos com hiperligação na parte de baixo do ecrã.

O modelo de representação baseado na composição de objectos audiovisuais está na base de todas as novas funcionalidades oferecidas pela norma MPEG-4 e é a maior diferença conceptual relativamente às normas MPEG anteriores (MPEG-1 [1] e MPEG-2 [2]). Na Figura 3 está representada uma versão simplificada da arquitectura de um sistema MPEG-4. No lado do emissor, os vários objectos audiovisuais e ainda a informação de composição da cena são codificados separadamente. Os fluxos elementares gerados são depois multiplexados para formar um único fluxo que é enviado para o canal. No receptor, o fluxo binário recebido é desmultiplexado para se obterem os fluxos elementares dos objectos audiovisuais e da informação de composição de cena. Os fluxos elementares são então descodificados e o compositor compõe a cena com base na informação de composição de cena.

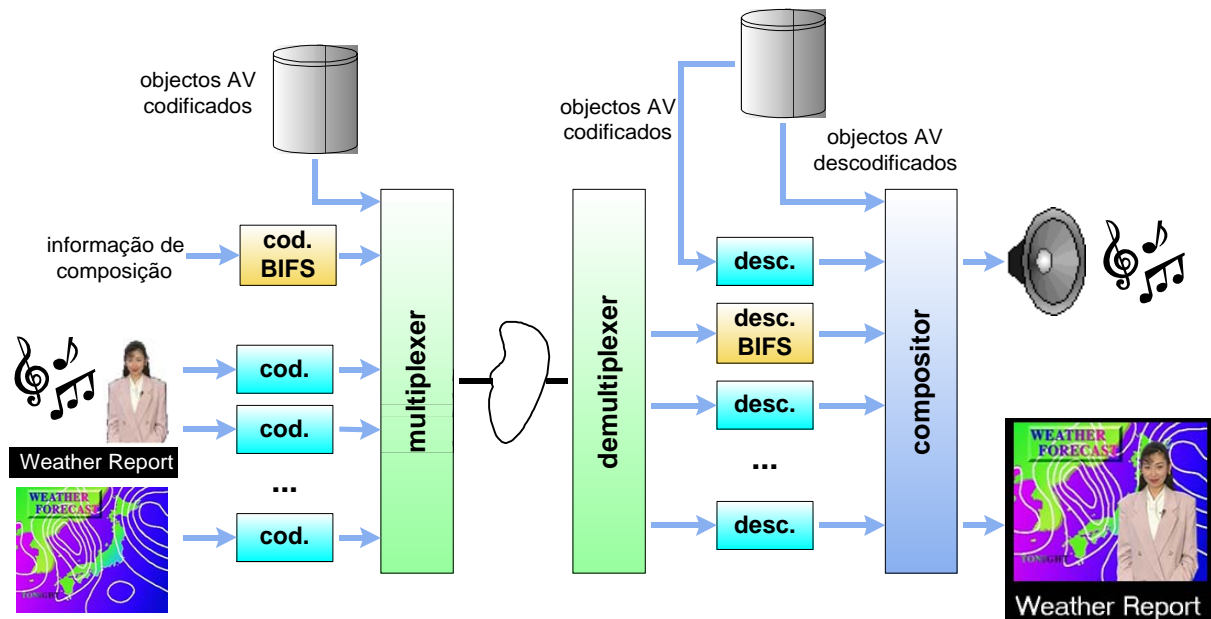


Figura 3 – Arquitectura de um sistema MPEG-4.

O facto de uma cena audiovisual ser modelada como uma composição de objectos, habitualmente com valor semântico ou seja com significado relevante no contexto da aplicação em questão, apresenta grandes potencialidades técnicas e funcionais, nomeadamente:

- **Processamento e codificação selectiva dos objectos** – O modelo baseado em objectos permite que tipos diferentes de objectos sejam processados e codificados de maneira diferente e adequada a cada um deles. Por exemplo, texto e vídeo não são codificados utilizando as mesmas ferramentas de codificação.
- **Reutilização de objectos** – Com a norma MPEG-4, qualquer objecto que seja colocado na cena permanece individualmente acessível, já que o fluxo codificado é independente para cada objecto, podendo deste modo ser facilmente reutilizado.
- **Integração de forma harmoniosa de conteúdos com origem natural e sintética** – A norma MPEG-4 permite integrar, numa mesma cena, objectos com origem natural ou sintética. Por exemplo, é possível integrar na mesma cena uma personagem de desenho animado e um actor real. Por outro lado, um actor pode fazer parte de um cenário virtual. Com a norma MPEG-4, conteúdos naturais e sintéticos “convivem” harmoniosamente na mesma cena sem qualquer discriminação.
- **Interação com e entre os objectos** – Como os objectos estão independentemente acessíveis, o utilizador pode interagir com a cena de várias formas, por exemplo, alterando a posição espacial de um objecto ou premindo o rato sobre o actor de um filme para obter informações sobre ele. Além disso, os objectos podem interagir entre si: por exemplo, um objecto pode alterar a sua posição ou aparência ou a de outro objecto se, ao moverem-se, chocarem na cena.

As vantagens deste novo modelo de representação são independentes do débito binário usado para a codificação, podendo ser utilizados débitos binários muito baixos, por exemplo

em comunicações móveis, ou débitos bastante elevados que permitam a grande qualidade de imagem e som exigida por utilizadores que têm ao seu dispor uma largura de banda e uma capacidade computacional cada vez maiores, por exemplo em estúdios televisivos.

Este capítulo pretende dar uma visão geral da norma de codificação audiovisual MPEG-4, ISO/IEC 14496, em particular a parte visual [3], apresentando as suas principais características, funcionalidades e ferramentas, bem como algumas aplicações relevantes. Será dada uma ênfase especial à parte da norma relacionada com a conformidade, ou seja à definição de perfis e níveis, devido à sua importância no contexto desta tese.

2.2 Funcionalidades

A norma MPEG-4 é o resultado de um projecto bastante ambicioso por parte do grupo ISO/MPEG no sentido de desenvolver uma norma de codificação audiovisual que inclua um conjunto vasto de ferramentas, de forma a oferecer inúmeras funcionalidades relacionadas com um maior grau de interacção com o conteúdo. Ao contrário do que sucedeu com a norma MPEG-1, que tinha como objectivo a gravação de conteúdo audiovisual em suporte CD-ROM, ou da norma MPEG-2 destinada principalmente à televisão digital de média e alta definição, a norma MPEG-4 não visa especificamente uma única classe de aplicações. Na verdade, a norma MPEG-4 especifica ferramentas de codificação eficientes para inúmeros tipos de aplicações, num vasto leque de débitos binários, usando pela primeira vez um modelo de dados baseado na composição de objectos. Para alcançar estes objectivos, a norma MPEG-4 trouxe inovações e melhorias relativamente às normas anteriores. Inicialmente foram identificadas oito funcionalidades principais que deveriam ser oferecidas pela norma MPEG-4 e que podem ser agrupadas em três classes [7]:

- **Interactividade baseada no conteúdo** – Esta classe inclui quatro funcionalidades destinadas a aplicações que envolvam interactividade entre o utilizador e o conteúdo:
 1. acesso baseado no conteúdo aos dados multimédia;
 2. manipulação do conteúdo e edição do fluxo binário;
 3. integração de conteúdo natural e sintético;
 4. acesso aleatório temporal melhorado.
- **Eficiência de compressão** – Esta classe considera duas funcionalidades que se destinam a aplicações que requeiram uma elevada eficiência de codificação para armazenamento ou transmissão de dados audiovisuais:
 1. eficiência de codificação melhorada;
 2. codificação integrada de fluxos binários concorrentes, e.g. estereoscópicos.
- **Acesso universal** – Esta classe inclui as restantes duas funcionalidades que permitem que dados codificados segundo a norma MPEG-4 sejam acessíveis numa vasta gama de equipamentos e canais, com vários níveis de qualidade em termos da resolução espacial e temporal de cada objecto:
 1. robustez em ambientes sujeitos a erros;
 2. escalabilidade baseada no conteúdo.

De seguida, descrevem-se as funcionalidades acima referidas com mais pormenor.

- **Acesso baseado no conteúdo aos dados multimédia**

Devido ao modelo de dados baseado em objectos, a norma MPEG-4 permite o acesso e a organização dos dados baseado no conteúdo audiovisual de um modo eficiente. As ferramentas de acesso podem incluir indexação, hiperligação ou navegação. Um exemplo da aplicação desta funcionalidade é a procura de informação baseada no conteúdo em bibliotecas *on-line* ou bases de dados. Com o MPEG-4, é possível encontrar individualmente o objecto que se deseja e não apenas uma cena em que este se encontra integrado mas que não permite o seu acesso independente. Este tipo de acesso é fundamental para a reutilização de conteúdos ao nível do objecto.

- **Manipulação do conteúdo e edição do fluxo binário**

Com o novo modelo de dados, o utilizador pode interagir com o conteúdo ao nível do objecto ou do fluxo binário sem ser necessária transcodificação, ou seja decodificar, ainda que parcialmente, a cena para a voltar a codificar. Até agora, a interacção era muito limitada, resumindo-se à alteração de algumas características do conteúdo audiovisual, como alterar uniformemente a cor de toda a cena ou aumentar o volume do som, já que a cena era vista como um todo. Com a norma MPEG-4, o utilizador tem mais graus de liberdade em termos de interacção com o conteúdo, dispondo de funcionalidades tais como alterar o tamanho de um objecto para melhor visualização, reposicionar o objecto espacialmente e/ou temporalmente para personalização da disposição da cena, determinar a evolução de uma história consoante o objecto seleccionado ou até mesmo reutilizar o objecto noutra cena. O utilizador pode também obter informação adicional sobre um dado objecto de vídeo, por exemplo, o nome de uma personagem num filme ou os dados biográficos do actor seleccionado, accionando uma hiperligação com um dispositivo apontador sobre o objecto em questão. Outra funcionalidade possível é a inclusão de informação adicional numa cena, por exemplo durante o visionamento de um filme o utilizador pode ver simultaneamente a programação dos canais ou informação sobre a bolsa ou então o operador pode facilmente inserir publicidade nacional num programa que vem do estrangeiro.

- **Integração de conteúdo natural e sintético**

A integração eficiente de conteúdos naturais e sintéticos nunca foi contemplada nas normas de representação audiovisual já disponíveis, o que representa uma grande limitação dada a crescente quantidade de material audiovisual que inclui componentes naturais e sintéticos. A norma MPEG-4 possibilita a integração de modo eficiente de objectos audiovisuais de origem natural e sintética numa mesma cena, como, por exemplo, vídeo natural com forma arbitrária ou rectangular, texto, gráficos, som natural ou sintético, faces e corpos animados segundo modelos 3D, etc. Esta funcionalidade traz algo de novo para a representação audiovisual, sendo deste modo dado um passo na direcção da integração de todos os tipos de informação audiovisual (e não o mapeamento dos objectos sintéticos em vídeo, como até agora). Exemplos da utilização da integração de conteúdo natural e sintético podem encontrar-se em aplicações de realidade virtual ou animações que misturem actores reais com cenários e/ou actores virtuais. A Figura 4 mostra duas cenas compostas por objectos de origem natural e sintética.

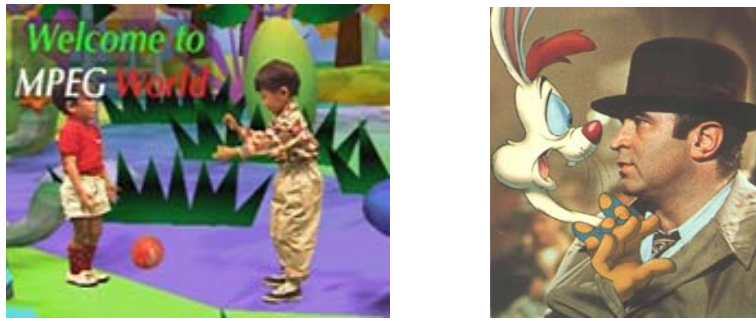


Figura 4 – Cenas compostas por objectos de origem natural e sintética.

- **Acesso aleatório temporal melhorado**

A norma MPEG-4 inclui métodos que permitem o acesso aleatório temporal eficiente a tramas ou objectos de uma sequência audiovisual de um modo rápido e com uma elevada granularidade (resolução fina). Com a utilização do acesso aleatório é possível, por exemplo, aceder aleatoriamente a dados audiovisuais a partir de um terminal remoto através de um canal com capacidade limitada ou efectuar um avanço rápido até um determinado instante de um objecto audiovisual numa sequência.

- **Eficiência de codificação melhorada**

A eficiência de codificação foi o objectivo fundamental nas normas MPEG-1 e MPEG-2, que permitiram o desenvolvimento da televisão digital e do DVD entre muitas outras aplicações e dispositivos importantes. Com a norma MPEG-4, para além da introdução de ferramentas de codificação mais eficientes, torna-se possível ajustar a técnica de codificação ao tipo de dados, o que permite que para cada objecto sejam escolhidas ferramentas de codificação ajustadas às suas características, aumentando a taxa de compressão para a mesma qualidade ou aumentando a qualidade para a mesma taxa de compressão. Por exemplo, o texto é codificado de um modo adequado às suas características e não como vídeo (como agora), o que permite alcançar excelente qualidade com um débito binário muito baixo. Para além disso, a expansão acelerada das redes móveis aumentou a necessidade de uma norma com eficiência de codificação melhorada, tendo a norma MPEG-4 como objectivo fornecer uma melhor qualidade audiovisual quando comparada com outras normas, utilizando débitos binários semelhantes³. Uma boa eficiência de codificação pode ser utilizada para transmissão eficiente de dados audiovisuais em canais de baixa largura de banda ou para armazenamento eficiente de dados em meios de capacidade limitada, tais como cartões magnéticos ou memórias de vários tipos.

- **Codificação integrada de fluxos binários concorrentes**

A norma MPEG-4 deveria permitir codificar de um modo eficiente várias vistas de um objecto, e.g. estereoscópico, e sincronizar os fluxos binários resultantes. Das oito funcionalidades inicialmente identificadas para a norma MPEG-4, esta foi a única não alcançada, não tanto pela complexidade associada mas pelo facto de não ter despertado grande interesse entre a comunidade científica e indústria. Para aplicações de vídeo estereoscópico ou com múltiplas vistas, a norma MPEG-4 deveria conter mecanismos que

³ É de referir que o consórcio 3GPP (*Third Generation Partnership Project*), responsável pelos sistemas móveis de terceira geração como o IMT2000 e UMTS, adoptou recentemente a norma MPEG-4 Visual para a codificação de vídeo – perfil *Simple* no nível zero.

permitissem explorar a redundância existente entre as várias vistas de um objecto. Esta funcionalidade forneceria representações eficientes de objectos naturais 3D, desde que estivessem disponíveis um número suficiente de vistas, e encontraria aplicação em ambientes multimédia (jogos de vídeo, aplicações de realidade virtual), apresentações multimédia, filmes estereoscópicos, etc.

- **Robustez em ambientes sujeitos a erros**

O acesso universal implica o acesso a conteúdo audiovisual utilizando diferentes tipos de redes (com ou sem fios) e de meios de armazenamento. Para o alcançar, a norma MPEG-4 dispõe de ferramentas que proporcionam robustez a erros, nomeadamente para comunicações a baixo débito sujeitas a condições de erros severas, como o são as comunicações móveis. O objectivo é maximizar a qualidade visual subjectiva na presença de erros, ou seja, minimizar o impacto negativo dos erros. A ideia não é substituir as técnicas de controlo de erros implementadas ao nível das redes (codificação de canal), mas sim fornecer resiliência a erros residuais (não corrigidos por estas técnicas), por exemplo através do aumento da capacidade de ressincronismo ou da ocultação (*concealment*) dos erros. Esta funcionalidade está claramente relacionada com ambientes de baixos débitos binários e com elevada taxa de erros, especialmente os móveis, e pode fornecer uma melhoria significativa em termos do impacto subjectivo final.

- **Escalabilidade baseada no conteúdo**

Escalabilidade significa a possibilidade de descodificar apenas uma parte do fluxo binário total, de acordo com os recursos do receptor, obtendo ainda assim uma representação útil da cena ou objecto codificados. Dado o mesmo fluxo binário, descodificadores com diferentes graus de capacidade computacional ou diferentes larguras de banda associadas podem ainda assim descodificar e mostrar o vídeo e o áudio de forma útil, ainda que não necessariamente com a mesma qualidade. Assim, um terminal MPEG-4 com menores recursos (em termos de banda ou capacidade computacional) pode descodificar apenas os objectos mais importantes – escalabilidade de conteúdo ou de objecto – e, eventualmente, cada um destes com maior ou menor qualidade ou resolução espacial ou temporal – escalabilidades de qualidade, espacial e temporal. Por outro lado, um descodificador com mais recursos consegue descodificar todo o fluxo que representa a cena, incluindo todos os objectos com a qualidade máxima. A norma MPEG-4 especifica ferramentas que oferecem escalabilidade de conteúdo, espacial e temporal, separadamente ou em combinação. Na escalabilidade espacial, as várias camadas de melhoria aumentam a resolução espacial da camada base, enquanto que a escalabilidade temporal aumenta a resolução temporal da primeira camada, tudo isto ao nível de cada objecto. As aplicações principais da escalabilidade baseada no conteúdo são a transmissão de vídeo através da Internet ou de terminais móveis, serviços de vídeo com qualidade variável, procura de vídeo em bases de dados com diferentes escalas, resoluções ou qualidades. Um exemplo paradigmático do uso desta funcionalidade é a difusão (*broadcasting*) de conteúdos através da Internet, onde o mesmo conteúdo deve poder ser utilizado de modo útil por variadíssimos tipos de utentes, em termos de capacidades do terminal e do canal.

A norma MPEG-4 não especifica normativamente soluções para alguns blocos do sistema global, em virtude da sua especificação normativa não ser essencial para a interoperabilidade entre terminais. Esta não especificação possibilita a competição e inovação por parte da

indústria, sem impacto na interoperabilidade, o que é extremamente importante para aumentar o “tempo de vida da norma” e a qualidade dos resultados obtidos. É o caso da análise de vídeo para segmentação, fundamental para se obterem objectos de vídeo a partir de cenas obtidas com uma câmara de vídeo, da detecção de movimento e do controlo de débito, todas técnicas indispensáveis para o bom desempenho de um sistema MPEG-4. Todos estes blocos não são especificados pela norma, continuando ainda hoje a evoluir e a distinguir o desempenho entre terminais de diferentes fabricantes.

2.3 Áreas de aplicação

A norma MPEG-4 encontra aplicação em muitas áreas. Como a experiência tem vindo a demonstrar, para além das aplicações que o grupo MPEG identificou como relevantes, muitas outras têm vindo a aparecer e a surpreender, geradas pela imaginação dos fabricantes, operadores e criadores de aplicações e pelo enorme potencial da tecnologia MPEG-4. De seguida, destacam-se algumas das mais importantes.

- **Vídeo na Internet**

A transmissão de vídeo através da Internet está a assumir um papel cada vez mais importante, como o demonstra a popularidade das notícias, filmes e dos concertos ao vivo que são transmitidos. Tal como acontece nos canais móveis, também aqui a largura de banda é muito limitada e variável, e pode existir perda de pacotes. Neste tipo de ambientes, uma melhor resiliência a erros e uma maior eficiência de codificação fornecidas pela norma MPEG-4 trazem enormes vantagens em termos da qualidade recebida. Para além disso, a escalabilidade do fluxo de dados, em termos de qualidade, resolução espacial ou temporal ou mesmo de objectos de vídeo, controlada pelo utilizador, permite que se possa obter informação útil mesmo com poucos recursos. Esta parece ser, sem dúvida, uma das áreas onde a norma MPEG-4 “explodirá” mais rapidamente.

- **Serviços móveis multimédia**

A enorme popularidade dos telefones móveis e dos dispositivos tipo *palm-pilot* evidencia o grande interesse provocado por este tipo de equipamento, e pelas correspondentes aplicações, nos utilizadores. As aplicações nesta área têm muito a ganhar com a introdução de informação multimédia. Devido às suas características, os meios móveis apresentam limitações, tais como baixa largura de banda, capacidade computacional limitada e reduzida fiabilidade na transmissão devido aos erros introduzidos pelo canal. A norma MPEG-4 fornece as ferramentas para tornar as aplicações multimédia em terminais móveis uma realidade, nomeadamente, através da resiliência a erros e escalabilidade do conteúdo. Com a recente adopção da norma MPEG-4 Visual pelo consórcio 3GPP, espera-se que os conteúdos MPEG-4 venham a ter um papel preponderante no futuro do UMTS.

- **Produção de televisão**

A criação de conteúdos para televisão caminha na direcção da utilização de técnicas de produção virtuais, em conjunto com as bem conhecidas técnicas de “fundo azul”. O fundo e os actores são gravados separadamente e podem ser misturados, juntando-lhes ainda efeitos especiais, gerados por computador. Se forem codificados objectos de vídeo em vez de tramas rectangulares, a composição e reutilização de conteúdos torna-se mais flexível. Os programas

de televisão que sejam compostos por objectos de vídeo, com gráficos e texto adicionais, podem ser transmitidos como tal para o espectador, com a vantagem de permitir o controlo e personalização dos programas de um modo mais sofisticado. Por outro lado, a reutilização de conteúdos/objectos permite que sejam produzidos programas de um modo mais fácil e rápido e de uma maneira mais criativa.

- **Televisão digital interactiva**

O grande crescimento da Internet provocou um aumento do interesse pela interactividade com o conteúdo que a televisão digital pode proporcionar. Texto, imagens, áudio e gráficos podem ser manipulados pelo utilizador para personalizar o conteúdo, por exemplo alterando a disposição dos objectos no ecrã, fornecendo informação adicional ao programa em questão, por exemplo a biografia do actor, ou mesmo fornecendo informação não relacionada com o programa em visualização mas com interesse para o espectador. Neste último caso podem-se incluir logótipos das televisões, publicidade escolhida para diferentes tipos de espectador, estatísticas de desporto ou as cotações da bolsa em tempo real. Para além destas funcionalidades, existem mecanismos de hiperligação que permitem, por exemplo, obter dados sobre determinado jogador numa transmissão desportiva ou sobre um actor num filme, por exemplo acedendo à sua página pessoal na Internet (se existir um canal de retorno) ou a objectos MPEG-4 adicionais que contenham esta informação (note-se que os objectos textuais e gráficos são, normalmente, extremamente baratos em termos de débito).

- **Jogos**

A popularidade dos jogos de vídeo em computadores pessoais ou em consolas demonstra o interesse existente pela interacção entre utilizador e conteúdo. A maioria dos jogos utilizam gráficos tridimensionais quer para o ambiente do jogo quer para as personagens controladas pelo jogador. A integração de objectos de vídeo tornaria estes jogos ainda mais realistas. O acesso a objectos de vídeo e a utilização de técnicas normativas de codificação tornam possível a personalização dos jogos, por exemplo através da utilização de bases de dados de vídeo personalizadas ligadas aos jogos em tempo real.

2.4 Organização da norma MPEG-4

Devido aos objectivos ambiciosos com os quais foi concebida, a norma MPEG-4 especifica um numeroso conjunto de ferramentas relacionadas com várias áreas tecnológicas. Para facilitar o seu uso na implementação de produtos de acordo com a norma, esta está organizada em sete partes, que podem ser utilizadas em conjunto ou separadamente:

- **Parte 1: Sistema** – Especifica a multiplexagem, o formato de descrição de cena (BIFS – *BInary Format for Scenes*), a sincronização dos fluxos elementares, informação relativa ao conteúdo (OCI), protecção da propriedade intelectual (IPMP), suporte para animação facial e corporal, etc. [8].
- **Parte 2: Visual** – Especifica a representação codificada de objectos visuais, naturais e sintéticos [3].
- **Parte 3: Áudio** – Especifica a representação codificada dos objectos áudio, naturais e sintéticos que não estão no formato BIFS [9].

- **Parte 4: Conformidade** – Define os testes para verificar se um decodificador ou fluxo binário está conforme a norma [10].
- **Parte 5: *Software de referência*** – Contém programas que implementam as partes 1, 2, 3 e 6 da norma (ferramentas normativas e não normativas) [11].
- **Parte 6: DMIF (*Delivery Multimedia Integration Framework*)** – Especifica o protocolo de sessão para os diversos tipos de rede que permitem o acesso das aplicações a fluxos MPEG-4 [12].
- **Parte 7: *Software otimizado para ferramentas MPEG-4*** – Inclui programas que implementam de forma otimizada algumas ferramentas normativas e não normativas de codificação como a detecção de movimento.

As partes 1, 2, 3 e 6 da norma especificam o núcleo da tecnologia MPEG-4, enquanto que as partes 5, 6 e 7 são partes de suporte. As partes 1, 2 e 3 são independentes do tipo de rede utilizada, cabendo à parte 6 estabelecer os protocolos para utilizar os fluxos gerados em vários tipos de redes.

2.5 Estrutura e sintaxe da norma MPEG-4 Vídeo

Como foi referido anteriormente, o conceito principal da norma MPEG-4 é o de objecto audiovisual, que é o elemento principal do modelo de dados baseado em objectos. Este tipo de representação é apropriado para aplicações interactivas e proporciona o acesso de um modo simples ao conteúdo das cenas. No contexto desta tese só se consideram objectos de vídeo, mas os princípios permanecem válidos para outros tipos de objectos audiovisuais.

Um objecto de vídeo pode ser constituído por uma ou mais camadas escaláveis, e.g. em termos de qualidade, espaço ou tempo. Utilizando o conceito de escalabilidade, um objecto pode ser decodificado partindo da camada base, à qual são acrescentadas outras camadas que o melhoram sucessivamente. Esta abordagem permite que, para cada objecto, seja gerado um conjunto eficiente de fluxos binários que é útil para uma grande variedade de larguras de banda e complexidades computacionais dos receptores. Estes fluxos não são independentes, já que cada camada acrescenta informação à camada anterior para melhorar a qualidade final do objecto.

Numa cena de vídeo MPEG-4, cada objecto de vídeo é caracterizado pela sua posição espacial e temporal, pela sua forma e pela sua textura. Para as aplicações que não necessitam de objectos de vídeo de forma arbitrária, devido à complexidade acrescida ou à dificuldade que existe em gerá-los, a norma MPEG-4 permite a codificação de objectos rectangulares, que constituem um caso particular de objectos de forma arbitrária.

Os fluxos binários correspondentes a cada um dos objectos numa dada cena seguem uma estrutura hierárquica, como se mostra na Figura 5.

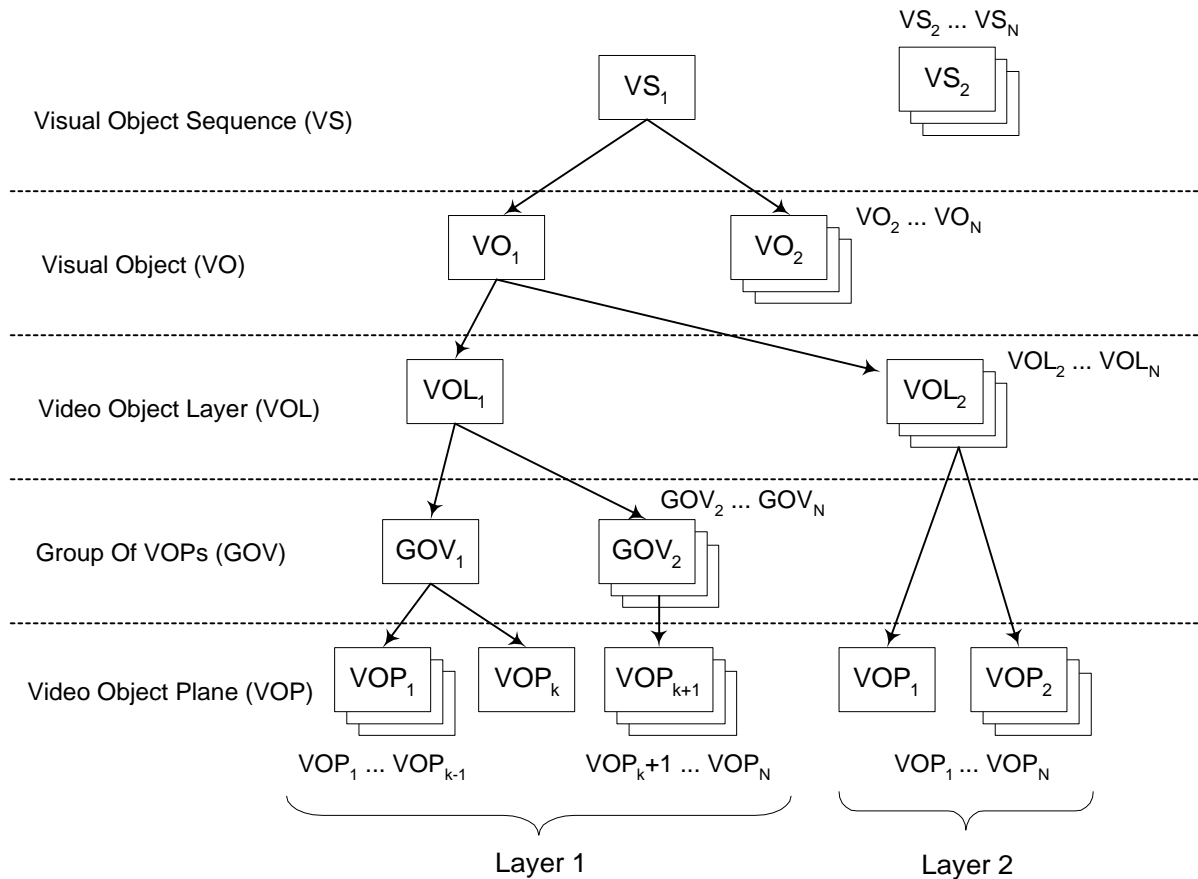


Figura 5 – Estrutura hierárquica de um fluxo binário de vídeo MPEG-4.

Os níveis hierárquicos que descrevem os objectos visuais numa cena são os seguintes:

- **Visual Object Sequence (VS)** – É a estrutura sintáctica que está no topo da hierarquia do fluxo codificado. Inclui objectos visuais 2D ou 3D, naturais ou sintéticos, e as respectivas camadas de melhoramento. Este nível hierárquico indica quais os objectos visuais presentes na cena e qual o perfil e o nível dessa cena.
- **Visual Object (VO)** – Um objecto visual é uma entidade presente na cena que o utilizador pode aceder e manipular. Pode ser um objecto visual (de origem natural ou sintética), uma textura estática, uma malha 2D ou 3D ou um objecto facial 3D. Habitualmente, um objecto visual corresponde a um objecto com significado semântico ou ao fundo de uma cena. Este nível hierárquico serve para indicar qual o tipo de objecto visual em questão e quais as camadas escaláveis que o constituem.
- **Video Object Layer (VOL)** – A sintaxe da codificação de vídeo na norma MPEG-4 inclui codificação escalável (várias camadas) e não escalável (uma única camada). Cada camada de um objecto de vídeo designa-se por VOL. A escalabilidade permite a reconstrução de um objecto partindo da camada de base (que pode ser decodificada sozinha) à qual são adicionadas camadas de melhoramento. Como já foi referido, um objecto de vídeo pode ser codificado utilizando escalabilidade temporal e espacial (a escalabilidade de qualidade pode ser obtida como um caso particular da escalabilidade de espaço), oferecendo assim vários níveis de resolução. Consoante a largura de banda ou a capacidade computacional, o receptor pode adequar a resolução

do vídeo recebido às suas características. Este nível contém toda a informação referente a uma camada de um objecto de vídeo, nomeadamente os parâmetros de codificação e o fluxo binário.

- **Group of Video Object Planes (GOV)** – Os GOVs agrupam VOPs e são essenciais para o acesso aleatório à informação de vídeo. No início de cada GOV existem tramas I onde o VOP é codificado de modo independente, sem necessitar de tramas anteriores ou posteriores, permitindo deste modo que existam pontos de acesso aleatório para que se possa decodificar só uma dada parte do fluxo binário.
- **Video Object Plane (VOP)** – Uma instância de um objecto de vídeo num determinado instante de tempo denomina-se por VOP. O processo de codificação gera representações codificadas de VOPs, juntamente com informação de composição que permite compô-los numa cena. A norma MPEG-4 suporta três modos de codificação para VOPs, ilustrados na Figura 6:
 - **I-VOP** – Um VOP pode ser codificado independentemente de qualquer outro VOP (*Intra* VOP ou I-VOP);
 - **P-VOP** – Um VOP pode ser codificado utilizando predição (eventualmente através da compensação de movimento) com base noutra VOP decodificado anteriormente. A estes VOPs chamam-se *Predicted* VOP ou P-VOP.
 - **B-VOP** – Um VOP pode ser codificado utilizando como predição VOPs passados e/ou futuros, em termos de apresentação. Estes VOPs chamam-se *Bidirectional* VOPs ou B-VOPs.

Um VOP é codificado utilizando macroblocos de dimensão 16×16 *pixels* de luminância. Cada um destes macroblocos é dividido em blocos de dimensão 8×8 que contêm componentes de luminância e de crominância, podendo a crominância ser sub-amostrada espacialmente. Assim, para o formato 4:2:0 (crominâncias com metade da resolução ao longo das linhas e das colunas), cada macrobloco contém 4 blocos de luminância e 2 blocos de crominância (um para cada uma das crominâncias).

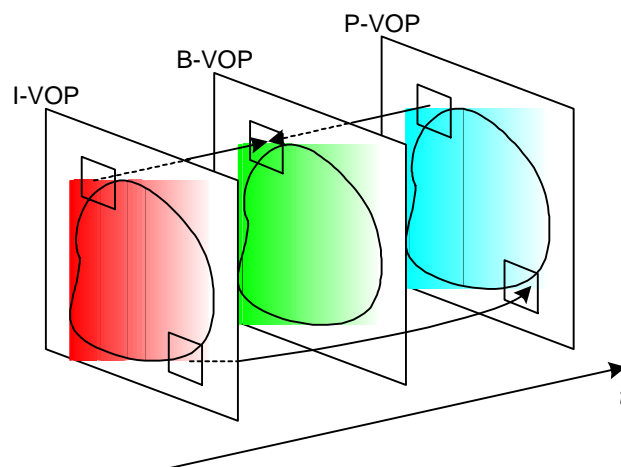


Figura 6 – Modos de codificação de um VOP: I, P e B.

Na norma MPEG-4 foi adoptado um novo modelo de dados mais flexível, onde a informação é estruturada como uma composição de objectos de vários tipos, nomeadamente, objectos de vídeo rectangulares ou de forma arbitrária, texto, gráficos, modelos tridimensionais de faces e corpos e objectos de voz e áudio genérico. Esta estrutura implica que exista informação adicional de composição de cena para além da informação relativa aos vários objectos. A informação de composição de cena inclui a descrição de quais os objectos que compõem a cena e as suas relações espaciais e temporais.

Para possibilitar o acesso aos vários objectos de forma independente, os dados codificados são organizados em termos de fluxos elementares. Na norma MPEG-4, toda a informação é transmitida através de fluxos elementares que contêm a representação codificada de um objecto audiovisual, a informação de composição da cena ou informação de controlo para, por exemplo, animar um modelo facial 3D. Os fluxos elementares são identificados e caracterizados por descritores de objecto [8], que contêm informação acerca do tipo de conteúdo e dos recursos necessários para a sua decodificação. Os descritores de objecto também são utilizados para o caso de existirem representações alternativas (com maior ou menor débito binário e logo qualidade) ou codificação escalável usando múltiplos fluxos binários para o mesmo objecto. A informação mais importante contida no descritor de objecto é a necessária para identificar a localização dos dados (fluxo codificado), em termos do canal lógico de transporte ou de um URL (*Uniform Resource Locator*). Os descritores de objecto também podem endereçar um fluxo binário com descrição do conteúdo (OCI – *Object Content Information*) ou incluir informação relacionada com a gestão e protecção da propriedade intelectual (IPMP – *Intellectual Property Management & Protection*) associada ao objecto e à cena em questão. Todos estes mecanismos são especificados no contexto da norma MPEG-4 Sistemas [8].

A Figura 7 mostra uma cena composta por vários tipos de objectos. A mulher é um objecto audiovisual que inclui um objecto de vídeo de forma arbitrária e um objecto de áudio correspondente à sua voz. A secretária e o globo são modelos 3D gerados por computador e o ecrã da apresentação é um objecto de vídeo rectangular com som estereofónico. A descrição de cena segue uma estrutura hierárquica, que pode ser representada por uma árvore (Figura 8).

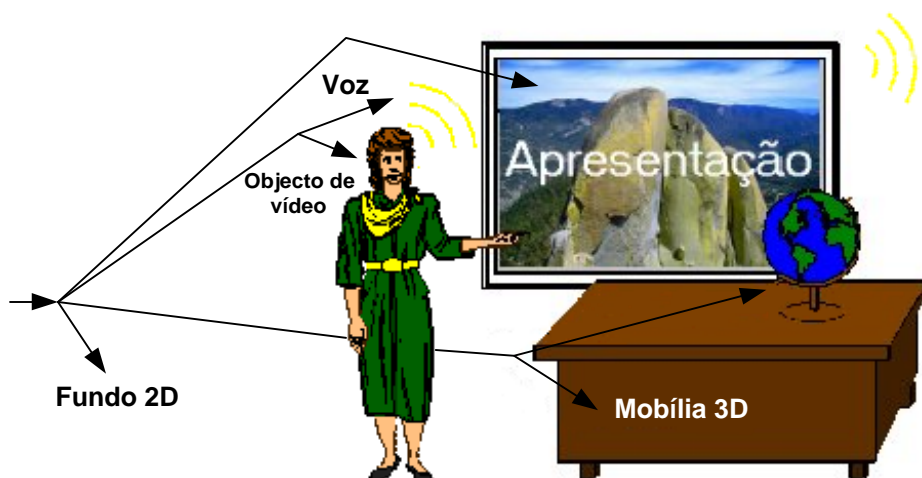


Figura 7 – Exemplo de uma cena com vários objectos.

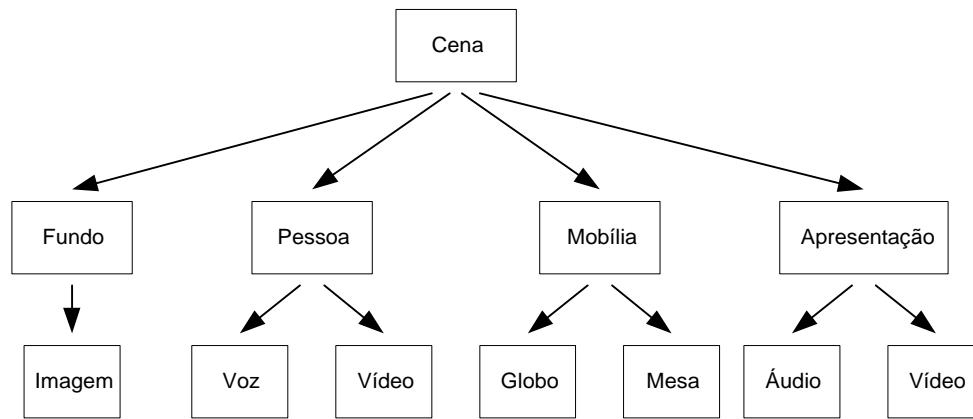


Figura 8 – Árvore correspondente à descrição da cena na Figura 7.

O formato de descrição de cena especificado pela norma MPEG-4 Sistema é designado por *Binary Format for Scenes* (BIFS). Este formato permite descrever a composição de uma cena através das relações espaciais e temporais dos vários objectos e também a sua evolução dinâmica; para além disso permite também criar novos objectos gráficos ou de texto e inseri-los na cena. Os comandos de BIFS podem ser utilizados para colocar objectos na cena ou para mudar dinamicamente as propriedades visuais ou acústicas de um objecto sem o modificar explicitamente, como por exemplo mudar a cor de um cubo 3D. Também podem ser utilizados para definir o comportamento de objectos, quer seja de um modo incondicional (por exemplo, colocar uma bola a rodar num instante pré-determinado) ou como resposta à acção do utilizador. O formato BIFS foi definido a partir do formato VRML (*Virtual Reality Modeling Language*) [13], tendo-lhe acrescentado inúmeras funcionalidades. Em contraste com o formato VRML que é textual, o formato BIFS é binário, o que é essencial para ambientes que exijam grande eficiência em termos de fluxo binário. O formato BIFS também acrescenta ao formato VRML o conceito de fluxo em tempo real. Ao contrário do VRML, onde uma cena tem de ser completamente transferida do servidor para o cliente e só depois apresentada, o formato BIFS permite que as cenas MPEG-4 possam ser mostradas em tempo real à medida que vão sendo recebidas de acordo com a largura de banda disponível (aqui a escalabilidade tem um papel importante). Este modo de funcionamento é conhecido por *streaming* (em alternativa ao modo *storage*).

2.6 Ferramentas de codificação de vídeo

Como já foi referido, as principais funcionalidades da norma MPEG-4 são a interactividade baseada no conteúdo, eficiência de compressão e acesso universal. As ferramentas de codificação de vídeo que proporcionam estas funcionalidades podem ser agrupadas em: codificação de forma, compensação de movimento, codificação de textura, *sprites*, codificação de texturas estáticas, resiliência a erros e escalabilidade. De seguida, descrevem-se as principais ferramentas de codificação de vídeo incluídas na versão 1 da norma MPEG-4 Visual, cuja descrição mais detalhada pode ser encontrada em [3][14].

2.6.1 Codificação de forma

A norma MPEG-4 é a primeira norma a codificar objectos de vídeo com forma arbitrária e, por isso, inclui ferramentas para a codificação da informação de forma. Existem dois tipos de informação de forma que são considerados como características intrínsecas de um objecto de vídeo: forma binária e forma multi-nível.

A forma binária é representada por uma matriz com a dimensão do menor rectângulo que envolve o objecto (*bounding box*), múltiplo de 16×16 *pixels*, e cujos elementos podem ter dois valores possíveis, consoante o *pixel* está dentro ou fora do objecto. A matriz de forma binária é dividida em blocos de dimensão 16×16 denominados BAB (*Binary Alpha Blocks*) e cada BAB pode ser codificado de vários modos, dependendo de os *pixels* que engloba serem transparentes ou opacos ou serem ou não utilizados vectores de movimento. A informação de forma binária é codificada através de uma técnica denominada *Context based Arithmetic Encoding* (CAE) [15][16], baseada na codificação de blocos (BAB) com compensação de movimento, e que permite codificação com ou sem perdas. Neste algoritmo, é calculado um contexto para cada *pixel*, baseado nos *pixels* vizinhos, que podem pertencer ao mesmo VOP (*IntraCAE*, Figura 9a) ou ao VOP anterior (*InterCAE*, Figura 9b). O contexto calculado serve para indexar uma tabela de probabilidades cujo valor é utilizado na posterior codificação aritmética.

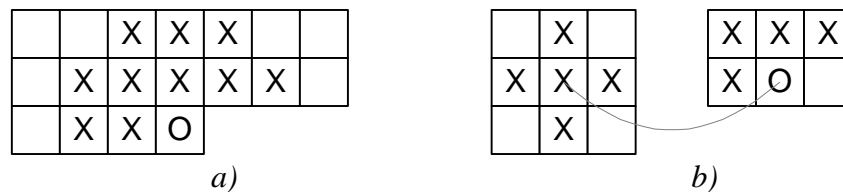


Figura 9 – Codificação aritmética baseada no contexto (CAE) no modo Intra (a) e Inter (b). O *shapel* a ser codificado está marcado com um círculo e os *shapels* que contribuem para o cálculo do contexto estão marcados com uma cruz.

A informação de forma multi-nível (transparência variável e não apenas completamente transparente ou completamente opaca como em cima) segue a mesma estrutura da informação de forma binária, mas cada *pixel* (elemento da matriz de forma) pode ter associado um entre vários valores (habitualmente entre 0 e 255) que representam o grau de transparência desse *pixel*. O valor 0 é utilizado para representar um *pixel* completamente transparente, enquanto que o valor 255 corresponde a um *pixel* completamente opaco. A informação de forma multi-nível é codificada através do processamento em separado do suporte (*pixels* onde o valor da forma é superior a 0, também designados por *shapels*) como uma forma binária e do valor da transparência dos seus *shapels* como valores de luminância. A forma binária é codificada com as ferramentas descritas acima. Os valores da transparência são codificados como macroblocos de luminância de uma textura com forma arbitrária, ou seja utilizando a transformada DCT.

2.6.2 Estimação e compensação de movimento

Como nas normas MPEG anteriores, as técnicas de estimação e compensação de movimento são utilizadas na codificação de textura (e agora também na forma) e destinam-se a comprimir sequências de vídeo através da exploração da redundância temporal entre tramas. A norma MPEG-4 utiliza uma abordagem semelhante às normas MPEG anteriores. As principais diferenças advêm do facto de as técnicas de estimação e compensação de movimento terem de ser adaptadas para lidar com objectos de forma arbitrária, representados por sequências de VOPs.

A estimação de movimento só é necessária para P-VOPs e B-VOPs, pois são estes tipos de VOPs que necessitam de predição. Para os macroblocos completamente fora do VOP mas dentro da *bounding box* – macroblocos transparentes – não é efectuada estimação de movimento. Para os macroblocos totalmente dentro do VOP – macroblocos opacos –, a estimação de movimento é efectuada de maneira semelhante às outras normas, através do emparelhamento de macroblocos de dimensão 16×16 (na luminância e dos correspondentes blocos de dimensão 8×8 nas crominâncias), resultando um vector de movimento por macrobloco, ou através do emparelhamento de blocos de dimensão 8×8 (na luminância e correspondentes crominâncias), resultando um vector de movimento por cada um dos quatro blocos de 8×8 *pixels* de luminância no macrobloco de 16×16 *pixels*. Todos estes vectores podem ser determinados com precisão de meio *pixel*. Os *pixels* de referência utilizados na estimação de movimento provêm de I-VOPs ou P-VOPs já (des)codificados e que podem corresponder a instantes de apresentação passados ou futuros em relação ao VOP considerado.

A estimação de movimento para os macroblocos que estão apenas parcialmente dentro do VOP é feita recorrendo a uma técnica de emparelhamento modificada. O critério para o cálculo do erro de emparelhamento não é normativo; contudo, um critério bastante usado é a soma do valor absoluto das diferenças dos *pixels* que estão dentro do VOP com os *pixels* correspondentes no macrobloco em análise, no VOP de referência. Como alguns dos *pixels* no macrobloco de referência podem estar fora do VOP de referência, é utilizada uma técnica de preenchimento (*padding*) para extrapolar o valor destes *pixels* a partir dos *pixels* que estão dentro do VOP de referência. Para lidar com movimentos de maior amplitude, uma técnica semelhante é utilizada para estender o VOP de referência para além dos macroblocos de fronteira (*extended padding*). Esta técnica melhora a eficiência pois permite que existam mais possibilidades na procura de vectores candidatos para a predição na fronteira do VOP de referência. A Figura 10 mostra um exemplo da utilização destes dois tipos de *padding*.

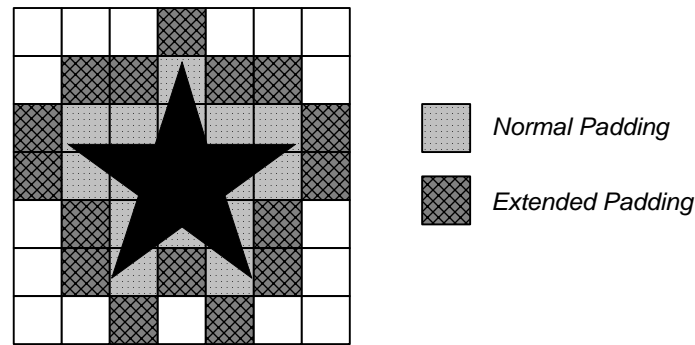


Figura 10 – Exemplo da utilização de padding para a estimação de movimento.

Para P-VOPs e B-VOPs, os vectores de movimento necessários para a sua predição são transmitidos no fluxo binário. A codificação dos vectores é diferencial, sendo baseada na mediana de três vectores de movimento vizinhos já transmitidos (dos macroblocos em cima, à esquerda e no canto superior esquerdo do macrobloco corrente). Se a estimação de movimento for baseada em blocos (em vez de macroblocos), utilizam-se nesse caso quatro vectores de movimento por macrobloco (um por cada bloco) para a compensação de movimento.

A norma MPEG-4 também suporta compensação de movimento com sobreposição (como na norma H.263). Neste caso, para cada bloco de um macrobloco, são usados três vectores de movimento (do próprio bloco (ou macrobloco) e dos seus dois vizinhos mais próximos). A predição para o bloco (ou macrobloco) em questão é obtida através da média das predições dadas pelos vários vectores candidatos, ponderada de acordo com pesos pré-definidos e que dependem da posição de cada *pixel* dentro do bloco (ou macrobloco).

2.6.3 Codificação de textura

Para codificar a informação de textura é utilizada a transformada DCT (*Discrete Cosine Transform*) aplicada sobre blocos de 8×8 *pixels*, já conhecida das normas anteriores, mas adaptada agora de forma a lidar também com objectos de forma arbitrária, como se descreve de seguida. A DCT é aplicada separadamente à luminância e às crominâncias de um dado macrobloco, totalizando seis blocos de dimensão 8×8 : quatro da luminância e um para cada uma das componentes de crominância para formatos 4:2:0. Tal como na estimação e compensação de movimento, existem três tipos de macroblocos:

1. **Transparentes** – Os macroblocos que estão completamente fora do VOP (mas dentro da *bounding box*) não são codificados, já que não contêm informação de textura.
2. **Opacos** – Os macroblocos que se encontram totalmente dentro do VOP são codificados com a técnica convencional da DCT.
3. **Fronteira** – Para os macroblocos que se encontram na fronteira do VOP, é necessário considerar a posição dos seus blocos. Se um bloco de 8×8 *pixels* se encontra parcialmente dentro do VOP, os *pixels* dentro desse bloco mas fora do VOP são preenchidos (*padding*), extrapolando o seu valor a partir dos *pixels* desse bloco que estão dentro do VOP, enquanto que se o bloco estiver completamente fora do VOP, os seus *pixels* são colocados a zero [3]. Depois deste processo, esses blocos são

codificados do mesmo modo que os blocos que estão dentro do VOP ou seja usando a DCT.

A Figura 11 mostra um exemplo de um VOP com forma arbitrária que contém os três tipos de macroblocos referidos.

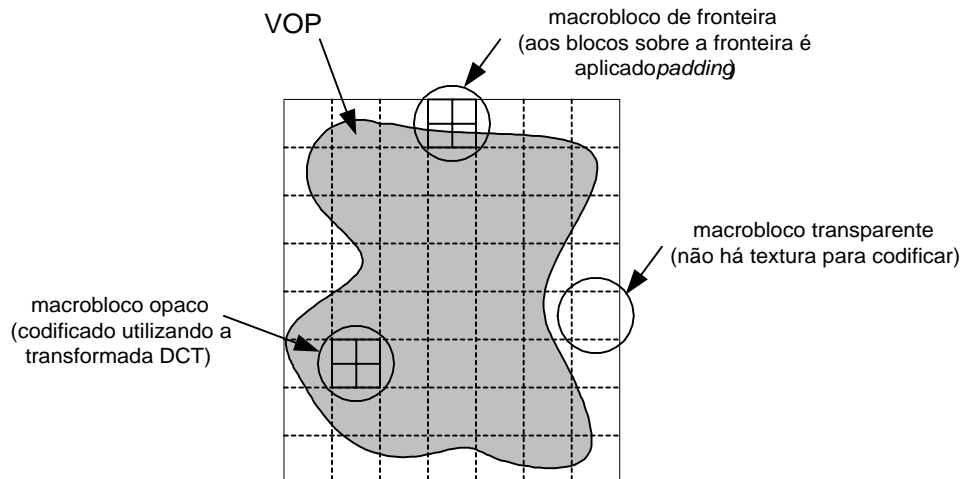


Figura 11 – Exemplo dos vários tipos de macroblocos existentes num VOP de forma arbitrária.

Após a transformada, os coeficientes da DCT são quantificados e depois varridos em *zig-zag* com vista a gerar símbolos que asseguram sempre a transmissão prioritária dos coeficientes mais relevantes, independentemente do número de coeficientes enviados, pois a maior parte da energia está concentrada no canto superior esquerdo (baixas frequências) de cada bloco. Pode também usar-se codificação preditiva para os coeficientes AC e DC da DCT, baseada no bloco imediatamente à esquerda ou acima. Finalmente, os símbolos bidimensionais contendo a posição e amplitude quantificada dos coeficientes a enviar são codificados com codificação entrópica de Huffman.

2.6.4 Sprites

Uma *sprite* é, tipicamente, uma imagem de grande dimensão que resulta da combinação de vários instantes de um objecto de vídeo. Assim, corresponde a uma imagem estática que só é transmitida uma vez no início da transmissão ou construída ao longo desta e da qual apenas é mostrada uma parte em cada instante, reunindo a informação espacial e temporal de um modo muito compacto. Um bom exemplo é o de um fundo durante um movimento de câmara ou um logótipo animado. A Figura 12 mostra um exemplo de uma *sprite*, que neste caso corresponde ao fundo de uma parte da sequência *Stefan*. Algumas partes do *sprite* podem não ser visíveis em algumas tramas devido a objectos que se sobreponham à imagem ou devido ao movimento de câmara. As *sprites* são utilizadas para reconstruir fundos estáticos ou como predição para fundos não estáticos, e são codificadas em modo *Intra*, ou seja, independentemente de outros instantes temporais e sem compensação de movimento. Os parâmetros que indicam qual a parte da *sprite* que é mostrada em cada trama são enviados no fluxo binário na forma de vectores de movimento. Também existem parâmetros de

transformação que indicam como a *sprite* se deve deformar para ser adequadamente usada para cada VOP.



Figura 12 – Exemplo de uma *sprite* para a sequência Stefan.

2.6.5 Codificação de texturas estáticas

A norma MPEG-4 suporta o mapeamento de texturas estáticas e de vídeo em malhas 2D ou 3D e especifica o modo eficiente de codificar este tipo de informação. A técnica de codificação de texturas estáticas utiliza a transformada de *wavelets* discreta (DWT) [3][17] para comprimir a informação de textura de um modo eficiente (solução semelhante à adotada na recente norma JPEG2000 [18]). As componentes de textura (luminância e crominâncias) são tratadas separadamente, sendo cada uma delas decomposta em bandas de frequência através de filtros recursivos. Devido a esta decomposição de frequências, a transformada de *wavelets* proporciona uma elevada granularidade em termos de escalabilidade espacial, nomeadamente superior às técnicas baseadas na DCT adoptadas para a codificação de vídeo. É esta, aliás, a principal razão que levou à adopção da transformada de *wavelets* a par da DCT na norma MPEG-4 Visual. A decomposição em frequência resulta numa árvore de coeficientes, cujos elementos são codificados utilizando codificação aritmética. Uma descrição completa do algoritmo pode ser encontrada em [19].

2.6.6 Resiliência a erros

A norma MPEG-4 inclui ferramentas que oferecem resiliência a erros acrescida [20], uma funcionalidade importante para se obter acesso universal em ambientes sujeitos a erros, nomeadamente em ambientes móveis. Existem quatro ferramentas relacionadas com a resiliência a erros na norma MPEG-4 Visual: ressincronização, partição de dados, código de extensão do cabeçalho e códigos reversíveis de comprimento variável (RVLC).

- **Ressincronização** – Normalmente, quando os dados de vídeo comprimidos são enviados através de um canal são introduzidos erros no fluxo binário. Um decodificador que processe este fluxo com erros pode perder o sincronismo sintáctico de decodificação, e se não forem tomadas medidas adequadas o vídeo deixa de poder ser decodificado⁴. Para resolver este problema, a ressincronização é a técnica mais

⁴ É importante salientar que todas as normas de codificação definem o comportamento normativo do decodificador supondo uma situação ideal ou seja fluxos binários sem erros. O comportamento do decodificador na presença de erros de canal não é definido normativamente.

utilizada e consiste em inserir marcadores únicos no fluxo binário que permitem ao decodificador recuperar o sincronismo em caso de erro. Estes marcadores consistem em palavras de comprimento fixo que não podem ser emuladas por qualquer outra combinação de palavras de código no fluxo binário. Quando detecta um erro (usando regras mais ou menos sofisticadas mas não normativas), o decodificador analisa os bits até encontrar um marcador (ponto de ressincronismo) e recomeça a decodificar a partir desse ponto (perdendo-se apenas a informação entre o erro detectado e o ponto de ressincronismo). Assim, a norma MPEG-4 divide o fluxo de vídeo codificado em pacotes correspondentes a um número inteiro (mas não fixo) de macroblocos e especifica a inserção de marcadores de ressincronização no início de cada pacote. Esta abordagem é diferente da adoptada pelas normas H.261 e H.263, que só permitem a ressincronização após um número constante de macroblocos (tipicamente uma linha de macroblocos).

- **Partição de dados** – Depois de detectar um erro no fluxo binário e se ressincronizar no próximo marcador de sincronismo, o decodificador isolou a área possivelmente afectada pelos erros, que está nos macroblocos que se encontram entre os dois marcadores, uma vez que não se sabe bem onde ocorreu precisamente o erro depois do marcador de sincronismo anterior. Como o processo de ocultação ou cancelamento de erros não é normativo, processo este que leva à diminuição do impacto subjectivo negativo dos erros, o que o decodificador faz nesta zona da imagem varia de implementação para implementação. Uma das técnicas mais simples para minimizar o impacto dos erros é substituir os macroblocos não decodificados (errados) por macroblocos do VOP anterior. A partição de dados especificada pelo MPEG-4 separa a informação de movimento da informação de textura, ou seja, os vectores de movimento dos coeficientes DCT, através de um marcador especial de movimento. Este marcador permite uma localização mais precisa do erro, antes ou depois do marcador de movimento, e consequentemente uma ocultação mais eficiente. Se o erro ocorre na informação de movimento, o decodificador substitui normalmente todos os macroblocos do pacote em questão pelos macroblocos correspondentes do VOP anterior. Quando um erro é detectado na parte da textura, o decodificador pode utilizar os vectores de movimento recebidos correctamente para obter melhores previsões para os macroblocos não decodificados, a partir da imagem anteriormente decodificada.
- **Código de extensão do cabeçalho** – Os dados presentes no cabeçalho dos pacotes de vídeo constituem informação essencial para o decodificador ser capaz de decodificar o fluxo binário, nomeadamente a posição espacial e informação temporal dos dados de vídeo e o modo como o objecto de vídeo foi codificado (por exemplo, *Intra* ou *Inter*). Se alguma desta informação estiver errada, o decodificador não pode de nenhum modo decodificar correctamente o VOP. Para reduzir a sensibilidade a erros destes dados, é possível adicionar informação redundante aos cabeçalhos dos pacotes. Assim, em cada pacote, existe a possibilidade de repetir no seu cabeçalho, parâmetros que permitem que o pacote seja decodificado de um modo independente. Se a informação presente no cabeçalho do VOP estiver corrompida por um erro de transmissão, pode ser corrigida pela informação adicional presente no cabeçalho do pacote e este pode

ser decodificado. O decodificador pode detectar um erro no cabeçalho do VOP se a informação decodificada não for sintáctica ou semanticamente consistente.

- **Códigos reversíveis de comprimento variável** – A utilização de códigos de comprimento variável (VLC) para transmitir vídeo codificado, de forma eficiente, em canais com erros traz problemas para a decodificação. Se durante o processo de decodificação o decodificador detecta um erro, perde o sincronismo e tem de ignorar todos os dados até ao próximo ponto de ressincronização. Para limitar o impacto negativo dos erros, é possível utilizar códigos reversíveis que podem ser decodificados em qualquer das direcções, ou seja de trás para a frente e vice-versa. Quando o decodificador detecta um erro enquanto está a decodificar na direcção normal (de trás para a frente), salta até ao próximo marcador de ressincronismo e começa a decodificar o fluxo binário na direcção oposta (da frente para trás) até encontrar novamente um erro (Figura 13). Baseado na posição dos dois erros detectados, o decodificador pode recuperar alguns dos dados de vídeo, junto ao marcador de ressincronismo mais adiantado, que seriam perdidos se não fosse utilizada esta técnica.

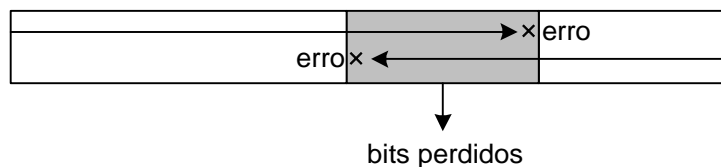


Figura 13 – Decodificação utilizando códigos reversíveis de comprimento variável.

A Figura 14 mostra o esquema do fluxo binário onde são salientadas as técnicas de resiliência a erros atrás referidas.

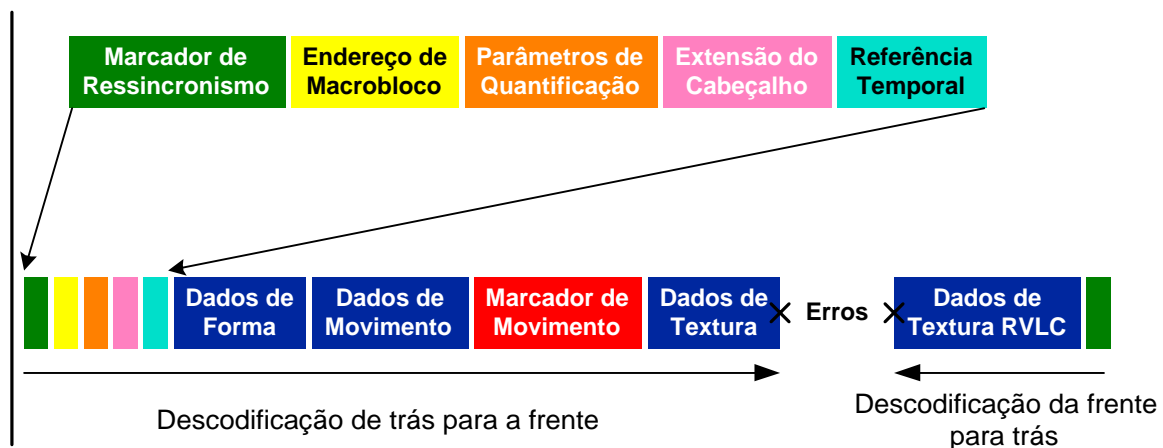


Figura 14 – Técnicas de resiliência a erros na norma MPEG-4 Visual.

2.6.7 Escalabilidade

A norma MPEG-4 versão 1 suporta escalabilidade temporal e espacial para a textura. Para se criar um fluxo de vídeo escalável, a sequência é codificada em duas ou mais camadas:

uma camada de base codificada de modo independente e uma ou mais camadas de melhoramento codificadas como melhoramentos em relação às camadas anteriores. A escalabilidade permite que existam disponíveis simultaneamente várias resoluções espaciais e temporais para o mesmo objecto de vídeo, sem que isso implique a codificação repetida do mesmo conteúdo. Deste modo, a partir de material codificado correspondente a um dado objecto, um decodificador de vídeo mais simples pode decodificar e produzir vídeo com uma qualidade básica e um decodificador mais evoluído pode produzir vídeo com uma qualidade melhor.

A escalabilidade temporal permite ao decodificador aumentar a resolução temporal da sequência de vídeo combinando os VOPs decodificados da camada de base com os VOPs decodificados das camadas de melhoramento, ou seja, as camadas de melhoramento acrescentam informação que é visualizada entre as tramas da camada de base. A escalabilidade espacial permite aumentar a resolução espacial de uma sequência de vídeo através da combinação das camadas de melhoramento com a camada de base de resolução inferior.

O mecanismo de escalabilidade da norma MPEG-4 utiliza um tipo de B-VOPs que só existem na camada de melhoramento. Estes B-VOPs modificados utilizam a mesma sintaxe dos B-VOPs normais mas uma semântica diferente, que permite que exista predição entre as várias camadas. A Figura 15 mostra um exemplo do esquema de predição utilizado na codificação de um objecto com escalabilidade temporal. Na Figura 15, a camada de base tem metade da resolução temporal total e a camada de melhoramento contém a informação restante. A camada de base é codificada de modo independente, do mesmo modo que uma sequência sem escalabilidade, enquanto que a camada de melhoramento utiliza B-VOPs que têm como predição VOPs da camada de base previamente decodificados e que são temporalmente anteriores e posteriores ao B-VOP em questão. A Figura 16 mostra um exemplo do esquema de predição utilizado na escalabilidade espacial. Neste caso, a resolução espacial da camada de base é quatro vezes inferior (em número de *pixels*) à resolução da camada de melhoramento. A camada de base é codificada de modo independente, enquanto que a camada de melhoramento utiliza VOPs que usam como predição VOPs temporalmente coincidentes na camada de base e podem usar VOPs da camada de melhoramento previamente decodificados. Como a camada de base serve de referência para a camada de melhoramento, os VOPs da camada de base têm de ser codificados antes dos VOPs temporalmente correspondentes na camada de melhoramento.

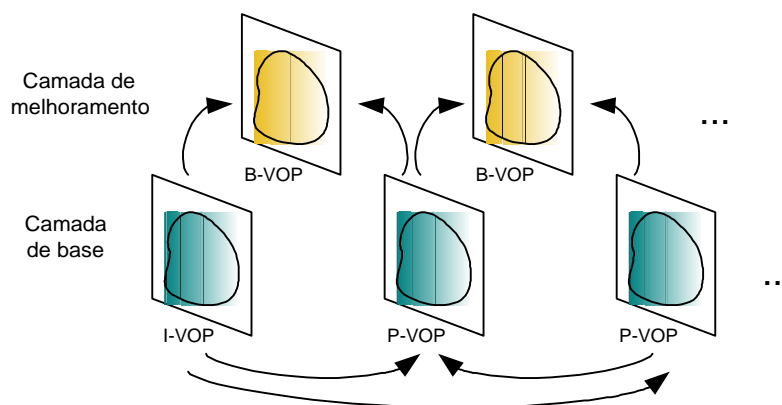


Figura 15 – Escalabilidade temporal de vídeo na norma MPEG-4.

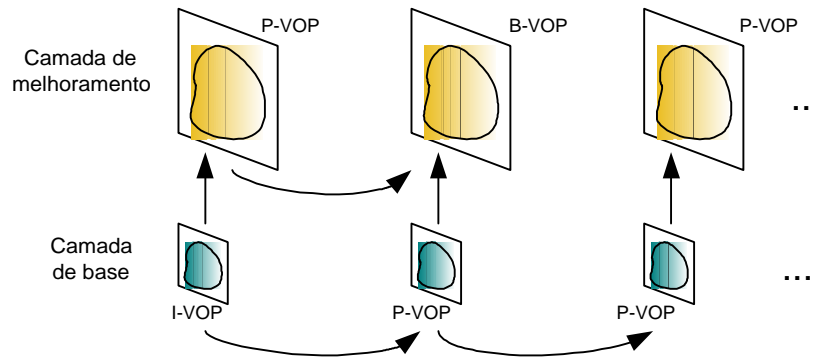


Figura 16 – Escalabilidade espacial de vídeo na norma MPEG-4.

2.7 Tipos de Objecto, perfis e níveis

A norma MPEG-4 especifica um elevado número de ferramentas associadas às inúmeras funcionalidades oferecidas, o que a torna bastante complexa e difícil de implementar na totalidade. Dado este facto, existe a necessidade de limitar a complexidade dos decodificadores que estão em conformidade com a norma, pois não é razoável esperar que um decodificador conforme implemente todas as ferramentas da norma com as respectivas capacidades máximas, por exemplo em termos de débito binário e de resolução, já que algumas ferramentas implicam uma grande complexidade de implementação e são desnecessárias para algumas classes de aplicações. Esta opção conduziria a decodificadores extremamente complexos e consequentemente a terminais com uma capacidade excessiva para classes de aplicações que requeiram menos ferramentas e uma variação de parâmetros mais limitada.

A necessidade de limitar a complexidade dos decodificadores levou à definição de subconjuntos de ferramentas e de limites para os parâmetros correspondentes (por exemplo, o débito binário) para que possam existir soluções de decodificação com um grau variável de complexidade que se adequem a cada classe de aplicações. Contudo, a definição de subconjuntos não é suficiente para garantir a interoperabilidade entre terminais, sendo necessário que estes subconjuntos sejam previamente conhecidos de todos os intervenientes no processo de comunicação. A interoperabilidade requer pois que estes subconjuntos estejam normalizados, para que os decodificadores possam ser implementados com a sua especificação e para que os codificadores possam produzir fluxos binários de acordo com a mesma especificação.

Para limitar a sua complexidade de implementação garantindo ainda assim interoperabilidade, a norma MPEG-4 utiliza um mecanismo baseado em *tipos de objecto*⁵, perfis e níveis. Segue-se a definição de cada um deles.

- **Tipo de objecto** – O *tipo de objecto* define a sintaxe do fluxo binário para um dado objecto que representa uma entidade com significado na cena audiovisual. Estabelece

⁵ Utiliza-se o termo *tipo de objecto* em itálico para indicar que se trata de um parâmetro específico associado ao objecto de vídeo.

uma lista de ferramentas de codificação que podem ser utilizadas para a codificação desse objecto. Definem-se *tipos de objecto* para objectos de áudio e visuais.

- **Perfil** – Um perfil define o conjunto de ferramentas que podem ser utilizadas num determinado terminal para reconstruir uma cena MPEG-4. Na versão 1 da norma MPEG-4, existem perfis de áudio, visuais, gráficos, de descrição de cena e de descritores de objecto. Os perfis de áudio e visuais são definidos com base nos *tipos de objecto*, especificando quais os *tipos de objecto* que podem ser usados para codificar os objectos de uma cena codificada de modo conforme com um dado perfil.
- **Nível** – Um nível especifica as restrições impostas aos perfis acima descritos, ou seja, às ferramentas por eles utilizadas, através de limitações impostas a alguns parâmetros relevantes.

O conceito de perfil já é conhecido da norma MPEG-2 Vídeo, mas enquanto que nesta norma os perfis indicam apenas quais as ferramentas de codificação a aplicar a sequências de vídeo rectangulares, na norma MPEG-4 tem que se considerar a existência de vários objectos na mesma cena. Os perfis destinam-se a limitar o conjunto de ferramentas que é necessário implementar no decodificador para garantir interoperabilidade entre terminais que só utilizem uma parte das ferramentas especificadas na norma. Por outras palavras, os perfis podem ser vistos como um compromisso entre a máxima interoperabilidade e a mínima complexidade para uma dada classe de aplicações [21]. Os níveis especificam limites para a complexidade computacional que é exigida, e definem um máximo para os codificadores e um mínimo para os decodificadores. Isto significa que os codificadores não devem produzir fluxos binários que requeiram uma complexidade de decodificação acima de um determinado limiar, pois nesse caso não existe a garantia que possam ser decodificados pelos decodificadores correspondentes. Por outro lado, os decodificadores têm de garantir pelo menos a complexidade computacional especificada pelo nível em questão para poder decodificar os correspondentes fluxos binários. Resumindo, uma dada combinação perfil e nível (denominada por perfil@nível) estabelece um limite superior da complexidade do fluxo binário criado no codificador e um limite inferior das capacidades do decodificador. É importante notar que a norma MPEG-4 é baseada em objectos, sendo as cenas compostas por vários objectos audiovisuais. Neste contexto, um determinado perfil@nível especifica a complexidade máxima para a totalidade dos objectos presentes na cena e não para cada objecto individualmente.

Tal como acontece com as anteriores normas MPEG-1 e MPEG-2, a norma MPEG-4 não especifica o processo de codificação, mas sim a decodificação e a sintaxe e semântica que o fluxo binário deve ter. Como o processo de codificação não está normalizado, existe a possibilidade de competição e inovação por parte da indústria, com resultados evidentes na obtenção de melhores e mais eficientes dispositivos e algoritmos. A única restrição que existe para a codificação é que deve ser gerado um fluxo binário em conformidade com a norma. Todavia, o codificador tem de ter em consideração a combinação perfil@nível para a qual o fluxo binário é criado, pois esta impõe restrições que devem ser observadas durante a codificação. O codificador deve permanentemente verificar se o fluxo binário está de acordo com os limites impostos pelo nível escolhido. Deste modo, o codificador tem de distribuir os recursos impostos pelo perfil@nível pelos vários objectos a codificar e pelos objectos já

codificados anteriormente mas que vão ser incluídos na cena e no respectivo conjunto de fluxos binários.

O grupo MPEG tem tentado limitar o número de perfis e níveis especificados no contexto da norma MPEG-4, já que a explosão deste número acaba por limitar a interoperabilidade entre terminais MPEG-4. Um grande número de perfis provoca confusão na indústria e utentes e cria dificuldades em termos de interoperabilidade. Novas ferramentas só são adicionadas à norma se trouxerem novas funcionalidades ou ganhos significativos no desempenho de funcionalidades já existentes. Isto implica que pequenas melhorias (por exemplo, na eficiência de codificação) não são suficientes para acrescentar uma nova ferramenta, o que tornaria a norma ainda mais complexa e difícil de implementar.

As ferramentas incluídas na norma devem ser rigorosamente testadas e devem prever futuras utilizações. Pelo contrário, definir um novo perfil ou nível não afecta a sintaxe do fluxo binário, o que implica que estes podem ser definidos mais tarde e de acordo com as necessidades. Contudo, um novo perfil só deve ser definido se não existir na norma nenhum outro, possivelmente mais poderoso, que possa ser utilizado. Para a definição de níveis para um dado perfil, aplicam-se as mesmas premissas: só são criados novos níveis quando os existentes não sejam claramente suficientes e quando haja garantias fortes de que vão ser efectivamente utilizados, demonstradas através das intenções expressas por empresas interessadas em desenvolver produtos.

2.7.1 *Tipos de objecto visuais*

Na norma MPEG-4, as cenas podem conter vários objectos audiovisuais de tipos diferentes, por exemplo, objectos de vídeo rectangulares ou de forma arbitrária, texto, gráficos, etc. Deste modo, é necessário que exista um nível intermédio entre as ferramentas e o perfil da cena, que é o *tipo de objecto* associado ao conjunto de ferramentas usadas para cada objecto (o perfil caracteriza a cena no seu conjunto e não os objectos). Cada *tipo de objecto* define o subconjunto de ferramentas da norma MPEG-4 Visual que pode ser usado para codificar um dado objecto na cena. A Tabela 1 mostra as ferramentas correspondentes aos vários *tipos de objecto* visuais especificados na versão 1 da norma MPEG-4 Visual [3].

Ferramentas	Tipos de Objecto								
	Simple	Simple Scalable	Core	N-Bit	Main	Still Scalable Texture	Animated 2D Mesh	Basic Animated Texture	Simple Face
<ul style="list-style-type: none"> I-VOP, P-VOP Predição AC/DC 4-MV, Unrestricted MV 	✓	✓	✓	✓	✓		✓		
Resiliência a Erros <ul style="list-style-type: none"> Ressincronização Partição de dados Códigos reversíveis Extensão do cabeçalho 	✓	✓	✓	✓	✓		✓		
Short Header	✓		✓	✓	✓		✓		
B-VOP		✓	✓	✓	✓		✓		
Quantificação Método 1/Método 2			✓	✓	✓		✓		
Escalabilidade temporal baseada em P-VOPs <ul style="list-style-type: none"> Rectangulares Forma arbitrária 			✓	✓	✓		✓		
Forma Binária			✓	✓	✓		✓	✓	
Forma Multi-nível					✓				
Conteúdo entrelaçado					✓				
Sprite					✓				
Escalabilidade Temporal (Rectangular)		✓							
Escalabilidade Espacial (Rectangular)		✓							
N-Bit (4-12 bit)				✓					
Scalable Still Texture						✓	✓	✓	
Malha dinâmica 2D com topologia uniforme							✓	✓	
Malha dinâmica 2D com topologia de Delaunay							✓		
Parâmetros de Animação Facial									✓

Tabela 1 – Ferramentas utilizadas para os vários tipos de objectos visuais na versão 1 da norma MPEG-4.

Existem cinco *tipos de objecto* de vídeo (objectos visuais naturais) especificados na versão 1 da norma MPEG-4 Visual [3]:

- **Simple** – O tipo *Simple* corresponde a objectos de vídeo rectangulares, com uma largura e altura arbitrárias, e suporta as ferramentas de resiliência a erros. Utiliza ferramentas de codificação simples, baseadas em VOPs I (*Intra*) e P (*Predicted*).
- **Simple Scalable** – O tipo de *objecto Simple Scalable* é a versão escalável do *Simple*. Suporta escalabilidade temporal e espacial, sendo a camada de base do tipo *Simple*. As camadas de melhoramento (como a de base) têm de ser constituídas por VOPs rectangulares.

- **Core** – O *tipo de objecto Core* utiliza um conjunto de ferramentas que estende o conjunto usado no tipo *Simple* visando uma maior eficiência de codificação. Fornece uma melhor qualidade através da utilização de B-VOPs, ou seja VOPs codificados com interpolação bidireccional. Além disso, suporta objectos de forma arbitrária pois permite utilizar máscaras binárias, que podem ter uma transparência constante.
- **N-Bit** – O *tipo de objecto N-Bit* é semelhante ao *Core*, mas permite que o número de bits por *pixel* e por componente varie entre 4 e 12 bits, tanto na luminância como na crominância; destina-se a aplicações onde as imagens possam ter uma origem diferente da usual, por exemplo imagens térmicas.
- **Main** – O *tipo de objecto Main* é aquele que fornece o conjunto de ferramentas com maior eficiência de compressão. Acrescenta ao *Core* o suporte de formas multi-nível, ou seja, objectos com transparência variável, *sprites* e vídeo entrelaçado.

O *tipo de objecto Still Scalable Texture* destina-se a representar imagens naturais fixas. Este *tipo de objecto* suporta objectos com forma arbitrária e utiliza codificação por *wavelets*, o que permite escalabilidade espacial com elevada granularidade e logo a reconstrução incremental da imagem no receptor.

Existem ainda três outros *tipos de objectos* correspondentes a objectos visuais sintéticos, por vezes incluindo também vídeo ou imagens de origem natural.

- **Basic Animated Texture** – Permite a animação de malhas 2D através da movimentação dos seus vértices mapeadas com imagens estáticas de forma arbitrária. Estas imagens são as especificadas pelo *tipo de objecto Still Scalable Texture*.
- **Animated 2D Mesh** – Este *tipo de objecto* utiliza uma malha 2D sintética em conjunto com vídeo natural de forma arbitrária. A codificação do vídeo natural utiliza as mesmas ferramentas do tipo *Core*; o vídeo é mapeado na malha e deformado através da movimentação dos vértices da malha 2D.
- **Simple Face** – O *tipo de objecto Simple Face* possui ferramentas para codificar informação de animação facial. Esta destina-se a animar modelos 3D de faces humanas, não normativos, ou seja que podem ser livremente escolhidos pelo receptor.

2.7.2 Perfis visuais

Os perfis visuais determinam quais os *tipos de objecto* visuais que podem estar presentes numa dada cena, ou seja quais os *tipos de objecto* que podem ser usados pelos objectos na cena. A Tabela 2 mostra quais os *tipos de objecto* suportados por cada perfil e ainda o número de níveis que foram definidos para cada perfil na versão 1 da norma MPEG-4 Visual [3]. Já depois da versão 1 da norma MPEG-4 Visual ter sido terminada, foram acrescentados à norma vários outros perfis, alguns dos quais usam mesmo novos *tipos de objecto*, tais como os tipos de objecto para estúdio e *Fine Granularity Scalability* (FGS) destinado ao *streaming* de vídeo na Internet.

Tipo de Objecto	Perfis								
	<i>Simple</i>	<i>Simple Scalable</i>	<i>Core</i>	<i>N-Bit</i>	<i>Main</i>	<i>Scalable Texture</i>	<i>Simple Face</i>	<i>Basic Animated Texture</i>	<i>Hybrid</i>
<i>Simple</i>	✓	✓	✓	✓	✓				✓
<i>Simple Scalable</i>		✓							
<i>Core</i>			✓	✓	✓				✓
<i>Main</i>					✓				
<i>N-Bit</i>				✓					
<i>Animated 2D Mesh</i>									✓
<i>Basic Animated Texture</i>								✓	✓
<i>Scalable Texture</i>					✓	✓		✓	✓
<i>Simple Face</i>							✓	✓	✓
Número de Níveis	3	2	2	1	3	3	2	2	2

Tabela 2 – Tipos de objecto suportados pelos perfis visuais definidos na versão 1 da norma MPEG-4.

De seguida faz-se uma breve descrição de cada perfil visual e indicam-se aplicações possíveis, apenas a título exemplificativo, já que as aplicações a que se destina cada perfil não estão normalizadas e ficam sempre ao critério e imaginação dos fabricantes.

- **Simple** – O perfil *Simple* só suporta objectos do tipo *Simple*. Foi desenvolvido para aplicações de baixa complexidade que tenham poucos recursos, como por exemplo serviços móveis, transmissão de vídeo na Internet, câmaras de vídeo portáteis que gravem vídeo digital para memória ou disco, etc. Existem três níveis com débitos binários que vão desde 64 kbit/s, no nível 1, até 384 kbit/s, no nível 3. Este perfil suporta um máximo de quatro objectos na cena, com a dimensão típica QCIF (176×144 para a luminância) no primeiro nível ou CIF (352×288 para a luminância) nos restantes. Estas dimensões não são impostas pela norma, mas tratam-se apenas de valores indicativos resultantes do número máximo de macroblocos de memória imposto pelo nível usado. O nível também especifica a dimensão das várias memórias necessárias para o processo de descodificação (ver a secção 2.8).⁶
- **Simple Scalable** – O perfil *Simple Scalable* inclui objectos *Simple* e objectos *Simple Scalable* que acrescentam a escalabilidade na lista de ferramentas do objecto *Simple*. Possui dois níveis com um débito binário máximo de 128 kbit/s e 256 kbit/s, respectivamente.
- **Core** – O perfil *Core* aceita objectos do tipo *Core* e *Simple*. Este perfil é útil para aplicações interactivas, pois combina uma boa eficiência de codificação com uma complexidade média e suporta objectos rectangulares ou com forma arbitrária. Existem dois níveis com débitos binários de 384 kbit/s, no nível 1, para cenas com a

⁶ Já depois da versão 1 da norma MPEG-4 Visual ter sido especificada foi acrescentado um novo nível a este perfil, designado por nível zero, onde apenas existe um objecto rectangular com um máximo de 64 kbit/s, para ser adoptado como norma pelo consórcio 3GPP (UMTS).

dimensão típica QCIF e um máximo de 4 objectos, e 2 Mbits/s, no nível 2, para cenas com a dimensão típica CIF e com um máximo de 16 objectos.

- **N-Bit** – Este perfil é semelhante ao *Core* mas acrescenta o *tipo de objecto N-Bit*, oferecendo assim a possibilidade de ter objectos na cena para os quais o número de bits por *pixel* é variável. Pode ser utilizado em imagens térmicas (em aplicações de vigilância ou mapas) ou médicas, onde um maior número de bits por *pixel* pode ser útil, no sentido de que aumenta a gama dinâmica da luminância e das crominâncias.
- **Main** – O perfil *Main* pode incluir objectos do tipo *Simple*, *Core*, *Main* e *Still Scalable Texture*. Destina-se principalmente a aplicações de difusão audiovisual, pois inclui todas as ferramentas que permitem oferecer máxima eficiência de codificação. Para além de proporcionar as melhores ferramentas de compressão de entre todos os perfis, permite o uso de objectos de forma arbitrária e que podem ter transparência não uniforme. Para este perfil existem três níveis com débitos binários elevados que vão dos 2 Mbit/s até aos 38 Mbit/s e um máximo de 32 objectos.
- **Scalable Texture** – Este perfil só aceita objectos do tipo *Still Scalable Texture*. É um perfil destinado principalmente a aplicações que conjuguem imagens ou gráficos baseados em BIFS e som de um modo sincronizado, por exemplo para terminais móveis.

A Figura 17 resume as relações existentes entre os *tipos de objecto* e os perfis naturais definidos na versão 1 da norma MPEG-4. O perfil *Simple* só pode incluir objectos do tipo *Simple*. O perfil *Simple Scalable* suporta objectos do tipo *Simple* e *Simple Scalable*. O perfil *Core* aceita objectos do tipo *Core* e *Simple*. O perfil *N-bit* pode incluir objectos do tipo *N-bit* e *Core*. O perfil *Main* é o mais abrangente, incluindo objectos do tipo *Main*, *Still Scalable Texture*, *Core* e *Simple*. Por último, o perfil *Scalable Texture* só aceita objectos do tipo *Still Scalable Texture*.

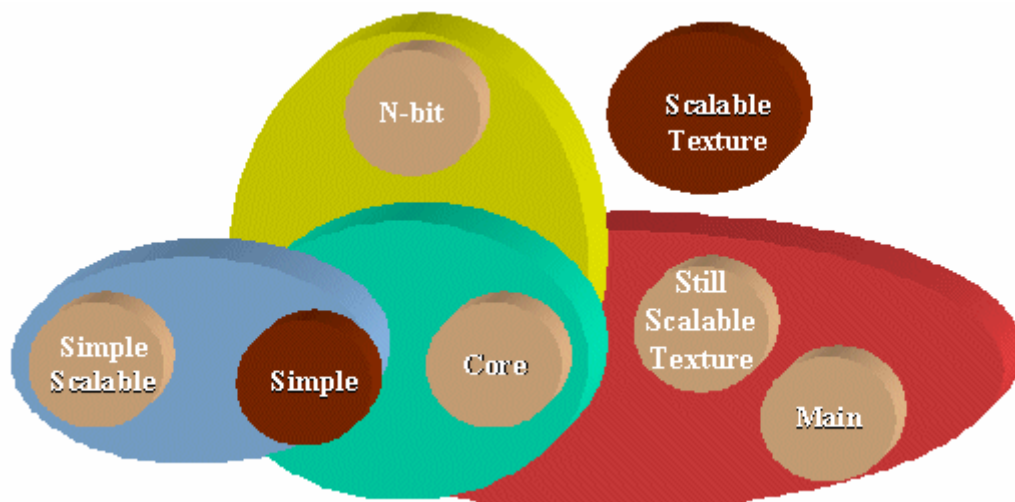


Figura 17 – Relações existentes entre os tipos de objecto e os perfis de vídeo definidos na versão 1 da norma MPEG-4.

Para além dos perfis de vídeo, existem outros perfis visuais, nomeadamente:

- **Simple Face** – Este perfil utiliza, exclusivamente, objectos do tipo *Simple Face*. Permite um máximo de quatro faces numa cena e pode ser utilizado, por exemplo, para reuniões virtuais ou videotelefonia. Os débitos binários são necessariamente baixos (16 kbit/s, no nível 1, e 32 kbit/s no nível 2), já que só são transmitidos os parâmetros que vão animar o modelo 3D disponível no receptor.
- **Basic Animated Texture** – Este perfil possibilita a animação de imagens estáticas sobre malhas 2D e de faces. Pode incluir objectos do tipo *Basic Animated Texture*, *Still Scalable Texture* e *Simple Face* e permite a criação de conteúdo animado a baixos débitos, que não ultrapassam os 128 kbit/s.
- **Hybrid** – O perfil *Hybrid* permite a integração de objectos de natureza sintética e natural numa mesma cena. Pode ser comparado ao perfil *Core* para os objectos visuais naturais e para os objectos sintéticos suporta a animação de malhas 2D, texturas escaláveis e faces animadas. Com estas ferramentas, este perfil pode ser utilizado para colocar objectos virtuais em mundos reais ou objectos reais em cenários virtuais. É, sem dúvida, o perfil mais complexo em termos de ferramentas de codificação.

Como já foi referido, as combinações perfil@nível impõem restrições na composição das cenas audiovisuais. A Tabela 3 indica quais os *tipos de objecto* visuais e o respectivo número máximo de objectos que podem existir para cada perfil e nível.

Perfil@Nível	Tipo de Objecto				
	Simple	Simple Scalable	Core	N-Bit	Main
Simple@L1	4	-	-	-	-
Simple@L2	4	-	-	-	-
Simple@L3	4	-	-	-	-
Simple Scalable@L1	4	4	-	-	-
Simple Scalable@L2	4	4	-	-	-
Core@L1	4	-	4	-	-
Core@L2	16	-	16	-	-
Main@L2	16	-	16	-	16
Main@L3	32	-	32	-	32
Main@L4	32	-	32	-	32
N-Bit@L2	16	-	16	16	-

Tabela 3 – Tipos de objecto visuais e respectivo número máximo de objectos para cada perfil@nível definido na versão 1 da norma MPEG-4.

2.8 Mecanismo de Verificação de Vídeo

Para poder garantir a conformidade dos fluxos binários de vídeo com uma dada combinação perfil@nível, a norma MPEG-4 define um mecanismo de verificação de vídeo (*Video Buffering Verifier*) que consiste em três modelos: *Video rate Buffer Verifier* (VBV), *Video Complexity Verifier* (VCV) e *Video reference Memory Verifier* (VMV).

A ideia da utilização de um mecanismo deste tipo para limitar a complexidade de descodificação de um fluxo binário⁷ foi já adoptada nas normas de codificação de vídeo MPEG-1 e MPEG-2. Nestas normas, o principal objectivo deste mecanismo era a introdução de algumas restrições à variação do número de bits por trama, já que a dimensão das imagens é fixa ao longo da sequência, e consequentemente à complexidade do vídeo codificado.

A complexidade do vídeo codificado está relacionada com o débito binário, com o ritmo a que são processados dados de imagem pelo codificador, ou seja, o número de macroblocos por segundo (MB/s), e também com a resolução espacial. Nas normas MPEG-1 e MPEG-2, o ritmo de dados de imagem é constante, pois as imagens têm dimensões fixas (a trama) e são codificadas a um ritmo de trama constante. Isto não acontece na norma MPEG-4, pois os vários objectos de vídeo que compõem uma cena podem variar em tamanho ao longo do tempo e podem ser codificados a frequências de trama diferentes. Assim, o número e o tipo de macroblocos (transparentes, opacos ou de fronteira) que o descodificador tem de processar por segundo pode variar muito ao longo do tempo. Deste modo, para limitar a complexidade de descodificação de um fluxo binário é necessário limitar o número médio de MB/s e a sua variabilidade, e ainda a sua área total (por determinar a memória necessária no descodificador). Esta é a maior novidade no mecanismo de verificação da norma MPEG-4 relativamente às normas anteriores, já que não limita só a dimensão da memória do fluxo binário mas limita também a memória necessária para a descodificação dos macroblocos da imagem e a capacidade computacional para a descodificação dos macroblocos.

Descrevem-se de seguida os três modelos que compõem o mecanismo de verificação de vídeo da norma MPEG-4 Visual [3]. Cada um deles impõe limites que devem ser respeitados no codificador para que o fluxo binário gerado possa ser descodificado por um dado descodificador conforme com o perfil e nível escolhidos.

- **Video rate Buffer Verifier (VBV)** – O modelo VBV é utilizado para verificar se a memória de fluxo binário necessária para o descodificador de vídeo não excede os valores especificados pelo perfil e pelo nível (ou seja para verificar se o codificador não está a gastar bits a mais). É definido em termos da capacidade da memória VBV, ou seja, do número máximo de bits que o descodificador pode guardar na memória de fluxo binário (à entrada). O modelo VBV aplica-se de modo independente a cada um dos objectos presentes na cena e, se existir escalabilidade, o modelo é aplicado a cada camada de codificação. Um fluxo binário em conformidade com o VBV é gerado se a memória virtual nunca encher ou esvaziar completamente.
- **Video Complexity Verifier (VCV)** – O modelo VCV é utilizado para verificar se a capacidade computacional, medida em macroblocos por segundo, necessária no descodificador não excede os valores especificados pela combinação perfil@nível escolhida. Neste modelo, uma memória virtual acumula os macroblocos codificados pelo codificador. A memória VCV tem uma capacidade pré-definida pelo perfil e pelo nível que, juntamente com o ritmo mínimo de descodificação dos macroblocos, permite ao codificador calcular quantos macroblocos pode codificar ao longo do tempo, e logo também em cada instante, para produzir um fluxo binário em conformidade com o VCV. Se existirem objectos de vídeo de forma arbitrária na

⁷ Note-se que se esta complexidade não for limitada, não há qualquer garantia que o conteúdo possa ser descodificado, por um dado descodificador, dentro dos tempos necessários para a sua normal visualização.

cena, são utilizados duas memórias VCV: uma para o conjunto de todos os macroblocos e outra apenas para os macroblocos de fronteira. Estas memórias são esvaziadas a ritmos diferentes, tentando expressar o facto da complexidade de descodificação dos macroblocos de fronteira relativamente a opacos e transparentes ser bastante diferente. A diferença de complexidade entre os vários tipos de macroblocos no contexto da codificação de vídeo segundo a norma MPEG-4 é, aliás, o tema central desta tese, conforme se verá mais adiante.

- ***Video reference Memory Verifier (VMV)*** – O modelo VMV é utilizado para verificar se a memória para os macroblocos (descodificados) requerida no descodificador para a descodificação de uma dada cena não excede os valores especificados pelo perfil e pelo nível escolhidos. Este modelo é definido em termos da capacidade da memória VMV, ou seja, do número máximo de macroblocos descodificados que o descodificador pode armazenar durante o processo de descodificação. A memória é preenchida a um ritmo constante definido pelo VCV enquanto o processo de descodificação decorre. O modelo VMV é aplicado ao conjunto de todos os macroblocos de todos os fluxos binários correspondentes a objectos de vídeo de origem natural presentes na cena.

2.8.1 Interacção entre VBV, VCV e VMV

O modelo VBV define o instante em que parte do fluxo binário está disponível para ser retirado da memória VBV e descodificado. O modelo VCV define a velocidade a que os macroblocos são descodificados. O modelo VMV define a quantidade de memória de descodificação que é utilizada e libertada ao longo do tempo. Para evitar problemas, deve existir um compromisso relativamente ao instante em que o descodificador de vídeo começa a descodificar um fluxo binário. Em princípio, parece vantajoso que o processo de descodificação seja iniciado o mais cedo possível, mas isso está dependente do VBV e do VMV. O modelo VBV indica que o descodificador só pode começar a descodificar quando existam bits suficientes na memória. Quando o processo de descodificação está em curso, são gerados macroblocos que ocupam a memória de descodificação, que ficará completamente cheia se a descodificação for iniciada demasiado cedo. Por outro lado, se o descodificador começar a descodificar muito tarde, então não será capaz de completar a descodificação a tempo e os bits serão “apagados” do VBV antes de terem sido processados. Do mesmo modo, a informação em memória necessária para a predição do VOP que está a ser descodificado será removida do VMV (o instante em que os bits são removidos do VMV é imposto pelo VBV).

Assim, para um codificador gerar um fluxo binário em conformidade com a norma MPEG-4 Visual para um dado perfil e nível, tem de controlar o fluxo binário gerado através da simulação do enchimento, no descodificador, do VBV, VCV e VMV em conjunto (Figura 18).

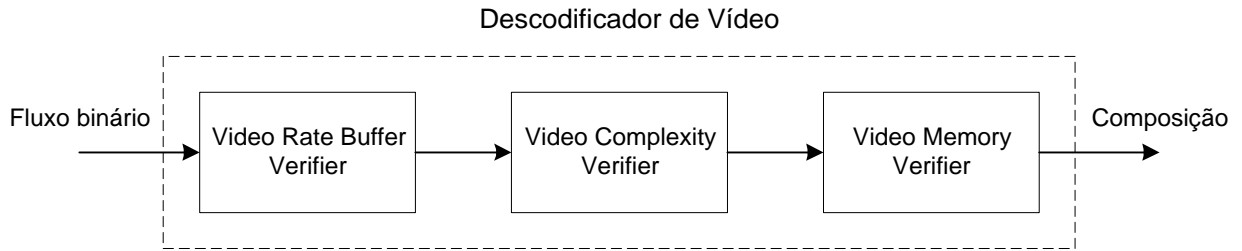


Figura 18 – Mecanismo de verificação de vídeo.

Descrevem-se, de seguida, as acções que o codificador deve tomar para controlar a ocupação das memórias VBV, VCV e VMV.

- **Memória VBV**

A Figura 19 mostra um exemplo simplificado da dinâmica de ocupação do VBV, correspondente à simulação do enchimento do VBV do decodificador feita pelo codificador. Segundo o modelo VBV, os bits de cada VOP codificado são retirados instantaneamente da memória VBV nos instantes de descodificação (t_i), ao contrário de uma memória real onde esta operação demora algum tempo. Deste modo, o modelo acomoda o pior caso, ou seja aquele onde o decodificador guarda na memória todos os bits do VOP corrente até proceder à sua descodificação. Como se pode observar na Figura 19, entre os instantes t_0 e t_4 são gastos cada vez menos bits por VOP ($d_0 > d_1 > d_2 > d_3$), o que faz com que a memória esvazie pouco em cada instante de descodificação. Isto tem como consequência que o VBV comece a encher demasiado, e se não forem tomadas medidas adequadas pode eventualmente exceder a sua capacidade. Assim, para evitar que a memória VBV encha por completo, o codificador deve gerar mais bits por cada VOP (reduzindo o passo de codificação) ou gerar bits de enchimento para obrigar o VBV a esvaziar. Isso acontece a partir do instante t_4 , quando o codificador começa a gerar mais bits por cada VOP. Se o codificador continuar a gerar muitos bits por cada VOP, a ocupação da memória VBV começa a diminuir; mantendo-se esta situação, o VBV pode eventualmente esvaziar por completo, como acontece no instante t_7 . Para contrariar esta situação, o codificador deve gerar menos bits por cada VOP, para que a memória comece a encher. Esta medida faz com que a ocupação da memória aumente, como se observa na Figura 19 a partir do instante t_7 . Pode-se concluir através deste exemplo que deve existir um compromisso entre os bits gerados e o enchimento do VBV, de modo a que este nunca encha ou esvazie completamente.

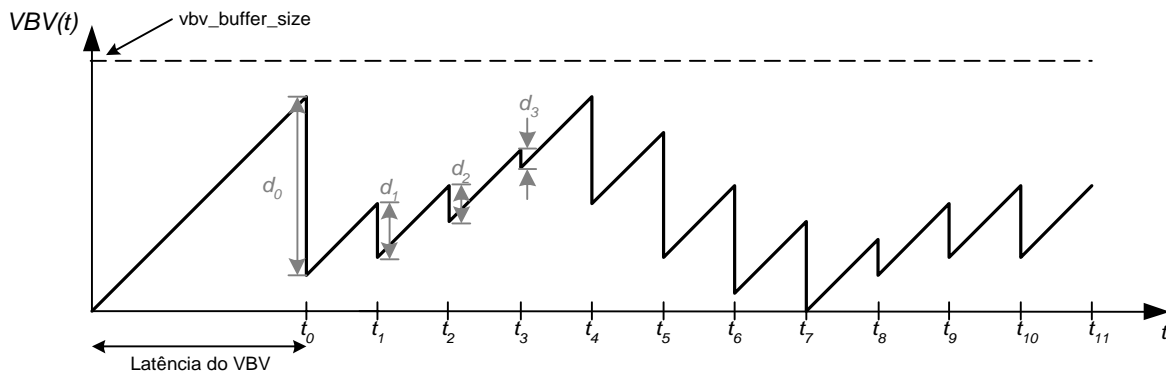


Figura 19 – Dinâmica da ocupação do VBV.

• Memória VCV

Se a ocupação da memória VCV se aproximar do seu limite máximo, o codificador deve dar mais tempo ao decodificador para decodificar os VOPs porque eles são complexos, muitos em quantidade ou são compostos por muitos macroblocos. Isto pode ser feito, por exemplo, através do aumento do intervalo entre VOPs codificados, recorrendo ao salto de tramas. Se forem os macroblocos com informação de forma o factor que determina a ocupação do VCV, o codificador/autor pode tentar mudar a forma dos objectos ou juntar objectos para assim reduzir o número de macroblocos de fronteira (se isso for aceitável para a aplicação em causa). Também se pode tentar reduzir a dimensão de alguns VOPs, dividindo um objecto em dois ou mais objectos para assim reduzir o número total de macroblocos dentro das *bounding boxes* resultantes. A Figura 20 mostra a dinâmica da ocupação do VCV. No instante de decodificação de cada VOP, as memórias VCV e B-VCV são cheias instantaneamente com os macroblocos do VOP, sendo depois esvaziadas a um ritmo constante até ao próximo instante de decodificação de um VOP. O capítulo 6 explica em pormenor o funcionamento e a dinâmica do modelo VCV.

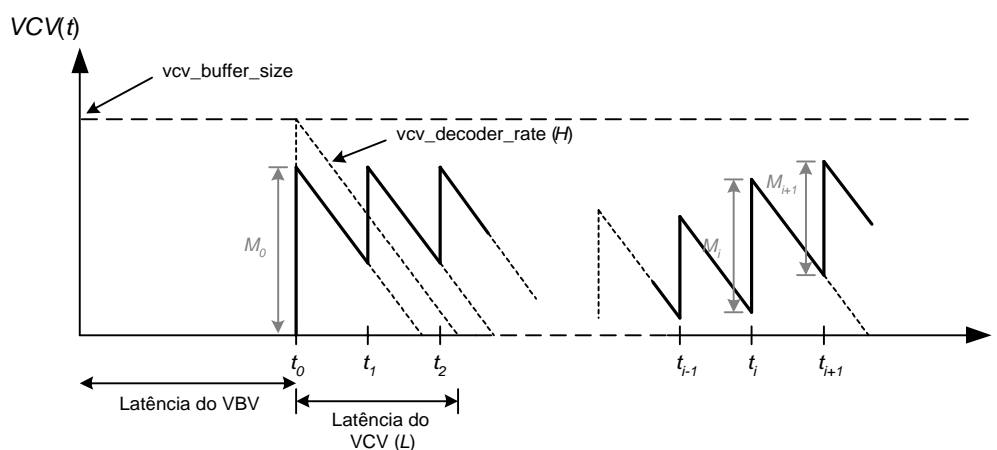


Figura 20 – Dinâmica de ocupação do VCV.

• Memória VMV

A Figura 21 mostra a dinâmica da ocupação do VMV. A memória VMV está inicialmente vazia e é preenchida com os macroblocos que são decodificados (e que vão servir de predição para alguns dos seguintes). A memória necessária para decodificar o VOP i é definida como o número de macroblocos do VOP, M_i , sendo consumida a um ritmo dado pelo modelo VCV. O esvaziamento do VMV depende do tipo de VOP que está a ser decodificado (I, P ou B). Se o VOP for I ou P, no instante de composição mais a latência do VCV é libertada instantaneamente a memória reservada para o VOP I ou P anterior na ordem de decodificação. Se o VOP for B, no instante de composição mais a latência do VCV é libertada instantaneamente a memória reservada para o VOP B corrente. A latência do VCV é definida como o tempo necessário para decodificar o conteúdo da memória VCV cheia e define o atraso mínimo que deve ser respeitado no início do processo de decodificação.

Para evitar que a memória VMV fique demasiado cheia, a utilização da memória do decodificador deve ser diminuída. Isto pode ser conseguido através da utilização de VOPs mais pequenos (que tenham menos macroblocos) ou evitando a inclusão de B-VOPs que

obrigam a uma maior utilização da memória, já que obrigam a guardar o VOP anterior e posterior para predição.

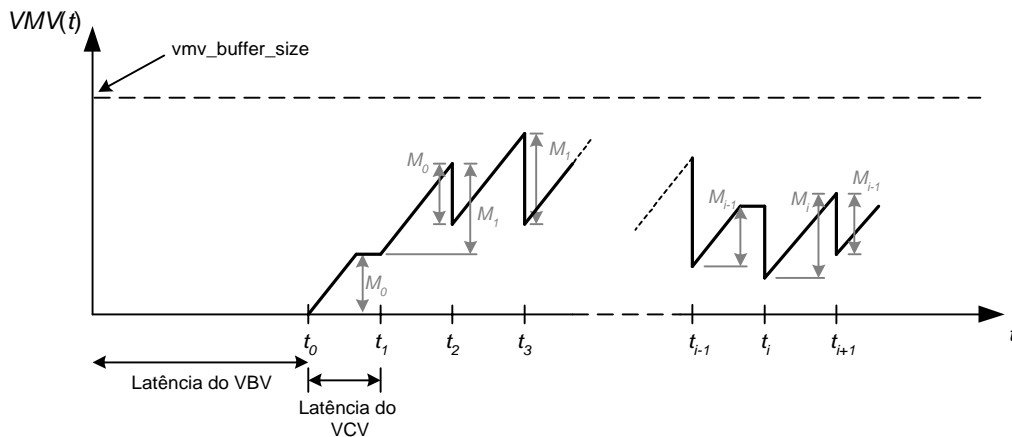


Figura 21 – Dinâmica da ocupação do VMV.

2.9 Conclusões

A norma MPEG-4 sucede às largamente utilizadas normas MPEG-1 e MPEG-2, que deram origem a produtos e sistemas como o CD interativo (CDi), difusão de áudio digital (DAB), *Digital Video Disc* (DVD) e televisão digital (DVB e ATSC). Mas o modelo de dados utilizado, baseado nas tramas, torna estas normas limitadas em termos de funcionalidades oferecidas. A norma MPEG-4 altera esta situação e oferece ao utilizador novos modos de criar, reutilizar, aceder e consumir o conteúdo audiovisual. A sua representação baseada em objectos, onde uma cena é composta por objectos naturais e sintéticos com um determinado comportamento no espaço e no tempo e com os quais é possível interagir, abre um novo mundo de possibilidades para as aplicações multimédia.

A norma MPEG-4 foi finalizada há já algum tempo (oficialmente, no início de 1999) e o seu sucesso irá depender de muitos factores, tais como as necessidades do mercado, a disponibilidade de outras soluções concorrentes, nomeadamente proprietárias, a implementação eficiente em *software* e *hardware* e os compromissos possíveis entre funcionalidades e complexidade. Técnica e funcionalmente, a norma MPEG-4 tem um enorme potencial, tal como foi descrito neste capítulo. Com o rápido desenvolvimento das tecnologias multimédia interactivas, certamente que a norma MPEG-4 se irá tornar na referência para a representação eficiente de conteúdos audiovisuais.

Capítulo 3

Vídeo MPEG-4: uma Aplicação em Conformidade

3.1 Introdução

Este capítulo descreve a aplicação de edição, codificação e decodificação de vídeo MPEG-4 desenvolvida no contexto da presente tese. A aplicação permite a criação de cenas compostas por vários objectos de vídeo através de uma interface potente, flexível e amigável. Uma cena composta por vários objectos de vídeo pode ser então codificada em conformidade com a norma MPEG-4 Visual segundo um determinado perfil e nível [3]. A decodificação dos fluxos binários resulta em objectos de vídeo que podem ser acedidos de um modo independente na cena decodificada. Com esta aplicação, é possível criar novas cenas para além daquelas que se encontram disponíveis, para assim exercitar e testar as soluções técnicas a desenvolver no âmbito da tese.

Em resumo, as principais motivações que levaram ao desenvolvimento da aplicação aqui descrita foram:

- Possibilidade de gerar fluxos binários conformes com a norma MPEG-4 Visual (perfil@nível) através de uma interface potente, flexível e amigável;
- Capacidade de gerar os fluxos binários acima referidos para cenas a criar segundo as necessidades de teste (e não só usando aquelas disponíveis), podendo deste modo exercitar e testar as soluções a desenvolver nesta tese com cenas adequadas;
- Obter uma aplicação que permita demonstrar e explicar a norma MPEG-4 de forma simples e clara, nomeadamente as suas novas funcionalidades, mesmo para não especialistas.

A aplicação foi desenvolvida com a linguagem de programação C++ utilizando o compilador Visual C++ 6.0 da Microsoft, o que possibilita que esta tenha uma interface amigável e intuitiva e possa correr no sistema operativo Windows, largamente difundido em todo o mundo. As potencialidades da aplicação e a sua fácil utilização permitem demonstrar e explicar a norma MPEG-4 de uma forma simples e clara a qualquer pessoa, mesmo sem formação na área da codificação de vídeo. Aliás, a aplicação é já hoje utilizada por muitas universidades e empresas, em vários países, para ilustrar as capacidades da norma MPEG-4. Para que a aplicação desenvolvida pudesse ser utilizada pelo maior número de pessoas possível, em qualquer país, a interface foi desenvolvida em Inglês, dada a quase universalidade desta Língua.

Este capítulo descreve a estrutura e o funcionamento da aplicação, que pode ser dividida, funcionalmente, em três partes principais: editor de cena, codificador MPEG-4 e descodificador MPEG-4.

3.2 Editor de cena

O editor de cena permite ao utilizador criar uma nova cena ou modificar uma cena já existente. O editor de cena apresenta três áreas distintas, como se pode ver na Figura 22, destinadas às seguintes tarefas ou visualização de informação:

1. **Árvore de objectos** – No lado esquerdo da área de trabalho é visualizada uma árvore que contém os objectos de vídeo já seleccionados, representados a vermelho se o objecto está já incluído na cena ou a cinzento caso o objecto ainda não esteja incluído na cena em construção. Todas as opções relacionadas com os objectos de vídeo podem ser acedidas através da árvore seleccionando o objecto com o botão direito do rato.
2. **Área de composição ou visualização de cena** – No centro da janela principal da aplicação encontra-se a cena que está a ser composta, juntamente com os botões que controlam a sua visualização. Dentro da cena, os objectos de vídeo podem ser arrastados para alterar a sua posição espacial e as suas propriedades de composição e codificação podem ser alteradas com o botão direito do rato.
3. **Barra de tempo ou visualização de informação auxiliar** – A parte de baixo do ecrã é usada, alternadamente, como barra de tempo ou para visualização de informação auxiliar:
 - **Barra de tempo** – Esta barra permite ao utilizador especificar a posição temporal de todos os objectos de vídeo, arrastando as barras azuis para as posições temporais desejadas para cada objecto. Estas barras de tempo mostram também, em cada instante, a posição temporal da cena através de uma barra vertical vermelha.
 - **Visualização de informação auxiliar** – Esta janela mostra informações úteis no decorrer do processo de codificação e surge no mesmo local que a barra de tempo. O utilizador pode escolher qual a janela que pretende ver em cada instante – barra de tempo ou informação auxiliar – através de um botão existente ao fundo do ecrã.

Tanto a árvore de objectos como a barra de tempo podem ser retiradas do ecrã para maximizar a área de composição da cena.

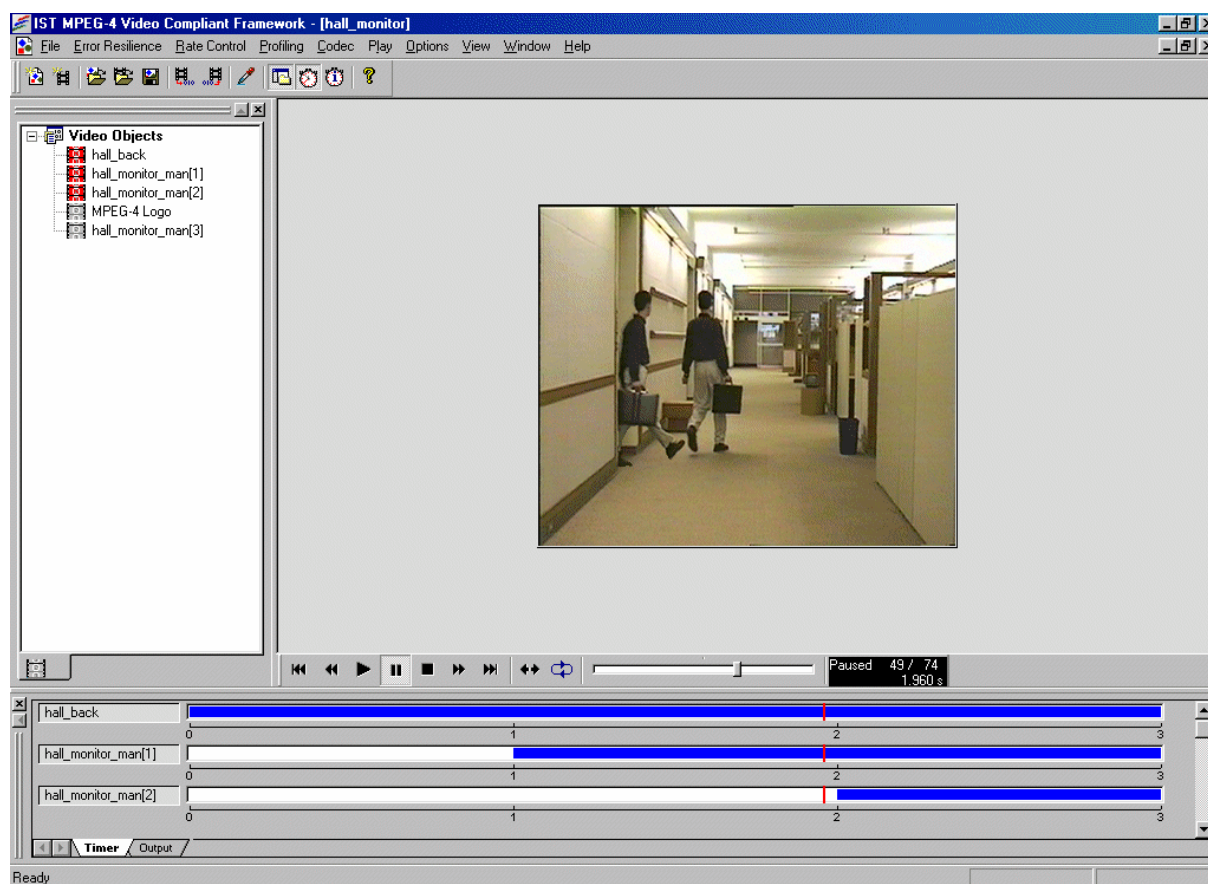



Figura 22 – Aspecto geral da aplicação.

3.2.1 Criação de uma nova cena

Uma nova cena, aqui entendida como uma composição de vários objectos de vídeo, é criada escolhendo a opção *New Scene* no menu *File* ou seleccionando o botão . Antes de se proceder à criação da cena, alguns parâmetros básicos têm de ser definidos (Figura 23):

- **Resolução espacial da cena** – Dimensão vertical e horizontal da cena, em termos de *pixels* de luminância. Podem ser escolhidos alguns formatos padrão tais como CIF, QCIF ou SIF⁸ ou o utilizador pode especificar a dimensão vertical e horizontal que desejar.
- **Duração da cena** – Duração da cena em segundos; a frequência de cena (frequência correspondente às tramas⁹ compostas) será igual à maior frequência de trama/VOP de entre os objectos que compõem a cena.
- **Nome da cena** – Nome que identifica a cena.

⁸ Formatos padrão para a luminância: CIF = 352×288; QCIF = 176×144; SIF = 352×240 *pixels*; tipicamente, as crominâncias têm metade da resolução em cada direcção ou seja uma sub-amostragem do tipo 4:2:0.

⁹ Na interface da aplicação, utiliza-se muitas vezes o termo *frame* para designar tanto objectos rectangulares como objectos com forma arbitrária. No contexto da norma MPEG-4, o termo correcto para objectos de vídeo com forma arbitrária é VOP (*Video Object Plane*).

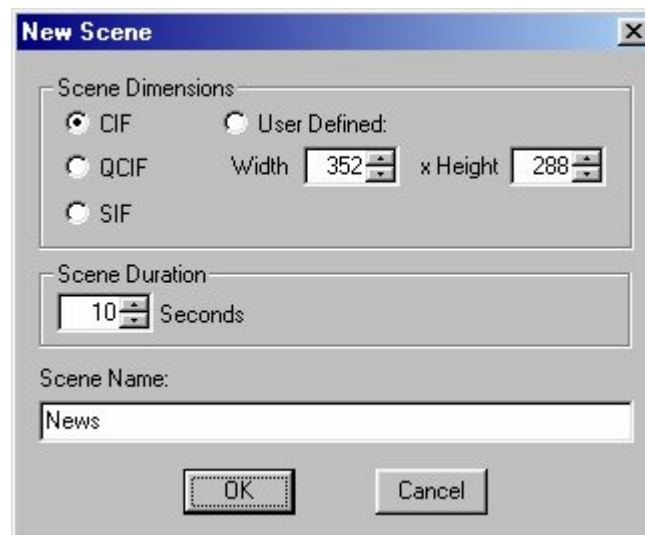



Figura 23 – Parâmetros básicos na criação de uma nova cena.

3.2.2 Criação de um novo objecto de vídeo

A criação de um objecto de vídeo corresponde a isolar um objecto de forma arbitrária da restante informação presente numa sequência de vídeo, para depois o poder usar e manipular individualmente, por exemplo adicionando-o como entidade independente numa nova cena. Um objecto de vídeo tem associado um ficheiro que contém a informação de vídeo (luminância e crominâncias) e um ficheiro que define a forma do objecto. No caso de se estar a tratar com objectos obtidos a partir da segmentação de sequências de vídeo rectangulares, como é o caso quando se está a criar um novo objecto de vídeo no contexto desta aplicação, a informação de vídeo corresponde à sequência a segmentar e a informação de forma contém a máscara que isola o objecto em questão da restante informação na sequência de vídeo. Para criar um novo objecto de vídeo, deve-se escolher a opção *New Video Object* no menu *File* ou seleccionar o botão  (Figura 24).

Para criar um novo objecto de vídeo, devem seleccionar-se a sequência (textura) e a máscara (forma), escolhendo os directórios que contêm os ficheiros `frames.yuv` e `frames.lum`, respectivamente, e ainda o ficheiro `frames.inf` que contém a resolução espacial e a frequência de trama da sequência em questão. O nome `<frames>` pode ser substituído por outro qualquer, desde que o nome seja igual para os três ficheiros referidos (exceptuando a extensão, que difere consoante o tipo de ficheiro). Os directórios não devem conter outros ficheiros (com excepção do ficheiro `frames.msk`, cuja função se explica mais à frente). Se o ficheiro `frames.inf` não existir, assume-se que a sequência tem o formato espacial CIF e que a sua frequência de trama é de 25 tramas por segundo.

Nesta aplicação, um novo objecto de vídeo só pode ser criado como parte de uma sequência rectangular disponível, devendo obter-se a máscara de segmentação usando uma aplicação de segmentação adequada [22]. A trama inicial e final indicam os limites temporais do objecto, relativamente à sequência original. O novo objecto pode ser visualizado premindo o botão *Preview* na janela de diálogo da Figura 24.

A criação de um novo objecto de vídeo pode ser facilitada se existir informação disponível acerca das máscaras existentes para uma determinada sequência (ver ‘*Available Masks*’, na interface à esquerda, na Figura 24). Com esta finalidade, um ficheiro `frames.msk` pode estar disponível no directório da sequência original em questão. Este ficheiro destina-se a indicar a localização das máscaras disponíveis para essa sequência. Deste modo, todos os objectos correspondentes às máscaras disponíveis para uma dada sequência podem ser facilmente seleccionados e visualizados. O exemplo seguinte mostra o conteúdo do ficheiro `frames.msk` que define os objectos e máscaras existentes para a sequência *Container*, tal como se mostra na Figura 24. Cada linha do ficheiro `frames.msk` contém o nome e a localização da máscara que define um objecto.

```
NAME = "sea" PATH = container_obj_0_qcif.all\frames.lum
NAME = "container" PATH = container_obj_1_qcif.all\frames.lum
NAME = "small boat" PATH = container_obj_2_qcif.all\frames.lum
NAME = "flag pole" PATH = container_obj_3_qcif.all\frames.lum
NAME = "sky" PATH = container_obj_4_qcif.all\frames.lum
NAME = "flag" PATH = container_obj_5_qcif.all\frames.lum
```

Recorrendo a este ficheiro, as máscaras podem ser adicionadas ou removidas de um modo bastante simples e intuitivo; para além disso, podem obter-se novos objectos de vídeo criados como a união de outros objectos de vídeo previamente disponíveis, simplesmente seleccionando simultaneamente várias máscaras na janela ‘*Available Masks*’. Os objectos de vídeo que existem na cena e que estão seleccionados em ‘*Available Masks*’ podem ser especialmente visualizados, juntamente com o seu nome, passando sobre eles o cursor do rato, como se ilustra na parte direita da Figura 24 para o objecto ‘*container*’.

A criação de um novo objecto fica completa com a atribuição de um nome em ‘*Object Name*’.

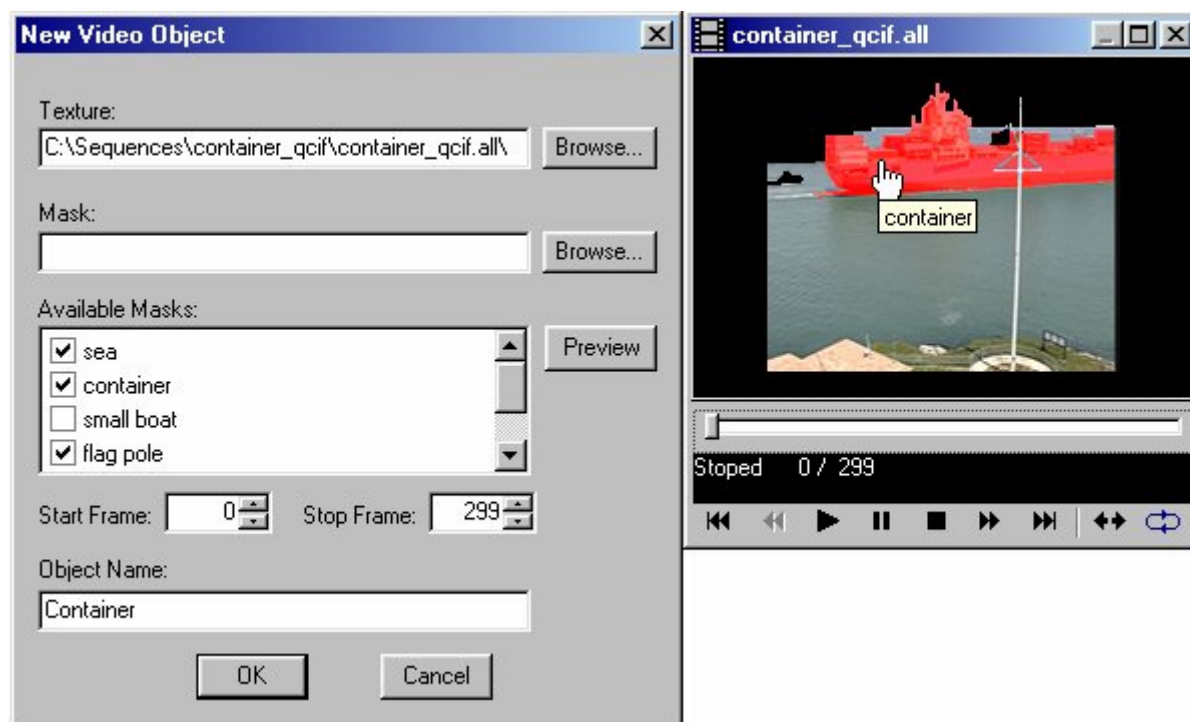


Figura 24 – Criação de um novo objecto de vídeo.

3.2.3 Adição de objectos de vídeo à cena

Depois de criado um novo objecto de vídeo ou seleccionado um objecto que foi previamente gravado, este aparece na árvore de objectos. Quando se selecciona um objecto de vídeo na árvore com o botão direito do rato, surge um menu que permite ao utilizador adicionar o objecto à cena, visualizá-lo, obter informações sobre ele, removê-lo, mudar o seu nome ou ainda gravá-lo (Figura 25). Se se pretender incluir o objecto na cena em construção e a duração do objecto de vídeo em questão for maior do que a duração da cena, aparece uma janela de diálogo que permite ao utilizador ajustar a duração do objecto à duração da cena. Nesta janela de diálogo, podem ser definidas quais as tramas inicial e final, relativamente à sequência original onde este objecto foi segmentado, que definem a parte do objecto a incluir na cena, determinando assim a duração do objecto de vídeo na cena em construção. Depois de o objecto estar incluído na cena, o seu *icon* na árvore de objectos fica vermelho. Por outro lado, os objectos de vídeo que não se encontram incluídos na cena têm um *icon* cinzento na árvore de objectos.

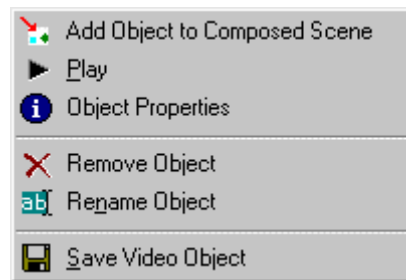


Figura 25 – Menu do objecto de vídeo na árvore de objectos.

Os objectos de vídeo compostos exclusivamente por um plano temporal (VOP no MPEG-4) podem ser adicionados à cena como objectos estáticos. Para estes objectos, surge a opção *Add As Still* no menu da Figura 25 que permite que a imagem em questão apareça em parte ou na totalidade da duração da cena. A definição das tramas em que o objecto estático está presente é feita antes de o adicionar à cena.

Quando o objecto de vídeo já está incluído na cena, surgem novas opções no menu da Figura 25 que acrescentam outras funcionalidades àquelas já referidas atrás (Figura 26).

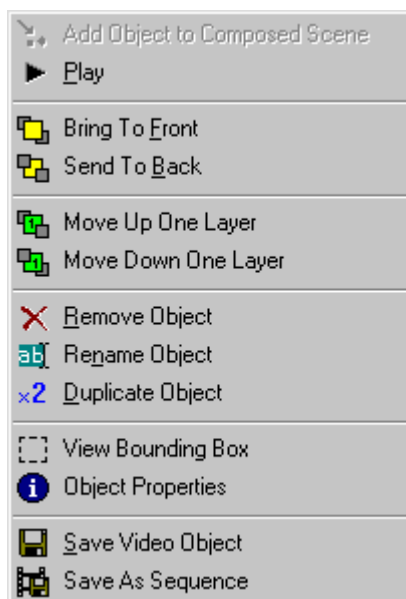



Figura 26 – Menu do objecto de vídeo na árvore de objectos depois de o objecto estar inserido na cena.

Para além das funcionalidades relacionadas com a edição de cena, que permitem definir o nível de profundidade do objecto, é também possível duplicar o objecto em questão, fazendo uma cópia do objecto seleccionado e colocando-a na cena com um pequeno deslocamento espacial relativamente ao original, para uma fácil distinção entre os dois. Para distinguir entre o original e o duplicado, é acrescentado um número (único e em sequência) ao nome do objecto original que foi duplicado. É também possível gravar o objecto em questão como uma sequência, o que permite que sejam armazenados em disco um ficheiro *.yuv com a textura e um ficheiro *.lum com a máscara respectiva. Esta opção é detalhada mais à frente na secção 3.2.5.

3.2.4 Edição da cena

A edição da cena consiste em posicionar os objectos de vídeo, espacial e temporalmente, e ainda em definir os seus parâmetros de codificação e composição.

- **Posição temporal dos objectos de vídeo**

A posição temporal (momentos inicial e final) de cada objecto de vídeo na cena pode ser definida através da barra de tempo, arrastando as barras azuis que indicam a posição inicial e a duração temporal de cada objecto. Para ajudar ao posicionamento temporal de um objecto de vídeo, existe uma janela de diálogo (Figura 27) que fornece toda a informação temporal necessária, e que pode ser mostrada seleccionando a opção *Timer Info* no menu *View* ou premindo o botão . Esta janela de diálogo actualiza os seus valores à medida que a barra de tempo muda de posição.

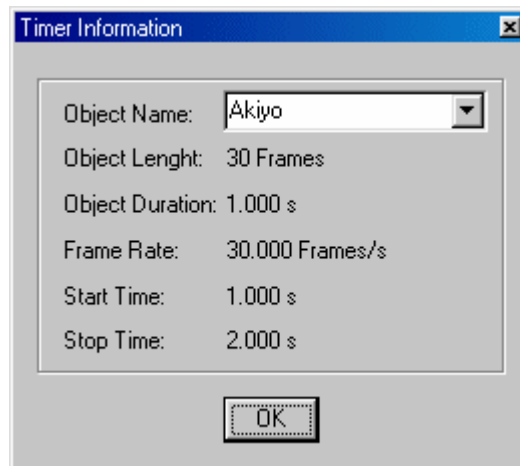


Figura 27 – Informação temporal para o objecto Akiyo.

- **Opções de codificação e composição**

Depois de os objectos de vídeo estarem incluídos na cena, podem ser arrastados com o rato para alterar a sua posição espacial. O nome do objecto aparece quando o ponteiro se mantém fixo sobre este. Quando se selecciona o objecto com o botão direito do rato, surge um menu (com o nome do objecto no topo) que permite ao utilizador controlar várias opções de composição e codificação referentes ao objecto de vídeo em questão (Figura 28). Este menu é uma versão mais simples do menu da Figura 26, que só contém as opções para o posicionamento espacial e para as propriedades do objecto de vídeo.

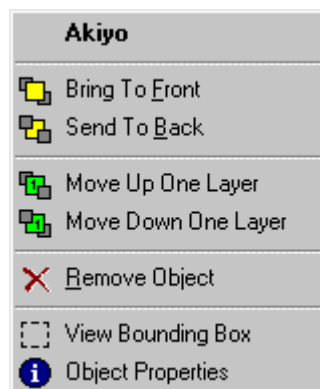






Figura 28 – Menu do objecto de vídeo na cena.

As opções disponíveis neste menu são:

-  **Definição do nível de profundidade** – Altera o nível de profundidade do objecto, deslocando-o mais para trás ou para a frente dos outros objectos. O comando *Bring to Front* coloca o objecto em questão na frente de todos os outros objectos, ou seja com o menor nível de profundidade, enquanto que o comando *Send to Back* coloca o objecto atrás de todos os outros objectos que constituem a cena, ou seja no nível de maior profundidade. Existem dois outros comandos, *Move Up One Layer* e *Move Down One Layer*, que permitem alterar a profundidade de um objecto nível a

nível para que este possa ser colocado na posição desejada, à frente ou atrás de outros objectos.

-  **Remoção** – Remove um objecto da cena, mas mantém-o na árvore de objectos.
-  **Bounding box** – Mostra a *bounding box* do objecto de vídeo seleccionado para o instante temporal corrente. Se a cena estiver em modo *play* quando esta opção é seleccionada, é colocada em pausa. Quando a cena retoma a visualização, a *bounding box* desaparece. Quando se move um objecto com esta opção activa, aparecem na parte inferior do ecrã as coordenadas espaciais do canto superior esquerdo da *bounding box* dentro da cena.
-  **Definição de parâmetros de composição e codificação** – Define as propriedades do objecto de vídeo em termos de composição e codificação (Figura 29). Na janela de diálogo *Size & Position*, o utilizador pode controlar a posição e a dimensão (absoluta ou usando um factor de escala) da *bounding box* do objecto de vídeo. Os valores especificados referem-se ao instante actual, já que a *bounding box* pode variar ao longo do tempo. A opção *Proportional* permite modificar as dimensões do objecto e correspondente *bounding box*, mantendo a sua relação altura/largura. Na janela de diálogo *Time* pode ser escolhida a trama inicial e final para o objecto relativamente à sequência de onde é extraído, podendo eventualmente ser diminuída a sua duração. Também se pode especificar uma nova frequência temporal para esse objecto bem como o momento em que o objecto entra na cena. Surgem depois os parâmetros de codificação que permitem definir de um modo preciso como o objecto de vídeo vai ser codificado. Toda a parte da aplicação relacionada com a codificação será apresentada na secção 3.3.

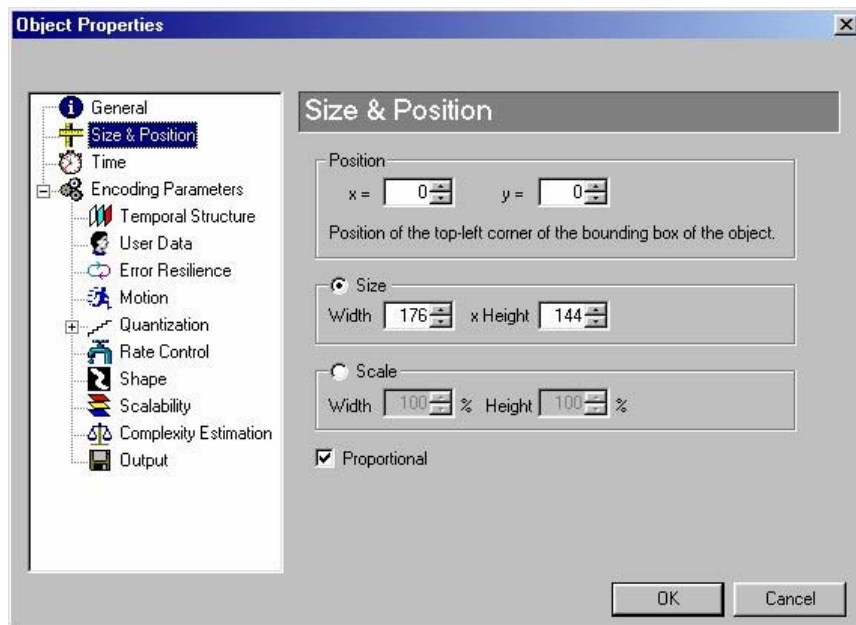











Figura 29 – Definição das propriedades de composição e codificação de um objecto de vídeo.

• Visualização da cena composta

Para visualizar a cena composta devem ser usados os botões representados na Figura 30. Estes botões servem para visualizar ou seleccionar qualquer trama da cena. Da esquerda para a direita na Figura 30, a função de cada um dos botões é:


-  – Saltar para a primeira trama;
-  – Trama anterior;
-  – *Play*;
-  – Pausa;
-  – Parar;
-  – Próxima trama;
-  – Saltar para a última trama;
-  – Saltar para uma trama qualquer, definida pelo utilizador através de uma janela que se abre;
-  – *Play* contínuo, ou seja faz *play* ininterruptamente.

Do lado direito dos botões existe uma barra que indica o progresso da visualização da cena e que pode ser directamente utilizada para avançar ou recuar a visualização da cena. Finalmente, existe um mostrador que indica o estado actual da cena (parada, em pausa ou em *play*), a trama actual e o número total de tramas, o tempo actual em relação ao início e a frequência de trama à qual a cena está a ser visualizada.





Figura 30 – Botões para controlo da visualização da cena.

3.2.5 Gravação de objectos de vídeo e cenas



Tanto os objectos de vídeo como as cenas podem ser gravados em disco. Uma cena é gravada seleccionando a opção *Save Scene* no menu *File* ou premindo o botão , sendo assim criado um ficheiro *.scn que define a cena em questão.

Para gravar um objecto de vídeo, a opção deve ser seleccionada no menu que aparece quando o utilizador prime o botão direito do rato sobre um objecto na árvore de objectos (Figura 26). A gravação de um objecto de vídeo pode ser feita de dois modos:

-  **Por referência** – Como um ficheiro *.vo que representa um objecto de vídeo através da localização da sequência e das máscaras que o definem e ainda a trama inicial e final do objecto relativamente à sequência original;
-  **Independente** – Como uma sequência, ou seja o objecto de vídeo é gravado como um ficheiro *.yuv juntamente com uma máscara que define o objecto (*.lum), sendo criado um novo objecto (através dos novos ficheiros) em disco. Esta opção pode ser útil quando se muda a dimensão de um objecto de vídeo em relação ao original ou quando se definem uniões de máscaras e se deseja guardar o resultado destas operações; esta opção só se encontra disponível para os objectos de vídeo já incluídos na cena em construção/visualização.

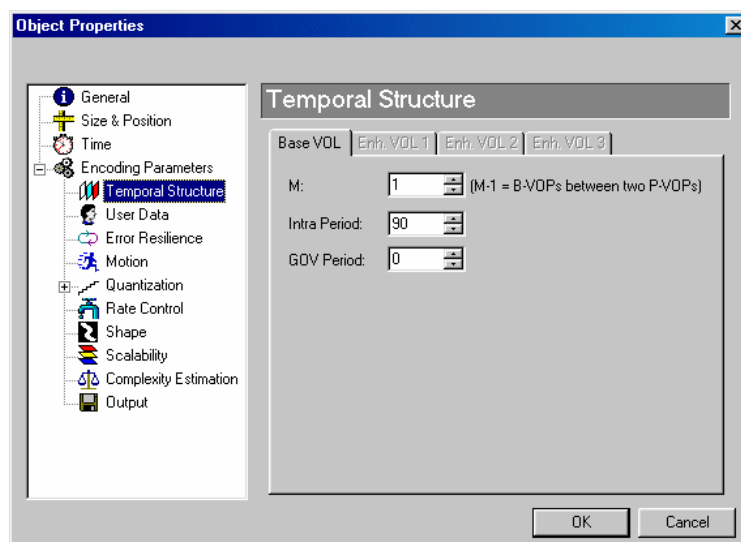
Os ficheiros de cena (*.scn) e de objecto de vídeo (*.vo) contêm referências para sequências que estão gravadas em disco, o que significa que se estas sequências forem mudadas de sítio ou apagadas, tanto os objectos de vídeo como as cenas deixam de poder ser utilizados.

3.2.6 Abertura de objectos de vídeo e cenas

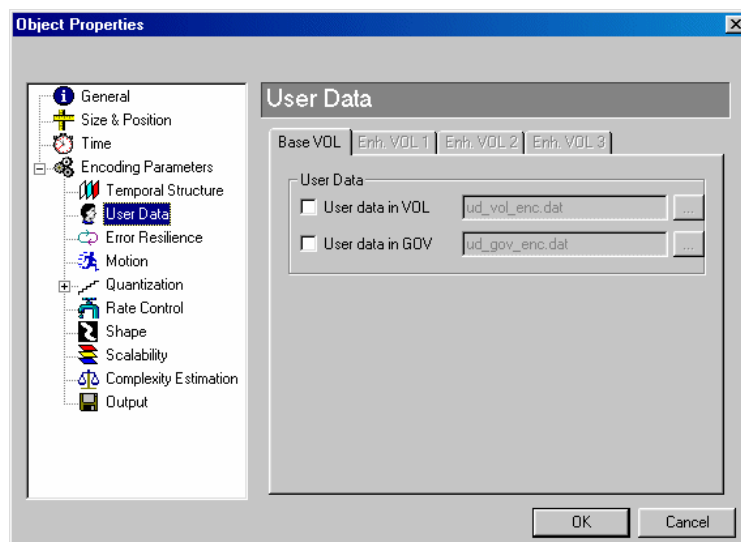
Todos os objectos de vídeo e cenas que foram gravados podem ser abertos para utilização posterior. Para abrir uma cena, deve ser seleccionada a opção *Open Scene* no menu *File* ou premido o botão . Como só uma cena pode ser editada de cada vez, é necessário fechar a cena corrente antes de abrir outra. Um objecto de vídeo pode ser aberto seleccionando a opção *Open Video Object* no menu *File* ou premindo o botão . O objecto de vídeo aberto é incluído na árvore de objectos, onde pode ser visualizado ou acrescentado à cena, como descrito anteriormente.

3.3 Codificador MPEG-4

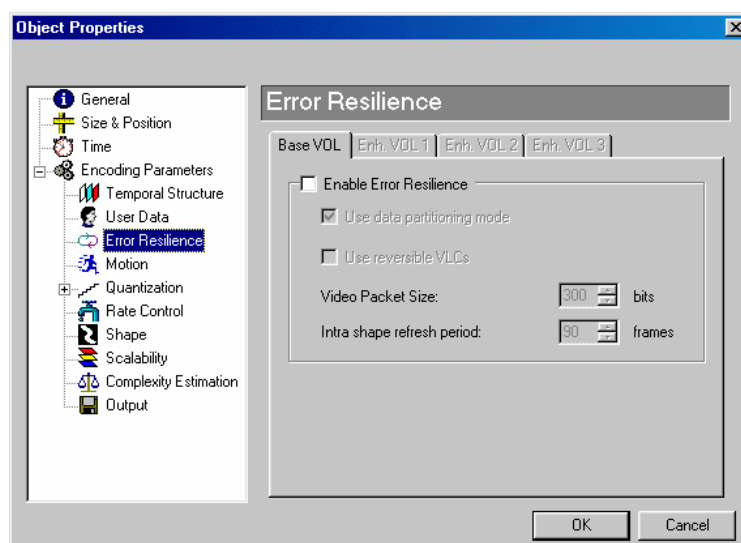
Para codificar uma cena, é necessário definir os parâmetros de codificação, o tipo de cada objecto e o perfil e o nível da cena, tal como explicado no capítulo 2. Os parâmetros de codificação de cada objecto são acedidos através da janela apresentada na Figura 29. A Figura 31 mostra todos os parâmetros que podem ser definidos para a codificação de um objecto de vídeo.



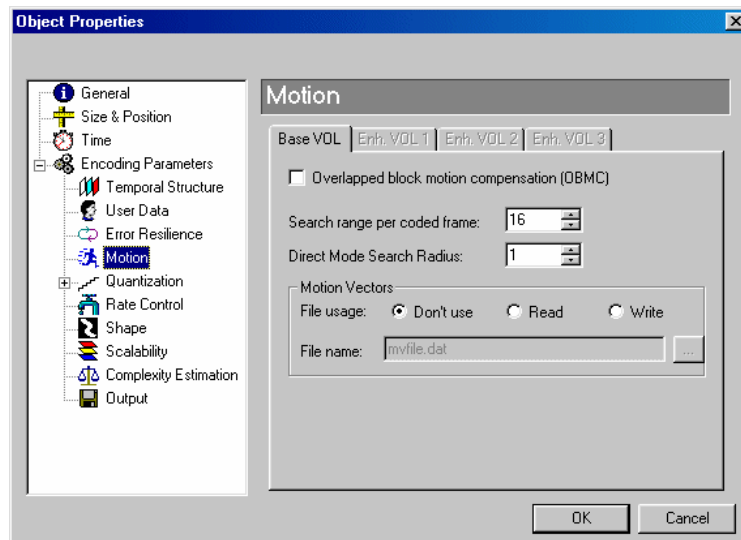
a)



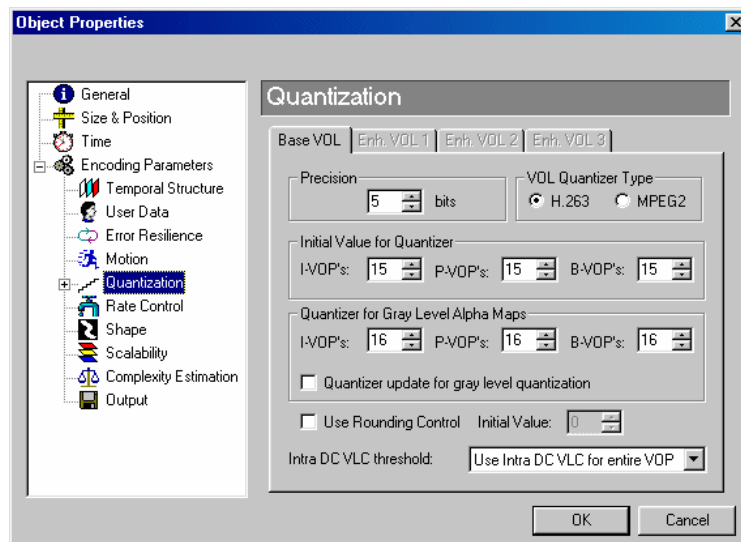
b)



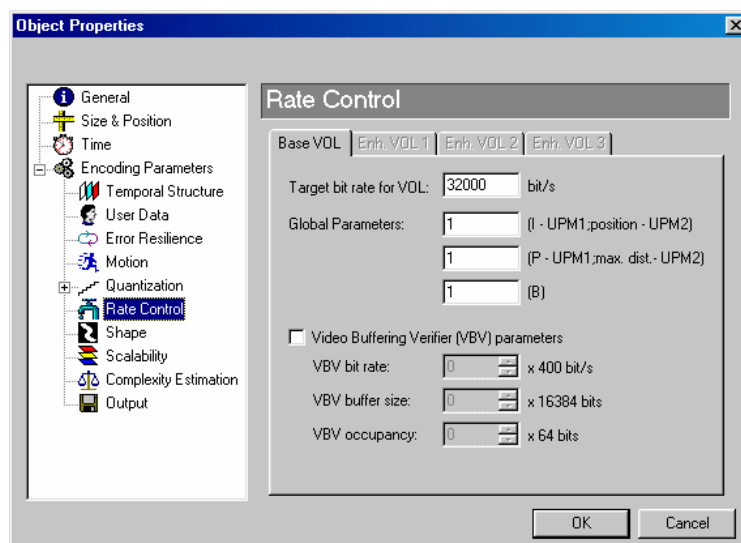
c)



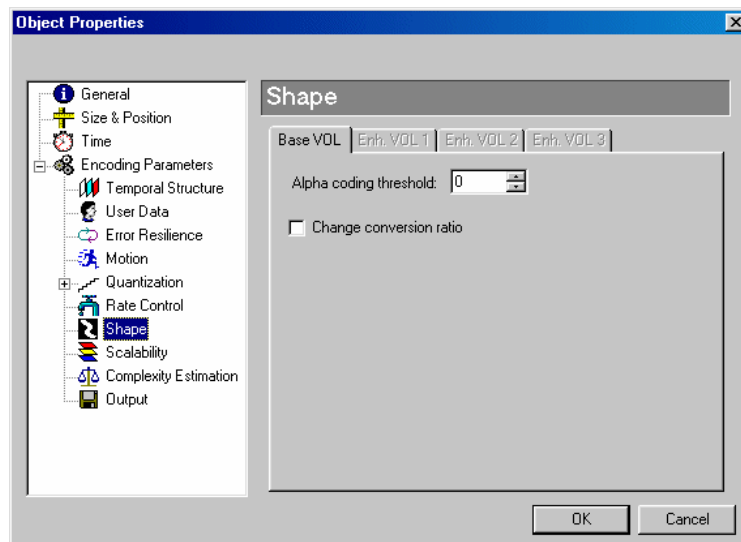
d)



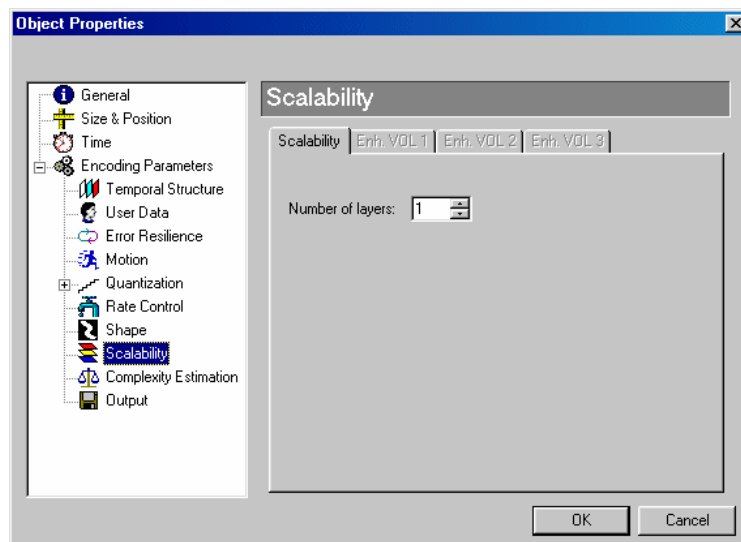
e)



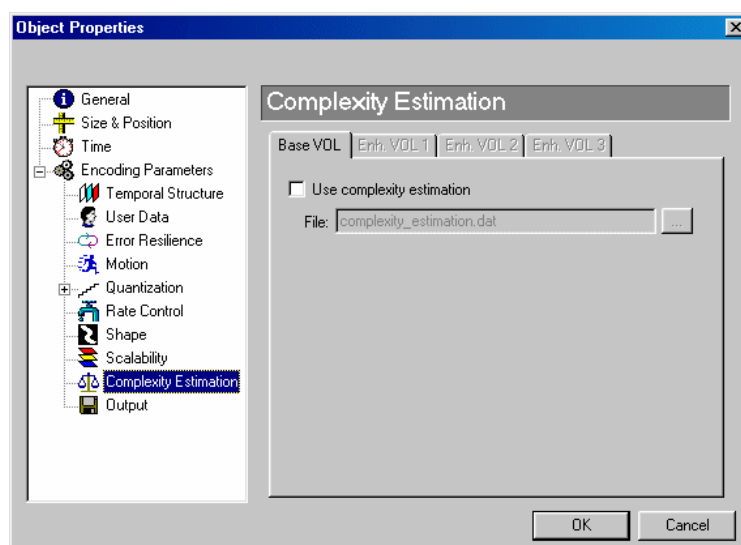
f)



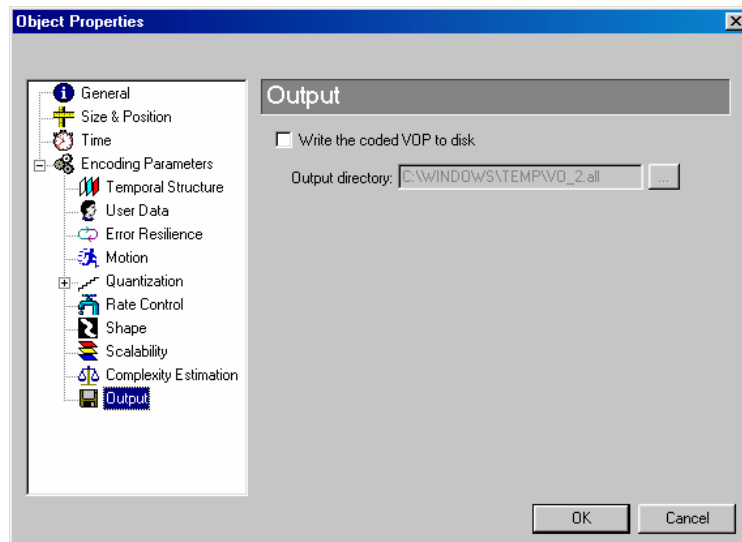
g)



h)







i)









j)

Figura 31 – Definição dos parâmetros de codificação de um objecto de vídeo: a) estrutura temporal; b) dados definidos pelo utilizador, que podem ser incluídos no fluxo binário; c) resiliência a erros; d) movimento; e) quantificação; f) controlo de débito; g) forma; h) escalabilidade; i) estimacão de complexidade; j) directório para a escrita do VOP codificado reconstruído pelo codificador.

Os parâmetros de codificação podem ser agrupados nas seguintes categorias (ver Figura 31):

-  **Temporal Structure** – Permite definir os parâmetros relacionados com a estrutura temporal do objecto de vídeo codificado. Pode-se especificar o número de B-VOPs entre dois P-VOPs, o período entre VOPs *Intra* e o período de inserção de GOVs.
-  **User Data** – Permite indicar a localização de ficheiros com dados definidos pelo utilizador que devem ser incluídos no fluxo binário.
-  **Error Resilience** – Define o uso ou não de ferramentas para resiliência a erros. Caso se desejem utilizar ferramentas para resiliência a erros, deve ser especificado se se utiliza partição de dados e/ou códigos reversíveis de comprimento variável. Também se pode definir o tamanho dos pacotes de vídeo e o período de refrescamento da forma através de codificação em modo *Intra*.
-  **Motion** – Esta janela de diálogo agrupa as opções relacionadas com a estimacão e compensação de movimento. Pode-se definir a utilização de compensação de movimento com sobreposição, mas se esta opção for utilizada o fluxo binário resultante não estará conforme com a norma, pois não existe nenhum *tipo de objecto* na versão 1 da norma MPEG-4 que inclua esta ferramenta (é aliás o único caso da norma MPEG-4 Visual). É também possível especificar a dimensão da zona de procura em *pixels* de (macro)blocos candidatos para a estimacão de movimento e ainda a zona de procura em modo directo (o modo directo consiste em obter os vectores de movimento de um B-VOP a partir dos vectores de movimento de VOPs anteriores e posteriores). Por fim, pode-se especificar um ficheiro donde se lêem ou para onde se escrevem os vectores de movimento, o que pode

facilitar o processo de codificação, por exemplo evitando o cálculo dos vectores quando se quiserem usar vectores já calculados.

-  **Quantization** – Permite definir os parâmetros relacionados com a quantificação. Pode-se escolher a precisão (ou seja o número de passos de quantificação possíveis), o tipo de quantificação (H.263 ou MPEG-2), o valor inicial do passo de quantificação para os VOPs I, P e B para a textura e para a informação de forma multi-nível, o controlo de arredondamento e ainda o limiar de decisão para a utilização de tabelas Intra DC ou Intra AC para os coeficientes DC.
-  **Rate Control** – Esta janela de diálogo permite definir os parâmetros relacionados com o controlo de débito. Pode-se escolher o débito binário alvo para cada camada do objecto de vídeo, parâmetros relacionados com o algoritmo de controlo de débito e ainda os parâmetros relacionados com o VBV, nomeadamente a sua capacidade, débito binário e ocupação inicial.
-  **Shape** – Permite definir parâmetros relacionados com a codificação da forma do objecto, nomeadamente o limiar de codificação (número máximo de *pixels* codificados incorrectamente dentro de um bloco de fronteira) e a utilização de um factor de conversão da dimensão da forma, que permitem codificação com perdas.
-  **Scalability** – Permite definir o número de camadas usadas na codificação do objecto de vídeo. Nesta versão da aplicação só se pode utilizar uma camada por cada objecto de vídeo.
-  **Complexity Estimation** – Permite escolher se é utilizada ou não a ferramenta de estimação de complexidade. Esta ferramenta possibilita ao decodificador o conhecimento de uma estimativa da complexidade da descodificação sem ser necessário proceder à descodificação dos VOPs, para que assim o decodificador possa adequar o seu procedimento à complexidade da cena a descodificar. Para isso, são transmitidas no fluxo binário estatísticas acerca dos algoritmos, modos e parâmetros utilizados para codificar o VOP em questão.
-  **Output** – Permite escrever o VOP codificado reconstruído pelo codificador para o disco como uma sequência *.yuv.

Todos os parâmetros referidos atrás têm valores de defeito, sendo assim possível codificar uma cena sem ser necessário especificar todos os parâmetros individualmente. Deste modo, um utilizador que não conheça o significado de todos os parâmetros pode compor e codificar uma cena de um modo simples.

3.3.1 Controlo de débito

Para que o fluxo binário esteja em conformidade com a norma, o codificador deve implementar algoritmos para o controlo do débito binário. O controlo de débito deve assegurar que o mecanismo de verificação de vídeo é cumprido, ou seja que a capacidade das memórias que o constituem não é ultrapassada. Os algoritmos de controlo de débito não são normalizados por isso não ser necessário para garantir interoperabilidade, o que fomenta a competição e inovação por parte da indústria, com resultados evidentes na obtenção de

melhores e mais eficientes equipamentos e algoritmos. Para esta tese utiliza-se um codificador com controlo de débito, já que é importante trabalhar com fluxos binários que não violem o mecanismo de verificação de vídeo.

O controlo de débito pode ser definido ao nível do objecto (débito atribuído independentemente a cada objecto) ou ao nível da cena (débito total atribuído dinamicamente ao conjunto dos objectos na cena tendo em conta as suas características). No controlo de débito ao nível do objecto, o utilizador pode definir os parâmetros de controlo de débito para cada camada de cada objecto (Figura 32). Como a escalabilidade não está implementada nesta versão do codec, só a camada de base está disponível para o controlo.

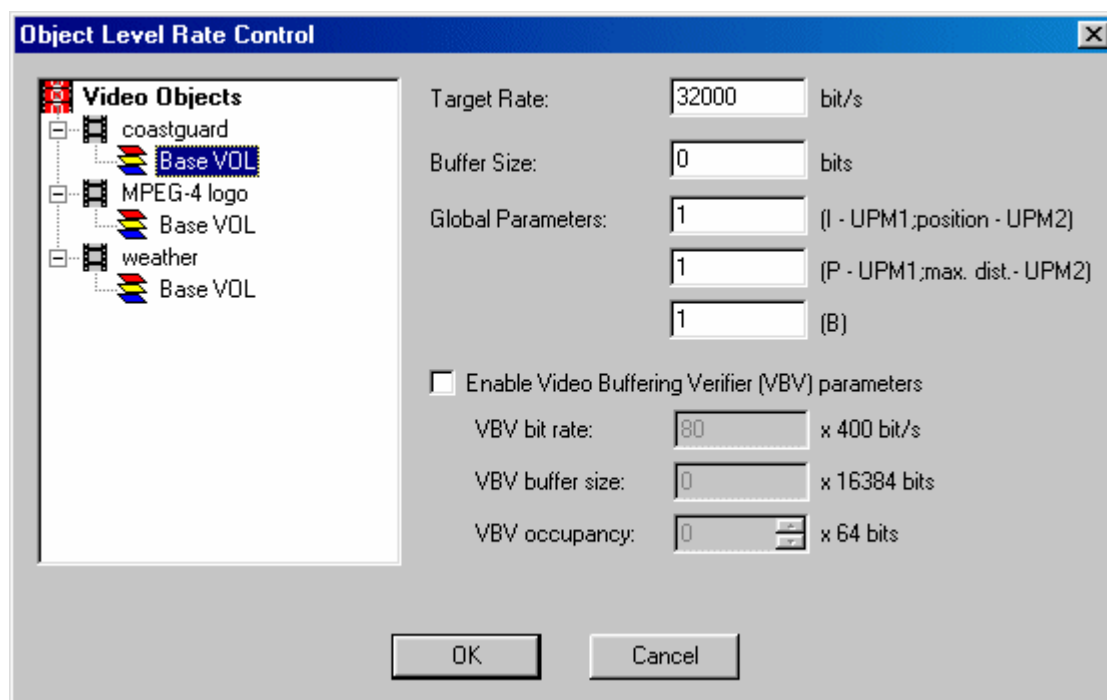


Figura 32 – Controlo de débito ao nível do objecto.

Na janela referente ao controlo de débito ao nível da cena é definido qual o tipo de controlo de débito que se pretende utilizar no processo de codificação (Figura 33). Os tipos de controlo de débito ao nível da cena podem ser:

- **None** – Não é utilizado qualquer tipo de controlo de débito. Neste caso, o codificador pode gerar fluxos binários que não estão em conformidade com a norma, caso estes não verifiquem o mecanismo de verificação de vídeo. Nestas condições, não há a garantia que um dado descodificador possa descodificar os fluxos binários recebidos.
- **Joint** – O débito binário total disponível é distribuído dinamicamente por todos os objectos presentes na cena, segundo algumas características previamente escolhidas. Neste caso, o utilizador define o débito alvo e o tamanho da memória de saída do descodificador para toda a cena. Os valores de defeito para estes dois parâmetros dependem da combinação perfil@nível escolhida para a cena. O utilizador também pode definir os pesos correspondentes aos vários parâmetros/critérios utilizados na distribuição do débito binário entre os vários objectos presentes na cena, e.g. tamanho, complexidade, actividade. Os valores destes pesos devem somar um.

- **Independent** – O débito é especificado de modo independente para cada objecto de vídeo e não varia ao longo do tempo segundo as características dos objectos como na solução acima, ainda que o valor do débito binário total utilizado possa ser o mesmo.

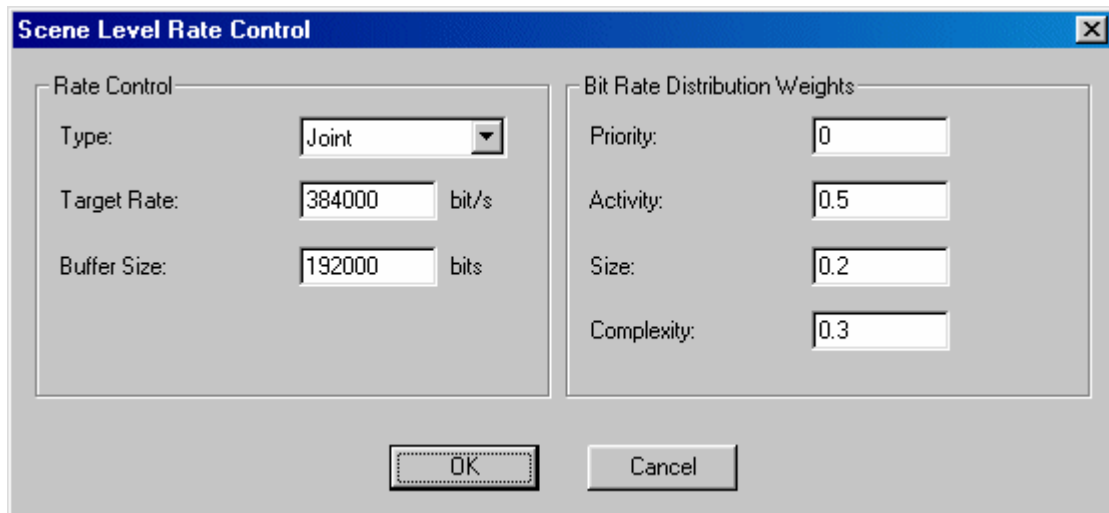


Figura 33 – Controlo de débito ao nível da cena.

3.3.2 Perfis, níveis e tipos de objecto

A definição do perfil e do nível da cena e do *tipo de objecto* para cada objecto, antes de se proceder à codificação, é um processo fundamental. O perfil define o conjunto de ferramentas a utilizar na codificação e o nível especifica as restrições impostas ao perfil, através da limitação de alguns parâmetros relevantes. O *tipo de objecto* corresponde a uma lista de ferramentas que podem ser utilizadas para a codificação de um dado objecto. Até à data da escrita desta tese, e tanto quanto é do conhecimento do autor, a presente aplicação é a primeira a nível mundial, disponível publicamente, a oferecer este tipo de funcionalidade, ou seja a capacidade de gerar fluxos binários de vídeo MPEG-4 conformes com uma dada combinação perfil@nível.

Na janela de diálogo que permite seleccionar o *tipo de objecto* para cada um dos objectos de vídeo presentes na cena (Figura 34), pode ser escolhido um dos seguintes tipos: *Simple*, *Simple Scalable*, *Core*, *N-Bit* e *Main*. Estes *tipos de objecto* são os definidos na versão 1 da norma MPEG-4 Visual [3] e estão descritos no capítulo 2 da presente tese. Já depois de concluída a versão 1 da norma foram adicionados os *tipos de objecto Studio* e *FGS* (*Fine Grain Scalability*) que não são suportados por esta versão da aplicação.

A aplicação rejeita a escolha incorrecta do *tipo de objecto*, por exemplo não permite que o utilizador seleccione o *tipo de objecto Simple* se o objecto de vídeo tiver forma arbitrária por este *tipo de objecto* não incluir a ferramenta de codificação da forma.

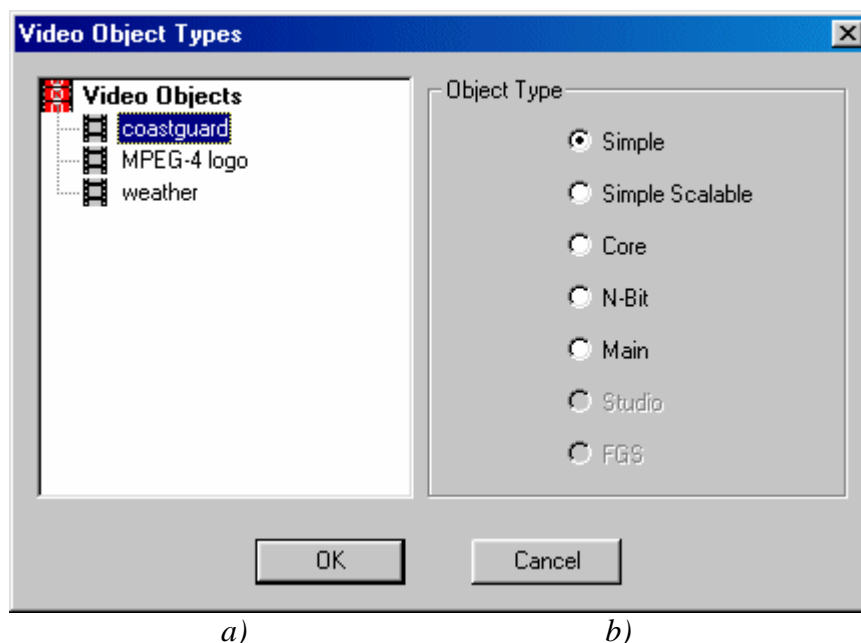


Figura 34 – Tipos de objecto de vídeo: a) objectos na cena; b) escolha do tipo de objecto.

O perfil e o nível da cena podem ser escolhidos pelo utilizador através da janela de diálogo da Figura 35. Os seguintes perfis estão disponíveis: *Simple*, *Simple Scalable*, *Core*, *Main* e *N-Bit*. Todos estes perfis foram definidos na versão 1 da norma MPEG-4 Visual e estão descritos no capítulo 2 da presente tese. Já depois de concluída a versão 1 da norma foi adicionado o nível 0 ao perfil *Simple* e foram incluídos dois novos perfis: *Studio* e *FGS* (*Fine Grain Scalability*). Os novos perfis e o nível 0 do perfil *Simple* não são suportados por esta versão da aplicação.

A aplicação rejeita a escolha incorrecta de perfis e níveis, por exemplo, não permite que seja escolhido o perfil *Simple* se existirem objectos com forma arbitrária presentes na cena. Também verifica se o número de objectos presentes na cena não excede o número máximo de objectos permitidos pela combinação perfil@nível escolhida.

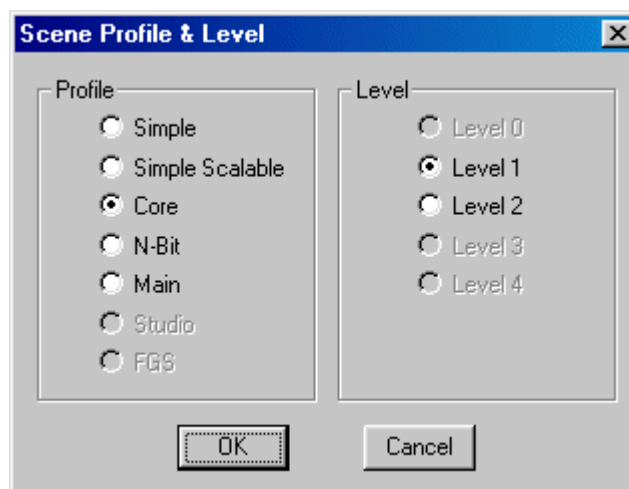



Figura 35 – Escolha do perfil e nível para a cena.

Depois de a cena ter sido composta, de terem sido escolhidos os parâmetros de codificação para cada objecto e de terem sido definidos correctamente o perfil, o nível e os tipos dos objectos, o processo de codificação pode começar. Para iniciar o processo de codificação deve ser seleccionada a opção *Encode* no menu *Codec* ou premido o botão . Uma cena só pode ser codificada se foi escolhido um *tipo de objecto* correcto para cada objecto de vídeo e se foi escolhida uma combinação perfil@nível adequada à cena a codificar. Se a cena contiver objectos de vídeo aos quais foi mudada a escala (dimensões com uma escala diferente de 100%) ou que foram definidos através de uma união de máscaras, estes objectos devem ser gravados antes de se iniciar o processo de codificação. O utilizador é avisado da necessidade de realizar esta operação através de uma mensagem no ecrã.

Durante o processo de codificação, aparece no ecrã um gráfico que mostra a evolução do preenchimento das memórias do mecanismo de verificação de vídeo (Figura 36), tal como foi descrito no capítulo 2. Nesse gráfico é mostrada a evolução das memórias dos *Video rate Buffer Verifier* (VBV), *Video Complexity Verifier* (VCV), *Boundary macroblocks Video Complexity Verifier* (B-VCV) e *Video reference Memory Verifier* (VMV), conforme se pode ver na Figura 36. Esta informação gráfica permite, muito facilmente, analisar a complexidade da cena em codificação em termos das várias memórias do mecanismo de verificação de vídeo, uma vez que indica o enchimento em termos percentuais destas memórias. Devido às acções do controlo de débito, nenhuma das memórias deve alguma vez passar os 100% ou descer abaixo dos 0%.

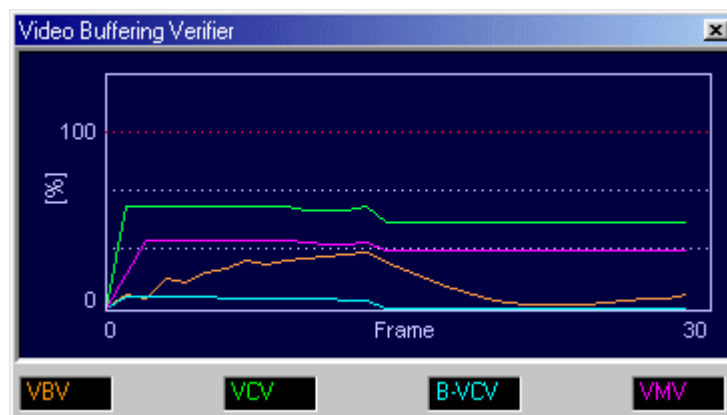


Figura 36 – Gráfico com o enchimento das várias memórias do mecanismo de verificação de vídeo.

Durante a codificação da cena são mostradas outras informações úteis na parte inferior do ecrã. Entre estas informações estão avisos que indicam se as dimensões típicas da cena para um determinado perfil e nível foram ultrapassadas, se a capacidade das memórias foi excedida (levando a fluxos binários que não estão em conformidade com a norma) ou se se “saltaram” VOPs (*skipping*). Todas estas acções indicam uma cena que é demasiado complexa para ser codificada “normalmente” no perfil@nível escolhido. A indicação de que a capacidade das memórias foi excedida só pode acontecer se o controlo de débito não estiver em funcionamento, pois caso contrário isso nunca acontecerá por serem tomadas as acções que o impedem.

Depois de terminado o processo de codificação, as estatísticas da codificação podem ser visualizadas seleccionando a opção *Encoder Statistics* no menu *Codec* (Figura 37). Os parâmetros medidos para cada objecto são o número de bits utilizados, o número médio de bits por VOP, o número médio de bits de forma por VOP e o valor médio da relação sinal ruído de pico (PSNR) para os sinais Y, U e V.

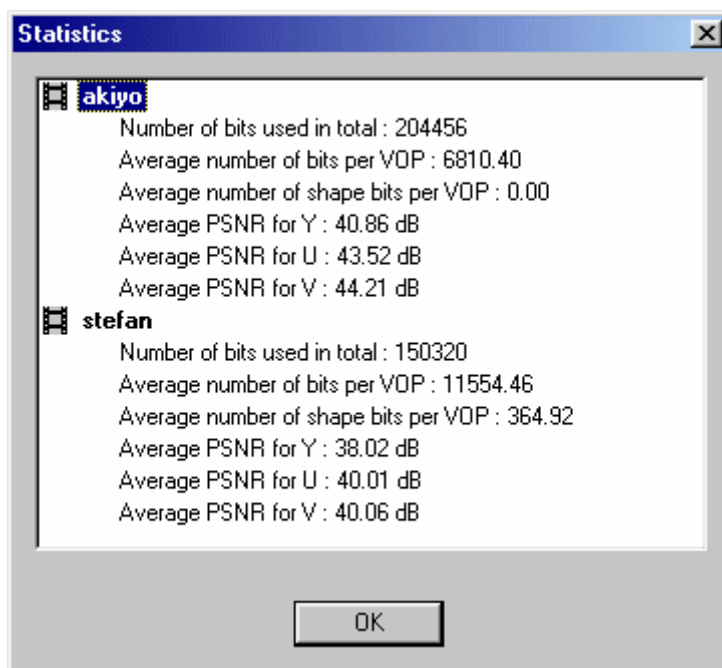



Figura 37 – Estatísticas da codificação dos objectos Akiyo e Stefan.

3.4 Descodificador MPEG-4

Com a codificação da cena são gerados ficheiros com os fluxos binários para cada objecto de vídeo (*.bits) e os ficheiros *.scn e *.cod com a informação de composição que permite decodificar os objectos de vídeo presentes na cena. Esta informação substitui o fluxo de composição BIFS, especificado na norma MPEG-4 Sistemas [8], que não foi implementado nesta aplicação. Para decodificar a cena, ou seja os objectos que a compõem, deve ser seleccionada a opção *Decode* do menu *Codec* ou o botão . Na janela de diálogo mostrada na Figura 38, o utilizador pode seleccionar quais os objectos de vídeo que deverão ser decodificados na cena em questão e que irão compor a cena resultante. O tempo inicial de acesso aleatório, que indica o instante do fluxo binário onde se inicia a decodificação, pode ser especificado em milissegundos. Note-se que esta opção só faz sentido se os fluxos binários contiverem GOVs, que definem pontos de entrada aleatória no fluxo. Os GOVs podem ser definidos através da especificação do período entre GOVs; esta definição é feita na janela correspondente à estrutura temporal nos parâmetros de codificação do objecto de vídeo (Figura 31a). Finalmente, deve ser especificado um directório para os objectos de vídeo decodificados, para a cena total (rectangular) composta por todos os objectos e para o ficheiro *.scn que descreve a cena decodificada.

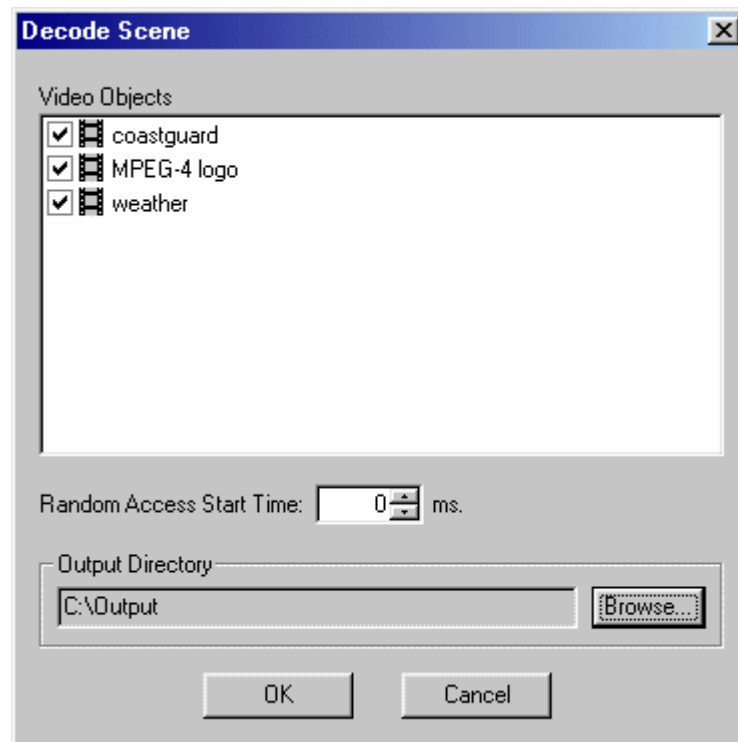


Figura 38 – Janela de diálogo para a decodificação de uma cena.

3.5 Visualização de cenas

No menu *Play*, o utilizador pode controlar a visualização de cenas baseadas em tramas ou baseadas em objectos. Uma cena baseada em tramas é uma cena rectangular já composta que só pode ser vista e não manipulada, por os objectos já não existirem independentemente uns dos outros; corresponde a um ficheiro *.yuv e o seu directório deve ser seleccionado para se visualizar a cena. Uma cena baseada em objectos é uma cena onde os objectos de vídeo podem ser acedidos e manipulados individualmente, e corresponde a um ficheiro *.scn que contém as dimensões da cena e a localização em disco dos vários objectos de vídeo que compõem a cena, bem como a sua posição espacial e temporal. Para abrir uma cena baseada em objectos, terá de se fechar a cena aberta (se houver uma cena em construção/visualização), uma vez que apenas uma cena pode ser processada de cada vez. As cenas baseadas em tramas podem ser visualizadas sem fechar a cena em construção. Neste menu existem também opções que permitem ver a última cena codificada ou decodificada, baseada em tramas ou baseada em objectos.

3.6 Ficheiros gerados pela aplicação

A aplicação gera ficheiros de vários tipos. Os directórios *.all contêm os ficheiros que constituem as sequências de vídeo (frames.yuv) e as máscaras (frames.lum). Os ficheiros *.bits guardam os fluxos binários gerados pelo codificador. Os ficheiros *.cod contêm informação de cena que é passada ao decodificador para que este possa decodificar os objectos seleccionados pelo utilizador, ou a cena completa se todos os objectos que a constituem forem seleccionados. Entre esta informação encontra-se a dimensão e frequência

de trama da cena e informação de composição¹⁰ relativa aos objectos de vídeo que a constituem. Para cada um destes objectos, é indicado o seu nome, a sua posição temporal e espacial dentro da cena e ainda a localização no disco da sequência e máscaras que definem o objecto. Os ficheiros com extensão `*.scn` representam uma cena, contendo informação acerca da sua duração e dimensões e ainda dos objectos de vídeo que a compõem, nomeadamente a sua posição espacial e temporal, frequência de trama, escala e localização no disco da sequência e máscaras que os definem. Note-se que os ficheiros `*.cod` contêm mais informação que os ficheiros `*.scn`, nomeadamente os parâmetros necessários à descodificação.

Existem ainda os ficheiros com extensão `*.vo` que representam um objecto de vídeo, contendo a localização da sequência e das máscaras que o definem e ainda a trama inicial e final do objecto relativamente à sequência original.

Quando se codifica e descodifica uma cena composta por objectos de vídeo, usa-se um conjunto de regras em termos de nomes, para uma mais fácil compreensão do conteúdo e objectivo dos ficheiros gerados:

- `NOME_CENA.scn` – Ficheiro que representa a cena `NOME_CENA`; contém informação sobre as suas dimensões, duração e objectos de vídeo que a compõem.
- `NOME_CENA_frmscn_enc.all` – Cena composta gerada pelo codificador para a cena `NOME_CENA`; esta cena é uma cena rectangular (baseada em tramas) com todos os objectos de vídeo já compostos.
- `NOME_VO_VOLn.bits` – Fluxo binário para a camada `n` do objecto de vídeo `NOME_VO`.
- `NOME_CENA.cod` – Ficheiro com informação necessária para descodificar a cena `NOME_CENA` (*input* para o descodificador).
- `NOME_CENA_frmscn_dec.all` – Cena composta gerada pelo descodificador para a cena `NOME_CENA`; esta cena é uma cena rectangular (baseada em tramas) com todos os objectos de vídeo já compostos.
- `NOME_CENA_objscn_dec.scn` – Cena composta gerada pelo descodificador para a cena `NOME_CENA`; esta cena é baseada em objectos e é constituída pelos objectos de vídeo que foram descodificados, podendo estes ser manipulados.
- `NOME_VO_vo_dec.all` – Directório onde está guardado o objecto de vídeo descodificado `NOME_VO` no formato YUV.
- `NOME_VO_vo_mask_dec.all` – Directório onde está guardada a máscara para o objecto de vídeo descodificado `NOME_VO`.

¹⁰ Esta informação de composição é semelhante à informação que se incluiria no fluxo de composição MPEG-4 definido em BIFS (*Binary Format for Scene Description*) mas que não foi implementado no contexto desta tese.


3.7 Outras funcionalidades

Para além de todas as funcionalidades da aplicação já referidas, existem outras que permitem especificar os directórios de defeito, “ler” a cor dos *pixels* da cena e ver o manual da aplicação.


3.7.1.1 Directórios de defeito

Com o comando *Default Directories* do menu *Options*, o utilizador pode especificar os directórios onde a aplicação pode abrir ou gravar ficheiros por defeito. Os directórios de defeito podem ser especificados para máscaras, objectos de vídeo, cenas e ficheiros de saída do codificador e do decodificador. Quando estes directórios são definidos, todas as acções são facilitadas porque todos os ficheiros são guardados ou abertos nos sítios adequados.

3.7.1.2 Cor de um *pixel*

O comando *Show Color* pode ser accionado através do menu *Options* ou através do botão  e permite ao utilizador “ler” a cor de qualquer *pixel* da cena, movimentando o cursor do rato sobre esta. Os valores das componentes de cor são mostrados nos espaços de cor RGB e YUV e aparecem do lado direito da área de trabalho por baixo da barra de tempo.

3.7.1.3 Manual da aplicação

O manual da aplicação pode ser visto seleccionado a opção *Tutorial* no meu *Help* ou premindo o botão . Este manual explica o funcionamento da aplicação, descrevendo os seus menus e janelas de diálogo, para que o utilizador possa mais facilmente tirar partido de todas as suas funcionalidades.

3.8 Conclusões

Neste capítulo foi feita uma apresentação detalhada da aplicação desenvolvida no contexto desta tese. A aplicação permite a criação, codificação e decodificação de conteúdo de vídeo MPEG-4 de um modo amigável e intuitivo. As cenas compostas segundo as necessidades e gosto dos autores podem ser codificadas com um determinado perfil e nível, seleccionado pelo utilizador, de acordo com a norma MPEG-4 Visual. Todos os parâmetros referentes à composição e codificação da cena podem ser modificados de um modo bastante simples. Tanto quanto se sabe, esta é a primeira aplicação a nível mundial a disponibilizar publicamente esta funcionalidade. O fluxo binário resultante pode ser decodificado e os objectos decodificados são individualmente acessíveis na cena.

Na página do Grupo de Imagem do Instituto de Telecomunicações (<http://amalia.img.lx.it.pt/ImageGroup>) está disponível um *tutorial* sobre a aplicação (Figura 39), semelhante ao manual da aplicação referido na secção 3.7.1.3. Este *tutorial* permite que qualquer pessoa conheça a aplicação, tendo a versão executável sido pedida por muitas empresas, universidades e institutos de investigação de vários países.

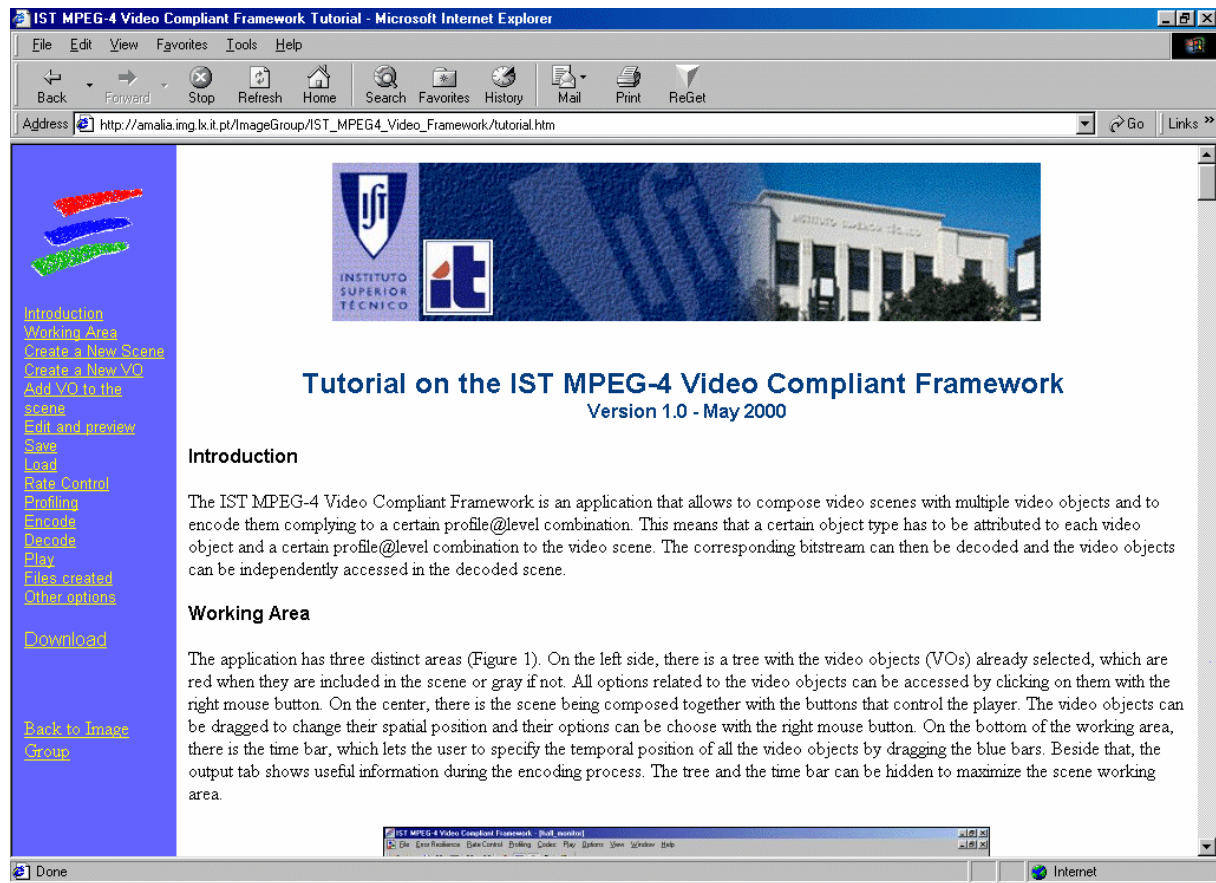


Figura 39 – Tutorial sobre a aplicação, disponível a página do Grupo de Imagem.

Capítulo 4

Optimização do Descodificador de Vídeo MPEG-4

4.1 Introdução

Para que se possa avaliar de forma credível a complexidade da descodificação dos vários tipos de macroblocos usados na codificação de vídeo segundo a norma MPEG-4 Visual é necessário utilizar um descodificador optimizado, ou seja um descodificador onde os algoritmos e processos implementados sejam tais que o peso em termos de cálculo computacional e de utilização de memória sejam o mais representativos possível da complexidade efectiva de cada tipo de macrobloco aqui medida através do seu tempo de descodificação. Com este objectivo, foram efectuadas modificações num dos descodificadores MPEG-4 vídeo incluídos na norma MPEG-4: Parte 5 – *Reference Software* [11] que, para além de permitirem a obtenção de estimativas mais fiáveis da complexidade de descodificação dos vários tipos de macroblocos, resultaram num descodificador significativamente mais rápido, como se mostrará mais adiante neste capítulo. A norma MPEG-4: Parte 5 – *Reference Software* inclui dois descodificadores de vídeo, um em linguagem C disponibilizado pelo projecto Europeu ACTS MoMuSys (*Mobile Multimedia Systems*) no qual o Instituto Superior Técnico participou e outro, em linguagem C++, disponibilizado pela Microsoft.

Neste capítulo descrevem-se as alterações efectuadas ao descodificador de vídeo MoMuSys. Estas alterações tiveram como principais objectivos o aumento da rapidez do descodificador e a obtenção de estimativas credíveis da complexidade relativa dos vários tipos de macroblocos relevantes.

4.2 O Codec de Vídeo MoMuSys/IST

No âmbito do desenvolvimento da norma MPEG-4 Visual surgiram duas implementações em *software* das ferramentas correspondentes à codificação de vídeo, denominadas como implementações de referência por servirem para esclarecer as ambiguidades que o texto da norma Visual (parte 2) possa ter, e que foram incluídas na parte 5 da norma MPEG-4. Estas implementações são comumente designadas por implementações Microsoft e MoMuSys devido à sua origem. Todas as ferramentas de codificação de vídeo, aceites no contexto da norma MPEG-4 Visual, tiveram a seu tempo de ser incluídas e testadas no contexto de cada uma destas duas implementações *software* para que pudessem passar a fazer parte da norma.

O codificador e o decodificador de vídeo MoMuSys implementam em *software* muitas das ferramentas da Parte 2 da norma MPEG-4 [3], ou seja todas as ferramentas de codificação de vídeo (excluem-se as ferramentas de codificação de objectos visuais sintéticos como faces 3D animadas e malhas 2D). Este codec foi desenvolvido no âmbito do projecto europeu ACTS MoMuSys com o objectivo de demonstrar e testar as funcionalidades da norma MPEG-4, permitindo melhorar a norma durante a sua fase de especificação através das experiências realizadas. Para a sua implementação foi utilizada a linguagem de programação C que é uma linguagem amplamente difundida e que permite que o *software* possa ser compilado em qualquer tipo de máquina e sistema operativo. A opção pela linguagem C em vez de uma linguagem de programação por objectos (por exemplo, o C++) traz contudo algumas dificuldades, nomeadamente na organização e acesso das estruturas de dados e na reutilização e extensão de partes do programa, entre muitas outras dificuldades que habitualmente surgem em programas grandes e complexos.

O designado *codec de vídeo* é constituído por um codificador e por um decodificador. O codificador e decodificador MoMuSys implementam todas as ferramentas de codificação de vídeo com excepção das ferramentas ligadas à conformidade, ou seja que aplicam os conceitos de perfil e nível através do correspondente mecanismo de verificação de vídeo, como explicado no capítulo 2. Nesta tese é utilizada uma versão do codec, já alterada no Instituto Superior Técnico pelo Eng. Paulo Nunes, e que tem já implementado o mecanismo de verificação de vídeo para vários perfis e níveis. Esta implementação será de seguida designada por codec MoMuSys/IST.

Em termos de código fonte, o codec está dividido em três grandes módulos: codificador, decodificador e funções comuns. A estrutura do código de partida privilegia a compreensão e a facilidade de implementação e extensão em detrimento da eficiência, quer em termos de memória como de velocidade¹¹. Por outro lado, alguns tipos de macroblocos, nomeadamente os que utilizam predição, são bastante penalizados no código original devido a sucessivas reservas e libertações de memória que conduzem a um acréscimo considerável do tempo para a sua decodificação. O codificador e o decodificador estão completamente integrados no ambiente de composição, codificação e decodificação desenvolvido no contexto desta tese, e que foi apresentado no capítulo 3. Nesse ambiente todos os parâmetros necessários para a

¹¹ Note-se que este código foi utilizado por dezenas de especialistas que ao longo do desenvolvimento da norma MPEG-4 Visual tiveram de nele integrar as muitas ferramentas de codificação desenvolvidas, com vista a testar o seu funcionamento numa plataforma comum. É pois natural que o código não seja um exemplo de boa estruturação e optimização porque esses nunca foram os principais objectivos em questão.

codificação e decodificação de uma cena com vários objectos podem ser controlados através de uma interface gráfica bastante amigável.

Tendo em vista os objectivos descritos anteriormente, foram efectuadas as alterações ao decodificador de vídeo MoMuSys/IST que se descrevem de seguida.

4.3 Melhorias ao Descodificador de Vídeo MoMuSys/IST

Para efectuar melhorias ao decodificador de vídeo MoMuSys/IST, foi necessário proceder à identificação das funções que condicionam com maior intensidade o desempenho do decodificador. Descrevem-se de seguida as condições de teste utilizadas, o desempenho do decodificador indicando quais as funções que exigem mais tempo de processamento e as alterações efectuadas para tornar o processo de decodificação mais rápido.

4.3.1 Condições de teste

Para avaliar o desempenho do decodificador, antes e depois da optimização, foram escolhidas várias sequências de teste que abarcam vários casos relevantes, nomeadamente sequências em formato QCIF (176×144 *pixels* de luminância) e CIF (352×288 *pixels* de luminância), com sub-amostragem de crominância 4:2:0, rectangulares (um único objecto rectangular na cena) e com objectos de forma arbitrária. Foram ainda utilizados dois perfis visuais diferentes (*Simple* e *Core*) e para cada um destes foram utilizados dois níveis diferentes.

Assim, as sequências utilizadas foram:

- **Akiyo** – Sequência em formato QCIF constituída por um único objecto de vídeo, com forma rectangular, e com uma frequência de trama de 30 tramas por segundo (Figura 40a). Esta sequência é constituída por 300 tramas (10 segundos) que foram codificadas no nível 2 do perfil *Simple*, com um débito binário de 128 kbit/s. Para estudar a influência da escolha do perfil no tempo gasto pelas funções no decorrer da decodificação, esta sequência foi codificada também no nível 1 do perfil *Core* utilizando o mesmo débito binário de 128 kbit/s. Para avaliar a influência do nível, a mesma sequência foi ainda codificada no nível 3 do perfil *Simple*, com um débito binário de 384 kbit/s.
- **News** – Sequência em formato CIF constituída por um único objecto de vídeo, com forma rectangular, a 30 tramas por segundo (Figura 40b). Foram codificadas 300 tramas (10 segundos) no nível 3 do perfil *Simple*, com um débito binário de 384 kbit/s. Para determinar a influência do perfil na decodificação, esta sequência foi também codificada no nível 2 do perfil *Core*, com o mesmo débito binário de 384 kbit/s.
- **Coastguard** – Sequência em formato QCIF constituída por quatro objectos de vídeo com forma arbitrária: a água, a terra e os dois barcos (Figura 40c). Esta sequência, ou seja todos os seus objectos, apresenta uma frequência de trama/VOP de 30 tramas/VOPs por segundo. Foram codificadas 300 tramas (10 segundos) no nível 1 do perfil *Core*, com um débito binário de 384 kbit/s. Para determinar a influência da escolha do nível no tempo gasto pelas funções durante o processo de decodificação,

esta sequência foi ainda codificada no nível 2 do perfil *Core* com um débito binário de 2000 kbit/s.

- **Stefan** – Sequência em formato CIF constituída por dois objectos de forma arbitrária: o fundo e o jogador (Figura 40d). A sequência/objectos tem 30 tramas/VOPs por segundo. Foram codificadas 300 tramas (10 segundos) no nível 2 do perfil *Core*, com um débito binário de 2000 kbit/s.



a)



b)



c)



d)

Figura 40 – Imagens das sequências de teste: a) Akiyo; b) News; c) Coastguard; d) Stefan.

Para garantir variedade em termos da actividade contida na cena, foram seleccionadas duas sequências com pouco movimento, *Akiyo* e *News*, e duas sequências com mais movimento, *Coastguard* e *Stefan*.

Para as simulações foi utilizado um computador pessoal equipado com um processador Intel Pentium II a 233 MHz e 128 MBytes de memória RAM. Se for utilizado outro computador, os tempos obtidos serão diferentes, mas a hierarquia das funções mais morosas deverá manter-se.

4.3.2 Desempenho do decodificador inicial

Através de uma análise detalhada do desempenho do decodificador original, foram identificadas as funções que gastam mais tempo no decorrer do processo de decodificação. Esta análise foi efectuada recorrendo à aplicação *profiler*, disponível no compilador Microsoft Visual C++ 6.0, que indica o tempo e a percentagem do tempo total gasto por cada função no decorrer do programa.

4.3.2.1 Descrição das principais funções do decodificador

O código do codec de vídeo MoMuSys/IST inclui centenas de funções. Antes de se avaliar onde gasta o decodificador mais tempo, apresenta-se uma descrição das funções mais importantes e que por serem aquelas onde o decodificador gasta mais tempo vão aparecer em várias tabelas ao longo deste capítulo.

- **AddImageI** – Adiciona duas imagens constituídas por valores inteiros, *pixel a pixel*; é utilizada para somar um VOP a outro depois de efectuada a compensação de movimento.
- **BitstreamReadBits** – Lê n bits de um ficheiro e avança o ponteiro de leitura.
- **BitstreamShowBits** – Permite “ver” n bits de um ficheiro sem avançar o ponteiro de leitura.
- **Blend** – Faz a composição de dois VOPs ao nível do *pixel*; esta função é utilizada para fazer a composição de vários objectos numa cena.
- **Block_IDCT** – Calcula a transformada DCT inversa para um bloco de dimensão 8×8 *pixels*.
- **BlockDequantH263** – Faz a quantificação inversa dos coeficientes DCT de um bloco de dimensão 8×8 *pixels*.
- **ClipImageI** – Limita o valor dos *pixels* de uma imagem, por exemplo, 255 como valor máximo; é utilizada para repor o nível máximo e mínimo dos *pixels* após a adição de um VOP a outro VOP depois de aplicada a compensação de movimento.
- **CopyImageI** – Copia imagens (VOPs) de uma fonte para um destino.
- **DecodeVol** – Descodifica uma camada de um objecto de vídeo.
- **DecodeVopCombinedMotionShapeTextureInter** – Descodifica um VOP Inter; para todos os macroblocos que constituem o VOP, descodifica a informação de forma, os vectores de movimento e a textura.
- **ecalloc** – Reserva uma porção de memória com tratamento de erros, ou seja verifica se o espaço de memória pretendido foi efectivamente reservado.
- **FreeImage** – Liberta toda a memória associada a uma imagem.
- **GetMBblockdata** – Descodifica a textura de um macrobloco, ou seja descodifica os coeficientes DCT que o compõem; para isso descodifica os quatro blocos de luminância e os dois blocos de crominância recorrendo à transformada DCT inversa (formato 4:2:0).
- **GetPred_Advanced** – Efectua a compensação de movimento para um bloco de dimensão 8×8 pertencente a um dado macrobloco.
- **GetPred_Chroma** – Faz a predição da crominância com compensação de movimento.
- **InterpolateImage** – Interpola uma imagem inteira para facilitar a predição com resolução de $\frac{1}{2}$ *pixel*.

- LoadArea – Faz uma cópia de parte de uma imagem; esta função é utilizada no processo de *padding*.
- MakeImageEdge – Cria uma imagem com uma moldura cuja largura é um macrobloco (16×16 *pixels*). Se a imagem original tem dimensões $W \times H$, a nova imagem terá dimensões $(W+2 \times 16) \times (H+2 \times 16)$; esta função é utilizada para aplicar o *padding* aos macroblocos que se encontram fora da *bounding box* do objecto, imediatamente junto à fronteira.
- PutSubImage – Extrai uma parte de uma imagem, ou seja copia parte de uma imagem para outra mais pequena.
- SetConstantImageI – Atribui a todos os *pixels* de uma imagem um dado valor constante.
- ShapeDecodingCAE – Descodifica um BAB (*Binary Alpha Block*) utilizando *IntraCAE* (*Context-based Arithmetic Encoding* no modo *Intra*).
- subsamp_alpha – Gera uma matriz com a informação de forma sub-amostrada em unidades de bloco ou macrobloco; a matriz é utilizada para verificar o tipo de forma de um macrobloco, que pode ser transparente, opaca ou de fronteira.
- SubsampleAlphaMap – Sub-amostra por um factor de 2 a forma de um objecto nas direcções horizontal e vertical; esta função é necessária para fazer a composição das componentes U e V de um VOP, já que estas têm metade da resolução espacial da componente Y em cada direcção (formato 4:2:0).
- unrestricted_MC – Função auxiliar utilizada na compensação de movimento sem restrições (*Unrestricted Motion Compensation*); devolve a posição do *pixel* na fronteira da *bounding box* do VOP.
- unrestricted_MC_chro – Função auxiliar utilizada na compensação de movimento sem restrições (*Unrestricted Motion Compensation*) para a cromaância; devolve a posição do *pixel* na fronteira da *bounding box* do VOP.
- VlcDecTCOEF – Descodifica um coeficiente DCT codificado com VLC (*Variable Length Coding*).
- WriteVopRaw – “Escreve” um VOP para o disco; consiste na escrita dos *pixels* da textura para um ficheiro e dos *pixels* da forma para outro ficheiro.

São estas funções que serão de seguida avaliadas em termos de tempo gasto durante o processo de descodificação.

4.3.2.2 Tempo gasto pelas principais funções de descodificação

Nas tabelas seguintes mostra-se o tempo e a percentagem do tempo total gasto por cada função para a descodificação dos fluxos binários resultantes da codificação das sequências descritas em 4.3.1. Para a presente análise só se mostram as funções que gastam mais tempo, já que todas as outras funções ocupam o processador durante percentagens de tempo muito pequenas e que juntamente com as funções anteriores perfazem a totalidade do tempo gasto.

		<i>Akiyo</i> (300 tramas)
Função	% de tempo gasto	Tempo gasto (s)
WriteVopRAW	37.1	18.7
Block_IDCT	10.9	5.5
GetPred_Advanced	6.4	3.2
InterpolateImage	5.4	2.7
Blend	4.0	2.0
DecodeVopCombinedMotionShapeTextureInter	2.9	1.5
Unrestricted_MC	2.8	1.4
SetConstantImageI	2.7	1.4
AddImageI	2.4	1.2
CopyImageI	2.3	1.2
ClipImageI	1.8	0.9
MakeImageEdge	1.7	0.9
Ecalloc	1.2	0.6
GetPred_Chroma	1.1	0.5
SubsampleAlphaMap	1.0	0.5
Tempo total	87.7%	50.3 s

Tabela 4 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Akiyo para Simple@L2.

		<i>News</i> (300 tramas)
Função	% de tempo gasto	Tempo gasto (s)
WriteVopRAW	39.5	68.5
BlockIDCT	11.2	19.4
GetPred_Advanced	6.2	10.7
InterpolateImage	5.0	8.6
Blend	4.9	8.5
SetConstantImageI	3.4	5.9
CopyImageI	3.3	5.7
AddImageI	2.9	5.0
DecodeVopCombinedMotionShapeTextureInter	2.6	4.5
ClipImageI	2.5	4.4
MakeImageEdge	2.4	4.2
unrestricted_MC	2.0	3.4
GetPred_Chroma	1.2	2.1
SubsampleAlphaMap	1.0	1.7
GetMBblockdata	0.9	1.5
Tempo total	89.0%	173.6 s

Tabela 5 – Percentagem de tempo gasto para as principais funções de descodificação: sequência News para Simple@L3.

		<i>Coastguard</i> (300 tramas)
Função	% de tempo gasto	Tempo gasto (s)
WriteVopRAW	34.3	45.9
BlockIDCT	12.7	17.0
Blend	4.4	5.9
InterpolateImage	3.7	4.9
subsamp_alpha	2.7	3.6
ecalloc	2.2	3.0
GetPred_Advanced	2.1	2.8
SetConstantImageI	1.6	2.1
DecodeVopCombinedMotionShapeTextureInter	1.5	2.0
AddImageI	1.3	1.8
CopyImageI	1.2	1.7
GetPred_Chroma	1.2	1.6
FreeImage	1.1	1.5
GetMBblockdata	1.1	1.5
LoadArea	1.0	1.4
Tempo total	72.1%	133.8 s

Tabela 6 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Coastguard para Core@L1.

		<i>Stefan</i> (300 tramas)
Função	% de tempo gasto	Tempo gasto (s)
WriteVopRAW	30.9	103.1
BlockIDCT	16.3	54.5
Blend	4.8	15.9
subsamp_alpha	3.8	12.7
InterpolateImage	3.1	10.3
GetPred_Advanced	2.4	7.9
CopyImageI	2.0	6.6
SetConstantImageI	1.9	6.5
AddImageI	1.7	5.7
GetMBblockdata	1.5	4.9
DecodeVopCombinedMotionShapeTextureInter	1.4	4.8
ClipImageI	1.4	4.7
MakeImageEdge	1.4	4.7
GetPred_Chroma	1.3	4.4
ShapeDecodingCAE	1.1	3.8
Tempo total	75.0%	333.8 s

Tabela 7 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Stefan para Core@L2.

A Tabela 8 mostra a média e a variância da percentagem de tempo gasto pelas principais funções na descodificação das sequências de teste.

Funções	Média (%)	Variância (%)
WriteVopRAW	35.5	10.3
Block_IDCT	12.8	4.6
GetPred_Advanced	4.3	4.1
InterpolateImage	4.3	0.9
Blend	4.5	0.1
DecodeVopCombinedMotionShapeTextureInter	2.1	0.4
Unrestricted_MC	1.7	0.6
SetConstantImageI	2.4	0.5
AddImageI	2.1	0.4
CopyImageI	2.2	0.6
ClipImageI	1.7	0.3
MakeImageEdge	1.6	0.3
Ecalloc	1.2	0.4
GetPred_Chroma	1.2	0.0
SubsampleAlphaMap	0.9	0.0
GetMBblockdata	1.1	0.1
Subsamp_alpha	3.3	0.3
FreeImage	0.7	0.1
LoadArea	0.8	0.1
ShapeDecodingCAE	1.0	0.0

Tabela 8 – Média e variância da percentagem de tempo gasto nas principais funções de descodificação das sequências de teste.

Como se pode comprovar nas tabelas anteriores, a maior parte do tempo gasto no processo de descodificação corresponde à escrita das imagens no disco rígido e ao cálculo da transformada DCT inversa (IDCT). As restantes funções estão relacionadas com operações que implicam uma grande carga computacional, por exemplo a função `Blend` que combina dois VOPs *pixel a pixel*, ou são chamadas muitas vezes, como é o caso da função `GetPred_Advanced` que efectua a compensação de movimento.

Neste caso, optou-se por tentar optimizar o descodificador optimizando as funções que gastam mais tempo, quer porque não estão convenientemente optimizadas, quer porque sendo chamadas muitas vezes qualquer pequeno ganho de optimização se pode transformar num grande ganho no conjunto. Note-se que uma função pode gastar muito tempo mesmo que esteja já bem optimizada pelo simples facto de que é uma função bastante complexa ou é usada muitas vezes.

Face à situação acima descrita, decidiu-se proceder à optimização do descodificador através de:

- Optimização da escrita em disco;
- Substituição da função que efectua o cálculo da IDCT por uma versão mais eficiente;
- Optimização de algumas funções mais pesadas computacionalmente e que podem ser melhoradas, como se descreve na secção 4.3.3.

4.3.2.3 Influência do perfil no desempenho do decodificador

Para determinar a influência da escolha do perfil na hierarquia das funções que gastam mais tempo de decodificação, foram codificadas duas sequências retangulares também com o perfil *Core* (já tinham sido codificadas com o perfil *Simple*). Foi mantido o débito binário relativamente às sequências codificadas no perfil *Simple* para que se possa estudar exclusivamente a escolha do perfil, que se reflecte no conjunto de ferramentas utilizadas, sem interferência da quantidade de recursos disponíveis em termos de débito.

A Tabela 9 mostra a percentagem de tempo gasto pelas principais funções na decodificação da sequência Akiyo no nível 1 do perfil *Core*, utilizando o mesmo débito binário da sequência codificada no perfil *Simple*, ou seja 128 kbit/s.

Função	Akiyo (300 tramas)			
	% de tempo gasto		Tempo gasto (s)	
	Simple@L2	Core@L1	Simple@L2	Core@L1
WriteVopRAW	37.1	36.5	18.7	17.9
BlockIDCT	10.9	11.1	5.5	5.4
GetPred_Advanced	6.4	6.9	3.2	3.4
InterpolateImage	5.4	5.3	2.7	2.6
Blend	4.0	4.0	2.0	2.0
unrestricted_MC	2.8	3.8	1.4	1.9
SetConstantImageI	2.7	2.7	1.4	1.3
DecodeVopCombinedMotionShape TextureInter	2.9	2.5	1.5	1.2
AddImageI	2.4	2.4	1.2	1.2
CopyImageI	2.3	2.3	1.2	1.1
ClipImageI	1.8	1.7	0.9	0.8
MakeImageEdge	1.7	1.7	0.9	0.8
GetPred_Chroma	1.1	1.6	0.5	0.8
Ecalloc	1.2	1.3	0.6	0.6
DecodeVol	0.6	1.2	0.3	0.9
Tempo total	83.3%	85.0%	50.3 s	49.0 s

Tabela 9 – Percentagem de tempo gasto para as principais funções de decodificação: sequência Akiyo para Simple@L2 e Core@L1 a 128 kbit/s.

Da Tabela 9 pode comprovar-se que as funções que gastam mais tempo de processamento são as mesmas, mantendo-se a sua hierarquia e percentagens de tempo de decodificação muito semelhantes. A escrita em disco domina o tempo de decodificação, e não muda com a alteração do perfil de codificação, uma vez que não depende desta escolha já que é sempre o mesmo número de tramas e com a mesma dimensão que são escritas para o disco. O cálculo da transformada DCT inversa gasta o mesmo tempo nos perfis *Simple* e *Core*, uma vez que se mantém o número de coeficientes DCT a decodificar já que o débito binário usado foi igual para ambos os perfis.

Na Tabela 10 estão indicados os tempos gastos na decodificação da sequência *News* nos perfis *Simple@L3* e *Core@L2* com um débito binário de 384 kbit/s.

Função	News (300 tramas)			
	% de tempo gasto		Tempo gasto (s)	
	Simple@L3	Core@L2	Simple@L3	Core@L2
WriteVopRAW	39.5	40.2	68.5	69.4
BlockIDCT	11.2	11.3	19.4	19.5
GetPred_Advanced	6.2	6.1	10.7	10.6
InterpolateImage	5.0	5.0	8.6	8.6
Blend	4.9	4.9	8.5	8.5
SetConstantImageI	3.4	3.4	5.9	5.9
CopyImageI	3.3	3.3	5.7	5.7
AddImageI	2.9	2.9	5.0	5.0
DecodeVopCombinedMotionShape TextureInter	2.6	2.6	4.5	4.5
ClipImageI	2.5	2.5	4.4	4.4
MakeImageEdge	2.4	2.4	4.2	4.2
unrestricted_MC	2.0	2.1	3.4	3.7
GetPred_Chroma	1.2	1.1	2.1	1.8
SubsampleAlphaMap	1.0	1.0	1.7	1.6
GetMBblockdata	0.9	0.9	1.5	1.6
Tempo total	89.0%	89.7%	173.6 s	172.7 s

Tabela 10 – Percentagem de tempo gasto para as principais funções de descodificação: sequência News para Simple@L3 e Core@L2 a 384 kbit/s.

As duas funções que dominam claramente o tempo de descodificação são WriteVopRAW e BlockIDCT. Tal como no caso anterior, a hierarquia das funções bem como os respectivos tempos de descodificação mantêm-se, pois foi utilizado o mesmo débito binário para ambos os casos.

Os dois casos analisados mostram que a variação do perfil escolhido para codificar uma cena não tem influência nas funções que gastam mais tempo no processo de descodificação.

4.3.2.4 Influência do nível no desempenho do decodificador

A Tabela 11 mostra a percentagem de tempo gasto pelas principais funções na descodificação da sequência Akiyo codificada no nível 3 do perfil Simple, com um débito binário de 384 kbit/s; esta sequência já tinha sido codificada no nível 2 do perfil Simple, com um débito binário de 128 kbit/s.

Função			Akiyo (300 tramas)	
	% de tempo gasto		Tempo gasto (s)	
	Simple@L2	Simple@L3	Simple@L2	Simple@L3
WriteVopRaw	37.1	33.6	18.7	19.0
Block_IDCT	10.9	18.8	5.5	10.6
InterpolateImage	5.4	4.3	2.7	2.4
Blend	4.0	3.5	2.0	2.0
GetPred_Advanced	6.4	3.4	3.2	2.0
DecodeVopCombinedMotionShape TextureInter	2.9	3.1	1.5	1.7
SetConstantImageI	2.7	2.5	1.4	1.4
AddImageI	2.4	2.1	1.2	1.2
CopyImageI	2.3	2.0	1.2	1.1
GetMBblockdata	0.9	1.6	0.5	0.9
MakeImageEdge	1.7	1.5	0.9	0.9
ClipImageI	1.8	1.5	0.9	0.8
Ecalloc	1.2	1.4	0.6	0.8
BlockDequantH263	0.7	1.3	0.3	0.7
DecodeVol	0.6	1.0	0.3	0.6
Tempo total	81.0%	81.6%	50.3 s	56.5 s

Tabela 11 – Percentagem de tempo gasto para as principais funções de decodificação: sequência Akiyo para Simple@L2 e Simple@L3 a 128 e 384 kbit/s.

Da Tabela 11 pode-se constatar que as funções que gastam mais tempo de processamento são as mesmas, independentemente do nível utilizado. As funções `WriteVopRaw` e `Block_IDCT` continuam claramente a ser as que gastam mais tempo. A função `WriteVopRaw` gasta quase exactamente o mesmo tempo para os dois níveis, já que a escrita para o disco não depende do nível escolhido nem do débito binário de codificação mas apenas da resolução espacial e duração da sequência. A função `Block_IDCT` gasta mais tempo no nível 3 porque neste nível o débito binário é maior, e como consequência existem mais coeficientes DCT para decodificar em cada bloco.

Na Tabela 12 mostra-se a percentagem de tempo gasto pelas principais funções na decodificação da sequência *Coastguard* codificada no nível 2 do perfil *Core*, com um débito binário de 2000 kbit/s; esta sequência já tinha sido codificada no nível 1 do perfil *Core*, com um débito binário de 384 kbit/s.

Função	<i>Coastguard</i> (300 tramas)			
	% de tempo gasto		Tempo gasto (s)	
	Core@L1	Core@L2	Core@L1	Core@L2
WriteVopRAW	34.3	31.8	45.9	45.5
BlockIDCT	12.7	12.7	17.0	18.2
Blend	4.4	4.2	5.9	6.0
InterpolateImage	3.7	3.5	4.9	5.0
Subsamp_alpha	2.7	2.5	3.6	3.5
GetPred_Advanced	2.1	2.1	2.8	2.9
Ecalloc	2.2	2.0	3.0	2.9
VlcDecTCOEF	0.4	1.9	0.6	2.8
DecodeVopCombinedMotionShape TextureInter	1.5	1.5	2.0	2.2
BitstreamShowBits	0.5	1.5	0.6	2.2
SetConstantImageI	1.6	1.5	2.1	2.1
BitstreamReadBits	0.5	1.4	0.7	2.0
GetMBblockdata	1.1	1.3	1.5	1.8
AddImageI	1.3	1.2	1.8	1.7
CopyImageI	1.2	1.2	1.7	1.7
Tempo total	70.2%	70.3%	133.8 s	143.1 s

Tabela 12 – Percentagem de tempo gasto para as principais funções de descodificação: sequência Coastguard para Core@L1 e Core@L2 a 384 e 2000 kbit/s.

Tal como no caso anterior, as funções que consomem mais tempo são as mesmas para ambos os níveis. Apesar de a função `Block_IDCT` gastar a mesma percentagem de tempo para ambos os níveis, em termos absolutos gasta mais tempo no nível 2 porque neste nível o débito binário é maior, e existem mais coeficientes DCT para descodificar em cada bloco. A diferença de tempo entre os dois níveis é aqui muito menor porque como se usam débitos mais elevados, o aumento do débito já não serve tanto para aumentar o número de coeficientes DCT codificados mas mais a precisão (qualidade da quantificação) desses coeficientes.

Pode-se concluir dos resultados anteriores que o nível de codificação não tem qualquer influência nas funções que gastam mais tempo de descodificação ainda que possa alterar as percentagens respectivas.

4.3.3 Optimização de funções seleccionadas

Com o objectivo de acelerar o processo de descodificação, o código do descodificador original foi melhorado, embora a estrutura global tenha sido mantida. Depois de identificadas as funções onde o processador gastava mais tempo, referidas anteriormente, foram reescritas algumas delas, nomeadamente aquelas onde era claramente possível optimizar o código de modo a tornar o processo de descodificação mais rápido. As alterações mais significativas efectuadas foram:

- **Escrita no disco** – A escrita dos VOPs ou tramas para o disco é uma operação utilizada frequentemente, já que neste descodificador todos os VOPs/tramas descodificados para cada objecto e também para a cena composta (constituída por todos os objectos de vídeo

compostos) são escritos em disco. A escrita era originalmente feita *pixel a pixel* e foi substituída por uma escrita VOP a VOP, que é efectuada para um ficheiro com a textura e para um ficheiro com a forma do objecto, o que acelera significativamente o processo. As alterações efectuadas permitiram obter um ganho de, aproximadamente, três vezes em termos do tempo gasto nesta função para as sequências testadas. A escrita em disco de cada VOP só é feita no final da descodificação de todos os macroblocos que o constituem, e não será contabilizada para o tempo de descodificação de um macrobloco o que é razoável uma vez que a escrita em disco não é parte integrante do processo de descodificação e da complexidade que lhe está associada.

- **Cálculo da IDCT** – Foi integrado no decodificador um novo algoritmo para o cálculo da IDCT em blocos de dimensão 8×8 , idêntico ao especificado pela norma JPEG [23]. Este algoritmo utiliza a mesma abordagem do algoritmo original, ou seja calcula a transformada 2D através da aplicação da transformada 1D a cada linha, seguida da aplicação da mesma transformada a cada coluna. Contudo, o algoritmo original utilizava aritmética de vírgula flutuante, o que associado a um elevado número de multiplicações tornava a implementação da IDCT bastante lenta. O novo algoritmo adoptado é baseado no algoritmo descrito em [24], e utiliza somente 12 multiplicações e 32 adições por cada passagem numa linha ou coluna do bloco 8×8 , enquanto que o algoritmo original executava 16 multiplicações e 26 adições. Apesar de o novo algoritmo efectuar mais 6 adições que o algoritmo original, efectua menos 4 multiplicações, que são bastante mais pesadas computacionalmente que as adições, obtendo-se assim um ganho substancial. A vantagem deste método reside em que nenhum caminho de dados, ou seja nenhum cálculo de uma variável, contém mais do que uma multiplicação, o que permite uma implementação simples e precisa com aritmética de vírgula fixa. Deste modo, consegue-se um aumento da rapidez nesta função de, aproximadamente, nove vezes em relação ao algoritmo original para as sequências testadas. No entanto, o código que efectua o cálculo da IDCT para a norma JPEG [23] apresenta alguns problemas de precisão quando aplicado a sequências de imagens, pois pode verificar-se a acumulação de erros que se propagam de imagem para imagem. Para impedir que estes erros ocorram, foram efectuadas algumas alterações ao código para a norma JPEG, nomeadamente através do aumento do número de casas decimais utilizadas, visando aumentar a precisão dos cálculos.
- **Ciclos de acesso aos *pixels*** – Nas funções que acedem a *pixels*, foram substituídos, sempre que possível, os morosos ciclos de acesso a *pixels* da imagem através da sua posição na linha e na coluna por ponteiros que são incrementados para endereçar esses *pixels*. A utilização de ponteiros acelera consideravelmente as funções pois evita multiplicações desnecessárias para aceder ao conteúdo da imagem. A Figura 41 mostra o pseudo-código em C das alterações referidas, para imagens bidimensionais e imagens armazenadas em vectores unidimensionais.

<pre> ORIGINAL: For (i=0;i<16;i++) { for(j=0;j<16;j++) { Image[i][j] = pixel; } } OPTIMIZADO: p = &(Image[0][0]); pf = p + 256; while (p != pf) { p = pixel_value; p++; } </pre>	<pre> ORIGINAL: for (m = 0; m < 16; m++) { for (n = 0; n < 16; n++) { Image[m*8+n]=pixel; } } OPTIMIZADO: p = Image; pf = p + 256; while (p != pf) { p = pixel_value; p++; } </pre>
---	--

a)

b)

Figura 41 – Alterações efectuadas para otimizar o acesso: a) imagens bidimensionais e b) imagens armazenadas em vectores.

- **Inicialização de VOPs e matrizes** – Outra operação onde se gastava bastante tempo no decodificador original era a colocação de todos os *pixels* de um VOP ou matriz (que pode guardar blocos, macroblocos, vectores de movimento, etc.) a zero através de um ciclo, onde só uma posição da matriz é inicializada por cada iteração. Esta função destina-se principalmente a inicializar zonas de memória onde se vão guardar imagens resultantes de outras operações. É utilizada, por exemplo, na decodificação de VOPs *Inter*. Depois de decodificadas as diferenças e efectuada a compensação de movimento, é necessário adicionar o resultado ao VOP anterior. Se o VOP foi inicialmente colocado a 0, os macroblocos que não foram codificados são automaticamente substituídos por macroblocos do VOP anterior. A solução inicial foi substituída pela instrução *memset* que realiza a mesma operação mas é muito mais rápida, pois trabalha sobre blocos contíguos de memória e inicializa várias posições de cada vez.
- **Libertação eficiente de memória** – Foram identificadas algumas funções onde a memória não era gerida de forma eficiente, nomeadamente devido a blocos de memória que eram reservados mas não eram libertados, tendo estas situações sido eliminadas. A reserva de memória que não é libertada pode originar paginação, que é um processo que ocorre quando a memória física é excedida e é substituída pelo disco rígido, o que também contribui para diminuir a rapidez da decodificação. Adicionalmente, existiam funções no decodificador original onde eram efectuadas operações de reserva e libertação de memória desnecessárias, que contribuíam com algum tempo adicional para a decodificação, tendo esta situação sido também resolvida.
- **Utilização da estrutura *Macroblock*** – Através da análise detalhada da memória utilizada pelo decodificador, observando para as sequências de teste qual o tipo de estruturas utilizadas e a sua evolução ao longo do tempo, verificou-se que a estrutura *Macroblock*,

que serve para guardar os valores da luminância e das crominâncias dos *pixels* presentes num macrobloco, era reservada e libertada centenas de vezes pelas várias funções, só sendo, no entanto, necessária uma estrutura de cada vez. Esta situação foi alterada passando a utilizar-se uma estrutura estática que só é reservada uma vez. Assim, é sempre a mesma estrutura que é usada sempre que necessário, evitando-se assim a repetição sucessiva da reserva e posterior libertação de memória.

4.3.4 Sistema de gestão de memória

A reserva e a libertação de memória deve ser feita de modo eficiente quando se pretende medir o tempo de decodificação dos macroblocos de forma fiável. Este processo é realizado milhares de vezes, dependendo este número da dimensão e da quantidade de objectos de vídeo presentes na cena. A ser realizado de forma demasiado ineficiente, este processo pode falsear os resultados que se pretendem alcançar, com a maior fiabilidade possível, em termos do tempo de decodificação dos vários tipos de macroblocos. Isto acontece porque a reserva e libertação de memória para variáveis auxiliares pode originar um incremento do tempo de decodificação que não resulta directamente da complexidade efectiva de decodificação do macrobloco. Para melhorar este aspecto, foi desenvolvido um novo sistema de gestão de memória que substitui o sistema operativo na morosa tarefa de reserva e libertação de memória tornando todo o processo mais rápido, o que permite medir com maior fiabilidade o tempo de decodificação dos vários tipos de macroblocos e logo obter uma medida mais credível da complexidade de decodificação dos vários tipos de macroblocos.

O sistema alternativo de gestão de memória proposto baseia-se na reserva inicial de porções de memória constituídas por blocos, que são atribuídos ao decodificador à medida das suas necessidades. Deste modo, evita-se a constante reserva e libertação de memória por parte do sistema operativo, sendo este processo substituído pela atribuição de memória que foi previamente reservada.

Em termos globais, este sistema não melhora o tempo global de decodificação de uma sequência, já que vai ser dispendido bastante tempo no início da decodificação para reservar a memória necessária ao funcionamento do sistema de gestão de memória. As vantagens deste sistema reflectem-se ao nível do macrobloco, pois a reserva e libertação de memória são feitas de um modo mais rápido influenciando menos o tempo de decodificação de cada macrobloco.

O sistema de gestão de memória proposto e implementado baseia-se nos seguintes passos:

1. Definição do número e tipo de blocos de memória a reservar

No início da decodificação são reservados conjuntos de blocos de memória de dimensão fixa. A dimensão dos blocos a utilizar, a sua quantidade e o número de classes (com diferentes dimensões) de blocos que se utilizam no sistema de gestão de memória têm influência no desempenho do decodificador, como se explica de seguida.

- **Dimensão dos blocos** – A dimensão dos blocos utilizados pelo decodificador está directamente relacionada com a dimensão espacial dos objectos presentes na cena. Foram implementados blocos cujas dimensões são potências de dois (medidas em

bytes) e que podem ser utilizados com qualquer tipo de sequência. Os blocos de dimensão mais pequena são os mais utilizados e servem para guardar uma grande variedade de estruturas, que podem ser blocos, macroblocos, matrizes com vectores de movimento, matrizes com passos de quantificação, etc. Os blocos maiores são geralmente utilizados menos vezes e destinam-se principalmente a guardar VOPs completos e as respectivas predições.

- **Número de classes de blocos** – O número de classes de blocos tem de ser ajustado às necessidades de memória do decodificador. Após numerosos testes para observação da utilização de memória por parte do decodificador, com várias dimensões de objectos e de cenas, optou-se pela implementação de sete tamanhos diferentes de blocos, cada um deles com a dimensão indicada na Tabela 13. Os blocos do tipo *short int* são utilizados para matrizes que guardam *pixels*, que são números inteiros, enquanto que os blocos do tipo *float* são utilizados para os vectores de movimento, cujos valores podem ser fraccionários. Um número pequeno de classes faz com que haja um grande desperdício de memória, pois poderão mais facilmente existir blocos grandes a armazenar pouca informação. Por outro lado, um número grande de classes desperdiça menos memória mas torna a inicialização mais lenta, porque é feita para mais classes. Também torna a implementação mais complexa pois têm de existir mais vectores de ponteiros para os blocos de memória.

	Tipo	Dimensão dos Blocos (bytes)	Número Inicial de Blocos
A	<i>short int</i>	8192	100
B	<i>short int</i>	65536	25
C	<i>short int</i>	262144	20
D	<i>short int</i>	524288	8
E	<i>short int</i>	1048576	4
F	<i>short int</i>	2097152	1
G	<i>float</i>	2048	4

Tabela 13 – Tipos de blocos implementados no sistema de gestão de memória.

- **Número de blocos** – A quantidade de blocos de cada tipo que são reservados de cada vez tem bastante importância no desempenho do decodificador. Se são inicialmente reservados poucos blocos, é necessário repetir a operação de reserva muitas vezes. Por outro lado, se forem inicialmente reservados muitos blocos é perdido muito tempo nesta operação e podem sobrar blocos que não são utilizados. Em qualquer dos casos, existe prejuízo para o desempenho do decodificador. O objectivo principal do sistema de gestão de memória no contexto desta tese é permitir medir com maior fiabilidade o tempo de decodificação de um macrobloco, ou seja o tempo total de decodificação é menos importante, já que as medições de tempo se farão no capítulo 5 ao nível do macrobloco. Neste contexto, é preferível adoptar a segunda solução, onde se reservam mais blocos, pois deste modo há uma maior probabilidade de que se disponha da memória suficiente para o processo de decodificação (sem ter de fazer reservas para além da reserva inicial). A Tabela 13 mostra a quantidade de blocos de cada tipo que se propõe reservar no início da decodificação.

2. Definição de vectores de ponteiros

Elementos centrais do processo de descodificação são os vectores de ponteiros que apontam para blocos de memória que foram previamente reservados (Figura 42). Existe um vector de ponteiros para cada classe de blocos implementada, que apontam para blocos que são todos da mesma dimensão (especificada pela classe). Estes vectores de ponteiros guardam a posição dos vários blocos de memória e o seu estado, ou seja se estão ocupados ou livres. A dimensão de cada vector de ponteiros é igual ao número de blocos existentes em cada classe.

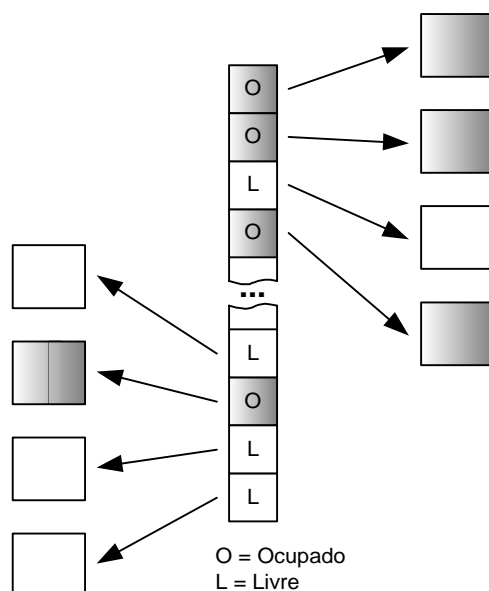


Figura 42 – Funcionamento do vector de ponteiros para o sistema de gestão de memória. O vector de ponteiros está representado ao centro e cada vector aponta para um bloco de memória.

3. Marcação dos blocos como ocupados ou livres

Durante o processo de descodificação, os blocos são marcados como ocupados ou livres sempre que se deseja reservar ou libertar memória, evitando deste modo a constante reserva e libertação de memória por parte do sistema operativo. A memória é inicialmente reservada com o tipo e número de blocos referidos na Tabela 13. Quando é pedido um bloco de memória, é percorrida a lista de blocos da classe correspondente (definida pela dimensão do bloco) até encontrar um livre, que é utilizado e marcado como ocupado. Para libertar a memória, marca-se simplesmente o bloco como livre. Este processo permite medir com maior fiabilidade o tempo de descodificação dos macroblocos, pois substitui as sucessivas reservas e libertações de memória a cargo do sistema operativo pela utilização de blocos que já estão reservados à partida e nunca são, na prática, libertados em termos de sistema operativo.

4. Reserva adicional de blocos

Se o descodificador necessitar de mais memória do que aquela que está reservada à partida, é reservado um novo conjunto de blocos. A quantidade de blocos que é reservada para cada classe é metade da reservada no início da descodificação, tal como descrito na Tabela 13. Este processo atrasa consideravelmente o processo de descodificação pois a reserva de uma quantidade grande de blocos é uma operação demorada.

Para além de reduzir o impacto da reserva e libertação de memória no tempo de descodificação dos macroblocos, outra grande vantagem do sistema de gestão de memória implementado no contexto da presente tese é a de todas as estruturas utilizadas pelo descodificador para o armazenamento de macroblocos passarem a ter o mesmo esquema de atribuição de memória, o que elimina um elemento que poderia indevidamente ser diferenciador em termos dos tempos de descodificação para os vários tipos de macroblocos. A implementação original do descodificador utiliza tanto a estrutura *Macroblock* como blocos de dimensão 16×16 (imagens) para armazenar macroblocos, o que dava origem a pesos diferentes em termos de tempo de descodificação para a mesma tarefa consoante se usava uma solução ou a outra. Isto acontece porque a estrutura do código inicial não é uniforme, devido ao grande número de pessoas que trabalharam no projecto. Com o sistema de gestão de memória proposto, a atribuição e reserva de memória é semelhante para ambos os casos, já que a memória é sempre previamente reservada; assim, este factor não introduz qualquer diferença indevida na complexidade associada aos vários tipos de macroblocos.

4.3.5 Desempenho do descodificador optimizado

Com as alterações descritas na secção 4.3.3, o descodificador ficou mais rápido, mais eficiente e mais bem estruturado. Na Tabela 14 mostram-se os tempos totais de descodificação e as respectivas percentagens para as funções que gastam mais tempo no processo de descodificação, para o descodificador original e para o descodificador optimizado. As funções estão ordenadas por ordem decrescente do tempo gasto no processo de descodificação para o descodificador optimizado.

Função			<i>Akiyo</i> (300 tramas)		
	% de tempo gasto		Tempo gasto (s)		
	Descodificador original	Descodificador optimizado	Descodificador original	Descodificador optimizado	Ganho (%)
WriteVopRaw	37.1	20.5	18.7	6.6	-65%
InterpolateImage	5.4	7.9	2.7	2.5	-7%
Blend	4.0	7.1	2.0	2.3	+15%
GetPred_Advanced	6.4	6.4	3.2	2.1	-34%
DecodeVopCombinedMotionShapeTextureInter	2.9	4.0	1.5	1.3	-13%
AddImageI	2.4	3.8	1.2	1.2	0%
CopyImageI	2.3	3.7	1.2	1.2	0%
SetConstantImageI	2.7	3.5	1.4	1.1	-21%
GetPred_Chroma	1.1	3.1	0.5	1.0	+100%
ClipImageI	1.8	2.8	0.9	0.9	0%
unrestricted_MC_chro	0.5	2.7	0.3	0.9	+200%
MakeImageEdge	1.7	2.7	0.9	0.9	0%
Ecalloc	1.2	1.9	0.6	0.6	0%
BlockIDCT	10.9	1.9	5.5	0.6	-89%
DecodeVol	0.6	1.8	0.3	0.6	+100%
Tempo total	81.0%	73.8%	50.3 s	32.2 s	-36%

Tabela 14 – Percentagem de tempo e tempo total gasto para as principais funções de descodificação: sequência *Akiyo* para Simple@L2.

Função	News (300 tramas)				
	% de tempo gasto		Tempo gasto (s)		
	Decodificador original	Decodificador otimizado	Decodificador original	Decodificador otimizado	Ganho (%)
WriteVopRAW	39.5	16.7	68.5	18.9	-73%
Blend	4.9	8.9	8.5	10.1	+19%
InterpolateImage	5.0	7.9	8.6	8.9	+3%
GetPred_Advanced	6.2	7.6	10.7	8.6	-20%
CopyImageI	3.3	5.2	5.7	5.8	+2%
AddImageI	2.9	4.4	5.0	5.0	0%
SetConstantImageI	3.4	4.2	5.9	4.8	-19%
DecodeVopCombinedMotionShapeTextureInter	2.6	4.2	4.5	4.8	+7%
ClipImageI	2.5	3.7	4.4	4.2	-5%
MakeImageEdge	2.4	3.7	4.2	4.2	0%
GetPred_Chroma	1.2	3.6	2.1	4.1	+95%
unrestricted_MC_chro	0.5	2.5	0.8	2.9	+263%
BlockIDCT	11.2	1.8	19.4	2.0	-90%
SubsampleAlphaMap	1.0	1.4	1.7	1.6	-6%
GetMBblockdata	0.9	1.3	1.5	1.4	-7%
Tempo total	87.5%	77.1%	173.6 s	113.2 s	-35%

Tabela 15 – Percentagem de tempo e tempo total gasto para as principais funções de decodificação: sequência News para Simple@L3.

Função	Coastguard (300 tramas)				
	% de tempo gasto		Tempo gasto (s)		
	Decodificador original	Decodificador otimizado	Decodificador original	Decodificador otimizado	Ganho (%)
WriteVopRAW	34.3	19.0	46.0	15.7	-66%
InterpolateImage	3.7	6.0	5.0	5.0	0%
Blend	4.4	5.9	5.9	4.9	-17%
ecalloc	2.2	3.4	3.0	2.9	-3%
GetPred_Advanced	2.1	3.0	2.8	2.5	-11%
DecodeVopCombinedMotionShapeTextureInter	1.5	2.7	2.0	2.2	+10%
BlockIDCT	12.7	2.3	17.0	1.9	-89%
CopyImageI	1.2	2.0	2.7	1.7	-37%
AddImageI	1.3	2.0	1.8	1.7	-6%
SetConstantImageI	1.6	2.0	2.1	1.7	-19%
FreeImage	1.1	1.8	1.5	1.5	0%
ClipImageI	1.0	1.7	1.4	1.4	0%
LoadArea	1.0	1.7	1.4	1.4	0%
GetMBblockdata	1.1	1.6	1.5	1.4	-7%
MakeImageEdge	0.9	1.4	1.2	1.2	0%
Tempo total	70.1%	56.5%	133.8 s	82.8 s	-38%

Tabela 16 – Percentagem de tempo e tempo total gasto para as principais funções de decodificação: sequência Coastguard para Core@L1.

Função	Stefan (300 tramas)				
	% de tempo gasto		Tempo gasto (s)		
	Decodificador original	Decodificador otimizado	Decodificador original	Decodificador otimizado	Ganho (%)
WriteVopRAW	30.9	14.9	103.1	31.0	-70%
Blend	4.8	5.7	15.9	11.9	-25%
InterpolateImage	3.1	5.0	10.3	10.3	0%
GetPred_Advanced	2.4	4.1	7.9	8.6	+9%
CopyImageI	2.0	3.1	6.6	6.5	-2%
BlockIDCT	16.3	3.1	54.5	6.5	-88%
AddImageI	1.7	2.7	5.7	5.7	0%
SetConstantImageI	1.9	2.5	6.5	5.3	-18%
DecodeVopCombinedMotionShapeTextureInter	1.4	2.5	4.8	5.2	+8%
GetMBblockdata	1.5	2.3	4.9	4.8	-2%
MakeImageEdge	1.4	2.3	4.7	4.8	+2%
ClipImageI	1.4	2.3	4.7	4.8	+2%
GetPred_Chroma	1.3	2.2	4.4	4.5	+2%
ShapeDecodingCAE	1.1	1.9	3.8	3.8	0%
PutSubImage	1.1	1.7	3.5	3.6	+3%
Tempo total	72.3%	56.3%	333.8 s	207.6 s	-38%

Tabela 17 – Percentagem de tempo e tempo total gasto para as principais funções de decodificação: sequência Stefan para Core@L2.

Como se pode observar nas tabelas anteriores, a escrita dos VOPs para o disco continua a representar a maior percentagem de tempo gasto na decodificação, embora em termos absolutos tenha havido um aumento de rapidez nesta função da ordem das três vezes. O cálculo da transformada DCT inversa, que no decodificador original ocupava a segunda posição na lista das funções mais morosas, passou a ocupar muito menos tempo no processo de decodificação. No entanto, entre as funções que ocupam mais tempo de decodificação existem algumas que já estavam originalmente otimizadas, como é o caso da função *InterpolateImage* ou *CopyImageI*, e como tal não são de esperar grandes ganhos através de qualquer optimização adicional destas funções. A função *Blend*, apesar de ter sofrido alguma optimização, nomeadamente a remodelação dos ciclos de acesso aos *pixels* para os tornar mais rápidos, continua a ser uma das funções que ocupa mais tempo de processamento.

A Tabela 18 mostra o tempo total necessário para a decodificação das sequências de teste referidas na secção 4.3.1 pelos decodificadores original e optimizado. Os ganhos em termos de tempo de decodificação total em relação ao decodificador original variam entre 34.8% e 38.1% para todas as sequências testadas.

	Decodificador Original	Decodificador Optimizado	Ganho
<i>Akiyo (Simple@L2)</i>	50.3 s	32.2 s	36%
<i>News (Simple@L3)</i>	173.6 s	113.2 s	35%
<i>Coastguard (Core@L1)</i>	133.8 s	82.8 s	38%
<i>Stefan (Core@L2)</i>	333.8 s	207.6 s	38%

Tabela 18 – Ganhos entre os decodificadores original e optimizado em termos de tempo total de decodificação.

4.4 Conclusões

Fica claro dos resultados apresentados que as alterações efectuadas ao código do decodificador MoMuSys/IST conduzem a uma melhoria significativa no seu desempenho em termos de rapidez. O aumento de rapidez situou-se entre os 35% e 38% para as sequências de teste utilizadas.

O novo esquema de gestão de memória permite minimizar o incremento de tempo na decodificação dos macroblocos originado pela reserva e libertação de memória. Para além disso, é usado um esquema semelhante para as várias estruturas utilizadas para armazenar macroblocos (blocos de dimensão 16×16 e estrutura *Macroblock*), o que é uma vantagem no contexto desta tese. Este facto é particularmente relevante uma vez que se pretende avaliar a complexidade dos vários tipos de macroblocos através do seu tempo de decodificação, pois a carga computacional necessária para a atribuição e libertação de memória é semelhante para ambas as estruturas e não contribui para aumentar indevidamente a complexidade de uns tipos de macroblocos relativamente a outros.

Em conclusão, o decodificador MoMuSys/IST optimizado parece reunir as condições para permitir a medição de modo fiável da complexidade relativa dos vários tipos de macroblocos usados na codificação de vídeo segundo a norma MPEG-4 Visual.

Capítulo 5

Estudo da Complexidade de Descodificação dos Macroblocos de Vídeo MPEG-4

5.1 Introdução

O estudo da complexidade dos vários tipos de macroblocos de vídeo MPEG-4 é fundamental para a especificação e implementação de um novo modelo, mais eficiente, para o VCV (*Video Complexity Verifier*). O modelo actualmente especificado na norma MPEG-4 faz apenas a distinção entre macroblocos de fronteira e todos os restantes macroblocos, ou seja considera implicitamente que todos os macroblocos (incluindo os transparentes), com excepção dos macroblocos de fronteira, têm igual complexidade. Para que se possa implementar um modelo de complexidade mais ajustado às características reais dos vários tipos de macroblocos de codificação de vídeo, é necessário conhecer a complexidade de descodificação efectiva dos vários tipos de macroblocos que podem compor os objectos de vídeo na norma MPEG-4 Visual [3]. Este capítulo descreve o processo de avaliação da complexidade relativa dos vários tipos de macroblocos de vídeo no contexto da norma MPEG-4 Visual. Esta avaliação é feita com base nos tempos de descodificação dos vários tipos de macroblocos, para várias sequências de teste relevantes, usando o descodificador optimizado MoMuSys/IST, já descrito no capítulo 4.

5.2 Tipos de macroblocos

Um objecto de vídeo é definido na norma MPEG-4 Visual através da sua textura e forma. A codificação da textura e da forma baseia-se numa grelha quadrada estruturada em blocos de 16×16 *pixels* de luminância, e correspondentes crominâncias, denominados macroblocos. Um macrobloco que pertença a um objecto de vídeo contém sempre informação de textura e pode conter ou não informação de forma consoante se localiza ou não sobre o contorno do objecto (Figura 43). No caso de objectos de vídeo rectangulares, todos os macroblocos contêm apenas informação de textura, mas se o objecto tiver forma arbitrária, então os seus macroblocos podem conter simultaneamente informação de textura e de forma como está assinalado na Figura 43. Para objectos rectangulares com dimensões que não sejam múltiplas de 16×16 , em termos de luminância, os macroblocos junto da fronteira da *bounding box* que só parcialmente contêm informação útil de textura são preenchidos para que contenham 16×16 *pixels* de textura. A forma como este preenchimento é feito não é normalizada e depende da implementação dos codificadores.

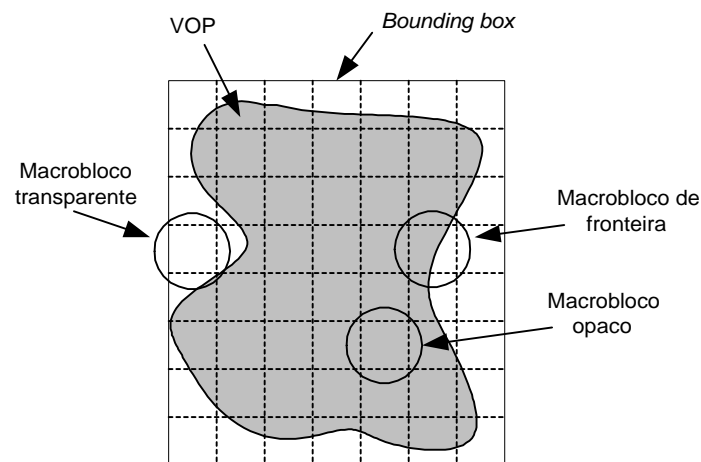


Figura 43 – VOP de forma arbitrária mostrando vários tipos de macroblocos.

5.2.1 Macroblocos com informação de textura

A informação de textura para um dado objecto pode ser codificada de seis modos diferentes, no contexto da norma MPEG-4 Visual [3]. A distinção fundamental baseia-se na exploração ou não da redundância temporal, ou seja na codificação dos valores absolutos dos *pixels* – *codificação Intra* – ou da sua diferença em relação a uma predição temporal – *codificação Inter*. Nos macroblocos do tipo *Intra*, o valor absoluto dos *pixels* é codificado recorrendo à transformada DCT, enquanto que nos macroblocos do tipo *Inter* a transformada DCT é aplicada sobre as diferenças entre o macrobloco a codificar e a sua predição temporal, eventualmente usando vectores de movimento.

Assim, os seis modos de codificação de textura previstos na norma MPEG-4 Visual para todos os *tipos de objecto* de vídeo são [3]:

- **Intra** – Macrobloco codificado independentemente de outros macroblocos, em instantes temporais passados ou futuros.

- **Inter** – Macrobloco codificado diferencialmente, recorrendo ou não à estimação e compensação de movimento (apenas um vector para P-VOPs e dois vectores para B-VOPs), em relação a VOPs passados (P-VOPs) ou VOPs passados e futuros (B-VOPs). Os macroblocos *Inter* P e B correspondem a tipos diferentes, mas nesta tese só se consideram os macroblocos *Inter* do tipo P já que o codificador utilizado não suporta B-VOPs.
- **Intra+Q** – Macrobloco *Intra* onde o passo de quantificação muda relativamente ao macrobloco descodificado imediatamente antes.
- **Inter+Q** – Macrobloco *Inter* onde o passo de quantificação muda relativamente ao macrobloco descodificado imediatamente antes.
- **Inter4V** – Macrobloco *Inter* codificado recorrendo a quatro vectores de movimento, um por cada um dos quatro blocos de luminância (8×8 *pixels*) que o constituem.
- **Skipped** – Macrobloco para o qual nenhuma informação de textura é codificada; neste caso a textura é copiada da posição correspondente no VOP anterior.

A norma MPEG-4 Visual suporta a codificação da textura de macroblocos *Intra* utilizando ou não predição para os coeficientes AC da transformada DCT ao nível do bloco. A predição AC de cada bloco é feita com base no bloco imediatamente à esquerda ou imediatamente acima do bloco considerado. Assim, é de esperar uma diferença de complexidade na descodificação de macroblocos *Intra*, consoante se utilize ou não predição dos coeficientes AC, uma vez que esta ferramenta tem efectivamente associada uma complexidade significativa devido à utilização de outros blocos na predição e ao preenchimento do bloco com o resultado dessa mesma predição. Esta característica não corresponde a tipos de macroblocos diferentes, já que o macrobloco continua a ser do tipo *Intra* pois a predição é efectuada com base em blocos do mesmo VOP e não de VOPs passados ou futuros.

5.2.2 Macroblocos com informação de forma

A informação de forma de um objecto pode ser codificada usando sete modos diferentes, para todos os *tipos de objecto* de vídeo que correspondem a objectos de forma arbitrária. Como se descreveu no capítulo 2, a codificação de forma na norma MPEG-4 Visual faz-se essencialmente recorrendo a uma técnica denominada por *Context-based Arithmetic Encoding* (CAE) [15][16].

Assim, os sete modos de codificação da forma especificados na norma MPEG-4 Visual são [3]:

- **NoUpdate && MVDS == 0** – A informação de forma para o macrobloco corrente é igual à do macrobloco correspondente no VOP anterior, já que não são transmitidas diferenças de vectores de movimento MVDS (*Motion Vector Difference for Shape*).

- **NoUpdate && MVDS != 0** – A informação de forma para o macrobloco corrente é obtida do VOP anterior depois de se aplicar compensação de movimento (é enviado um vector de movimento¹² por macrobloco).
- **Opaco** – Todos os *pixels* de forma (*shapels*) para o macrobloco pertencem ao suporte¹³ do objecto.
- **Transparente** – Nenhum dos *pixels* deste macrobloco da *bounding box* do objecto pertence ao suporte do objecto.
- **IntraCAE** – A forma é codificada utilizando *Context-based Arithmetic Encoding* (CAE) no modo *Intra*, ou seja sem predição temporal.
- **InterCAE && MVDS == 0** – A forma é codificada utilizando *Context-based Arithmetic Encoding* (CAE) no modo *Inter*, sem usar compensação de movimento.
- **InterCAE && MVDS != 0** – A forma é codificada utilizando *Context-based Arithmetic Encoding* (CAE) no modo *Inter*, ou seja com predição temporal, mas neste caso como MVDS != 0 é utilizada compensação de movimento com um vector por macrobloco.

5.2.3 Tipos relevantes de macroblocos

Os tipos de macroblocos cuja complexidade se sabe à partida ser semelhante foram agrupados em classes. É o caso dos macroblocos do tipo *Intra* e *Intra+Q* ou *Inter* e *Inter+Q*, onde a alteração do passo de quantificação não implica um acréscimo significativo de complexidade e dos macroblocos cuja diferença de vectores de movimento (MVDS) é zero e diferente de zero uma vez que para ambos os casos há que obter uma predição a partir da imagem precedente ainda que numa posição diferente. Assim, todos os resultados apresentados ao longo deste capítulo referem-se aos tipos de macroblocos apresentados na Tabela 19; estes tipos de macroblocos são aqueles que se considera valer a pena distinguir por se esperar que a sua complexidade seja razoavelmente diferente.

Informação de textura	Informação de forma
<i>Intra</i> (<i>Intra</i> e <i>Intra+Q</i>)	<i>NoUpdate</i> (MVDS==0 e MVDS!=0)
<i>Inter</i> (<i>Inter</i> e <i>Inter+Q</i>)	Opaco
<i>Inter4V</i>	Transparente
<i>Skipped</i>	<i>IntraCAE</i>
	<i>InterCAE</i> (MVDS==0 e MVDS!=0)

a)
b)

Tabela 19 – Tipos de macroblocos a estudar: a) em termos de informação de textura; b) em termos de informação de forma.

¹² Os vectores de movimento da forma podem ser diferentes dos vectores de movimento da textura.

¹³ O suporte de um objecto de vídeo é definido como a parte da respectiva *bounding box* onde o valor da forma é superior a 0, ou seja onde existem *shapels*.

5.3 Modelos de complexidade dos macroblocos

O modelo VCV permite ao codificador controlar e limitar a capacidade computacional necessária a um decodificador para decodificar os objectos de vídeo que constituem uma cena. No modelo VCV especificado na norma MPEG-4, a carga computacional do decodificador é medida pela ocupação das memórias do modelo VCV, ou seja em termos do número de macroblocos nestas memórias; quanto maior for a sua ocupação, maior é a carga no decodificador.

A complexidade de decodificação dos dados codificados pode ser relacionada com o volume de dados que o processador deve decodificar, ou seja pode ser relacionada com o número de macroblocos por segundo que o decodificador tem de processar. Contudo, o esforço computacional que o decodificador necessita de despender para decodificar um macrobloco pode variar bastante, dada a diversidade de tipos de macroblocos (por exemplo, relativamente ao tipo de forma: opacos, transparentes e de fronteira) e de ferramentas de codificação (por exemplo, em termos de codificação de textura: *Intra*, *Inter* e *Inter4V*), associadas aos vários tipos de objecto, que podem ser utilizadas no processo de codificação. A escolha da medida de complexidade a utilizar depende do grau de fiabilidade que se pretende atingir para a contabilização da complexidade de decodificação. Porém, quanto mais próximo da complexidade real o modelo pretender chegar mais difícil é o modelo ser genérico, pois a complexidade fica altamente dependente de questões de implementação relacionadas com o *software* e *hardware* específicos utilizados.

Existem várias abordagens que podem ser seguidas para modelar a complexidade da cena codificada, que vão da simples medida do número de macroblocos até medidas mais complexas relacionadas com o número de instruções necessárias para decodificar os dados. Estas medidas podem ser agrupadas em quatro classes [25], ilustradas na Figura 44:

- **Número de macroblocos** – Esta abordagem é a mais simples de implementar, já que modela a complexidade de uma cena através do número de macroblocos dos objectos de vídeo que a constituem. Contudo, a utilização unicamente do número de macroblocos para modelar a complexidade de uma cena resulta numa medida fraca pois, como se mostrou no capítulo anterior, os macroblocos podem ser de vários tipos com diferentes complexidades de decodificação associadas.
- **Número de macroblocos para cada tipo de forma do macrobloco** – Esta medida é um pouco mais realista do que a anterior para avaliar a complexidade de uma cena, já que pressupõe que existem tipos de macroblocos que têm complexidades diferentes. Neste caso, a distinção da complexidade dos macroblocos é feita através da sua forma: macroblocos opacos, de fronteira e transparentes. Porém, para cada um destes tipos de forma do macrobloco, a complexidade de decodificação pode ainda variar bastante: por exemplo, os macroblocos de fronteira podem ter a textura (*Intra*, *Inter*, etc.) ou a forma (*IntraCAE*, *InterCAE*, etc.) codificadas de vários modos. A distinção entre a complexidade dos macroblocos usando o tipo de forma (apenas macroblocos fronteira versus todos os macroblocos) é utilizada na norma MPEG-4 Visual.
- **Número de macroblocos para cada tipo de codificação de textura e forma do macrobloco** – Tanto a textura como a forma dos macroblocos podem ser codificadas de muitas maneiras diferentes, tal como referido no capítulo 5. Um modelo de complexidade

mais realista que os anteriores é um modelo que considere as diferenças de complexidade de decodificação inerentes aos vários tipos de codificação de macroblocos existentes, pois é efectivamente daí que advém as diferenças de complexidade de decodificação entre os macroblocos. A avaliação da complexidade em termos do tipo de codificação pode ser feita de três modos:

1. **Codificação da textura** – O modelo distingue a complexidade dos macroblocos através da codificação da sua textura, por exemplo, *Intra*, *Inter* ou *Inter4V*.
 2. **Codificação da forma** – O modelo distingue a complexidade dos macroblocos através da codificação da sua forma, por exemplo, *NoUpdate*, *IntraCAE* ou *InterCAE*.
 3. **Codificação de textura e forma** – O modelo distingue a complexidade dos macroblocos através da codificação da sua textura e forma, por exemplo, *Intra+IntraCAE*, *Inter+NoUpdate* ou *Inter4V+InterCAE*.
- **Número de instruções de decodificação e de operações de escrita/leitura de memória** – Esta abordagem é a que modela de um modo mais realista a complexidade de decodificação de uma cena, pois conta o número de instruções necessárias à decodificação dos macroblocos. Apresenta como grande desvantagem o facto de ser altamente dependente da implementação do decodificador, ou seja o modelo desenvolvido só é válido para o decodificador específico que se estudou.

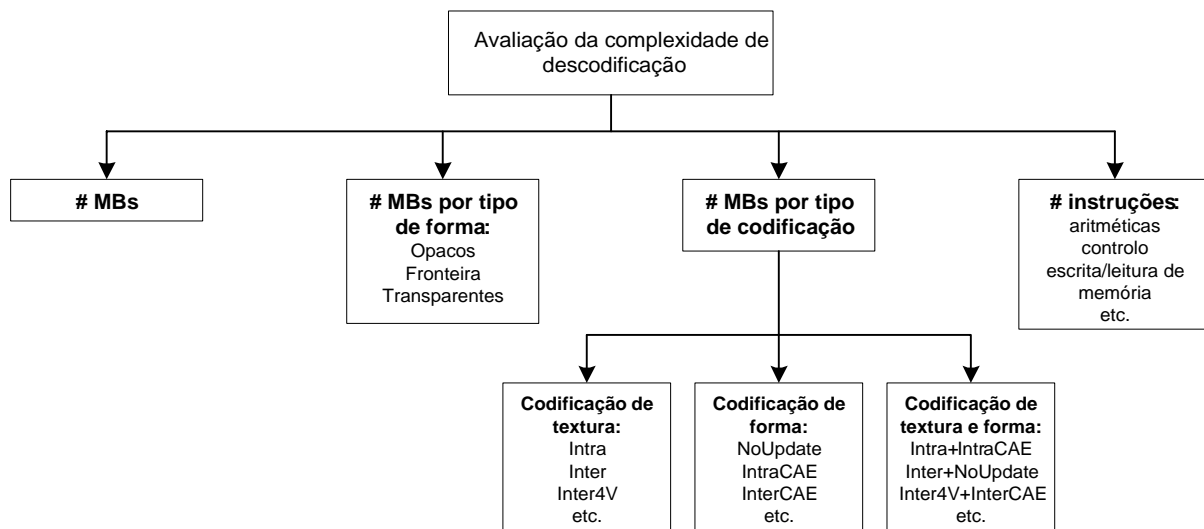


Figura 44 – Estratégias de avaliação da complexidade de decodificação para vídeo MPEG-4.

Em termos da operação do modelo VCV, as abordagens acima são semelhantes. Em cada instante de decodificação de um VOP, a medida de complexidade do VOP é adicionada à memória VCV (#MBs, #instruções, etc.), cuja ocupação diminui a um ritmo constante definido em unidades de complexidade apropriadas pela combinação perfil@nível (#MB/s, #instruções/s, etc.) até ao próximo instante de decodificação.

A ocupação das memórias do modelo VCV fornece um modo para o codificador estimar a carga computacional do decodificador em cada instante. Deste modo é possível controlar e

limitar a capacidade computacional necessária no decodificador, pois o codificador pode utilizar a ocupação das memórias para impedir que a complexidade da cena codificada requeira, para ser decodificada a tempo, uma velocidade de decodificação superior àquela que o decodificador pode atingir.

O modelo de complexidade sugerido nesta tese é baseado no número de macroblocos por tipo de codificação (textura e forma), já que este modelo parece levar em conta os principais factores que determinam as diferenças de complexidade entre macroblocos. Este modelo permite modelar a complexidade de uma cena de vídeo de um modo mais eficiente que o modelo especificado na norma MPEG-4 Visual [3]. Na verdade, este modelo permite explorar as diferenças de complexidade de decodificação entre os vários tipos de macroblocos de um modo preciso sem, no entanto, ir ao nível das instruções utilizadas pelo decodificador, o que só faria sentido caso o decodificador fosse extremamente otimizado. Assim, este capítulo apresenta a avaliação da complexidade baseada no tempo de decodificação dos vários tipos de macroblocos, classificados em termos da codificação da textura e da forma.

5.4 Metodologia para a avaliação da complexidade de decodificação dos macroblocos

Para analisar a complexidade de decodificação dos vários tipos de macroblocos de vídeo é necessário estabelecer um critério de avaliação, ou seja uma medida de complexidade. O critério adoptado nesta tese é o tempo de decodificação de cada macrobloco, já que este deverá determinar directamente o número de macroblocos por segundo que podem ser decodificados, parâmetro usado para dimensionar o VCV [3]. Outro critério que poderia ser utilizado é o número de instruções necessárias para decodificar um macrobloco, mas para além de este parâmetro ser difícil de medir é altamente dependente da implementação do decodificador (o tempo também pode ser, nomeadamente se o decodificador não estiver otimizado). Para que as medidas dos tempos de decodificação obtidas sejam credíveis, é necessário dispor de um decodificador minimamente otimizado e foi por isso que se procedeu ao trabalho de optimização do decodificador, descrito no capítulo 4.

Para se medir com exactidão o tempo de decodificação de um macrobloco, para os vários tipos relevantes de macroblocos, podem ser utilizados dois métodos principais:

- **Descodificação de sequências de vídeo onde todos os macroblocos são do mesmo tipo** – Esta técnica consiste em medir o tempo de decodificação de uma sequência de vídeo onde todos os macroblocos foram codificados com o mesmo tipo por um codificador modificado para o efeito, dividindo-se depois o tempo total de decodificação pelo número total de macroblocos decodificados. Esta solução apresenta a desvantagem de codificar certos macroblocos com modos de codificação que um codificador minimamente “inteligente” nunca usaria, por exemplo, codificar em modo *Intra* macroblocos numa zona extremamente estável da sequência de vídeo. Este facto pode distorcer os valores do tempo de decodificação alcançados em relação aos que seriam os valores de tempo efectivamente típicos de um dado tipo de macrobloco (ou seja quando esse tipo de macrobloco é usado porque é o mais adequado). Quando se utiliza este método para macroblocos de fronteira, tem de se usar uma máscara com a mesma forma para todos os macroblocos de um VOP, para

que todos eles tenham informação de forma semelhante a que corresponda sempre ao mesmo tipo de macrobloco. As máscaras podem variar de VOP para VOP se se pretender codificar a forma no modo *InterCAE*, pois caso não variem a forma é codificada como *NoUpdate*. A máscara a utilizar deve ser crítica em termos de complexidade para que se obtenham os valores máximos de complexidade para o tipo de macrobloco considerado.

- **Codificação de sequências de vídeo utilizando os tipos de macroblocos mais adequados em cada momento** – Este método consiste em recorrer a objectos de vídeo codificados de um modo “normal”, ou seja utilizando um codificador que escolhe para cada macrobloco a codificar o melhor modo de o fazer, e medir o tempo de decodificação para cada macrobloco individualmente. Deste modo, medem-se os tempos de decodificação para todos os macroblocos dos vários tipos que foram codificados, utilizando o tipo de macrobloco mais adequado. Como o tempo de decodificação de um macrobloco pode ser muito pequeno para ser medido utilizando as funções disponíveis, pode-se optar por decodificar cada macrobloco várias vezes, contabilizando-se no fim este facto.

Nesta tese utiliza-se o segundo método de avaliação da complexidade pois cada macrobloco é codificado com o tipo mais adequado às suas características, obtendo-se deste modo uma estimativa mais fiável para a complexidade de decodificação dos vários tipos de macroblocos.

Para os resultados apresentados neste capítulo, cada macrobloco foi decodificado 1000 vezes (de cada vez que tinha de ser decodificado) para que se pudessem obter valores mais fiáveis para a resolução das funções de medição do tempo utilizadas. Além disso, as medidas foram obtidas usando o sistema operativo Linux, pois verificou-se que oferece uma exactidão muito maior que o Windows. Para medir o tempo de decodificação foi utilizada a função `clock` disponível na linguagem de programação C, e que pode ser facilmente integrada no código do decodificador também escrito em C.

5.4.1 Processo de decodificação

A Figura 45 e a Figura 46 mostram os fluxogramas correspondentes à decodificação de VOPs *Intra* e *Inter*, respectivamente, tal como é realizada pelo decodificador MoMuSys/IST. Os fluxogramas referem-se à decodificação de todos os macroblocos de um VOP, assumindo-se que o seu cabeçalho já foi decodificado e, portanto, já se conhece o tipo de VOP em questão: *Intra* ou *Inter*. A classificação do VOP é incluída no seu cabeçalho e indica se a textura de todos os macroblocos do VOP é forçosamente *Intra* (VOP *Intra*) ou se podem existir macroblocos cuja textura é *Intra* ou *Inter* (VOP *Inter*). Um VOP *Intra* só pode conter macroblocos com textura e forma *Intra*, enquanto que um VOP *Inter* pode conter macroblocos com informação de textura e forma do tipo *Intra* ou *Inter*. Em ambos os casos, se a textura do macrobloco for *Intra* a forma tem necessariamente de ser *Intra*.

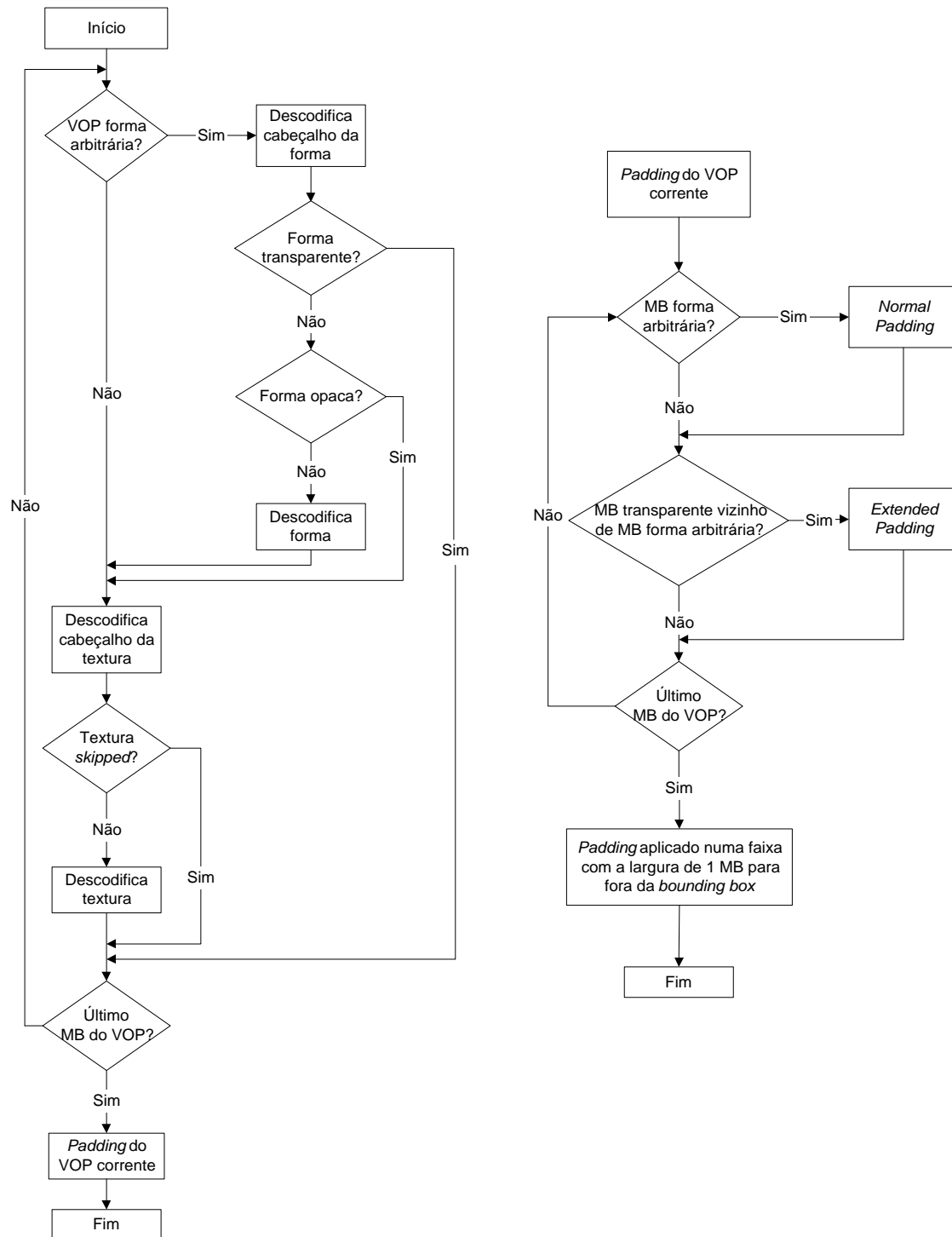


Figura 45 – Fluxograma de descodificação para VOPs Intra.

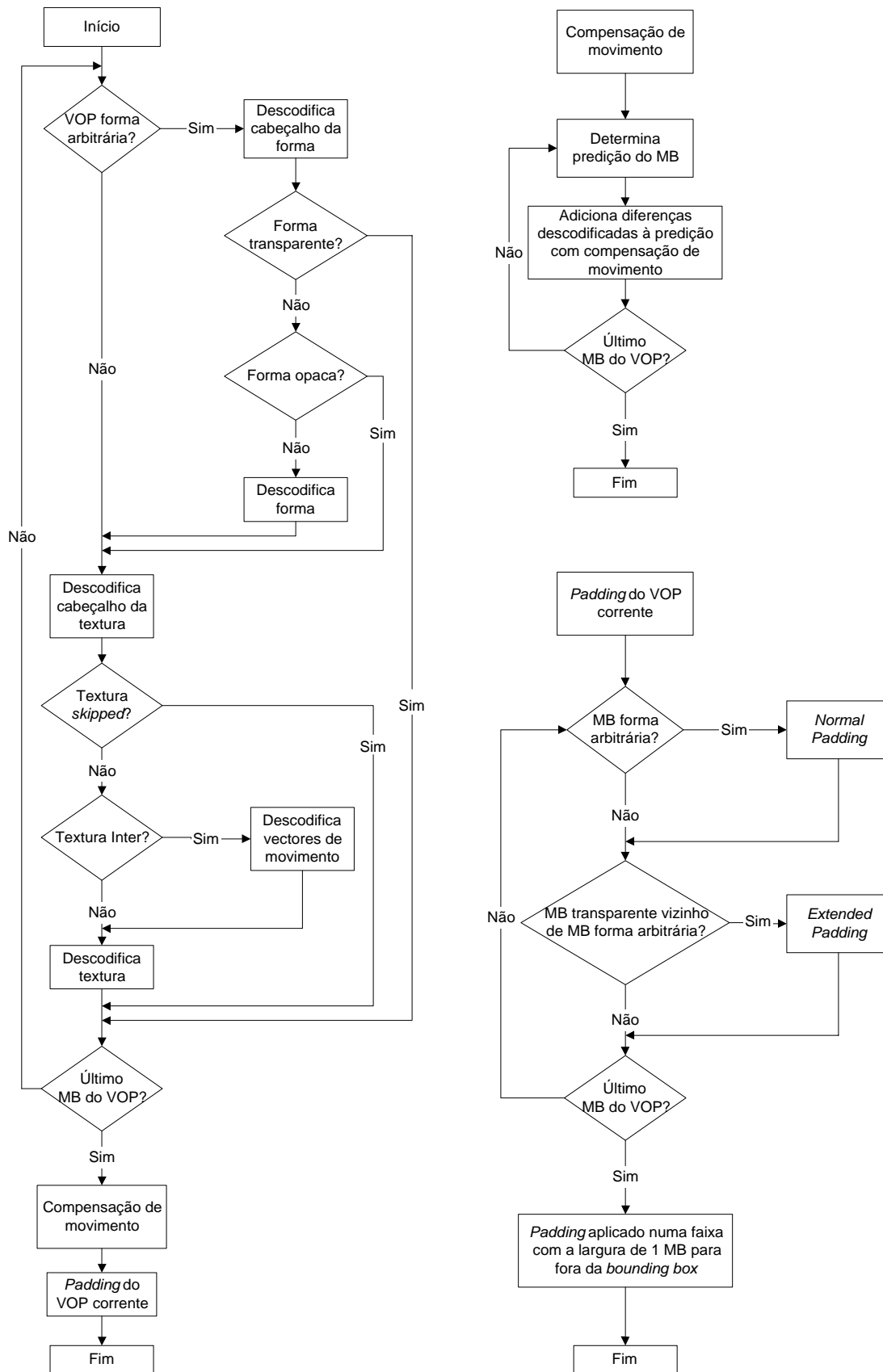


Figura 46 – Fluxograma de decodificação para VOPs Inter.

A descodificação de cada VOP consiste num ciclo em que todos os macroblocos que o constituem são sucessivamente descodificados. Caso o VOP seja rectangular, todos os macroblocos pertencem completamente ao VOP, ou seja não há macroblocos de fronteira nem macroblocos transparentes. Se o VOP tiver forma arbitrária, são descodificados todos os macroblocos pertencentes à sua *bounding box*.

5.4.1.1 Descodificação de VOPs *Intra*

Um VOP *Intra* é constituído exclusivamente por macroblocos *Intra*, em termos de textura e de forma. O ciclo de descodificação de cada um dos macroblocos que constituem o VOP inicia-se pela determinação do tipo de forma do macrobloco. Se o VOP tiver forma arbitrária, é descodificado o cabeçalho de forma do macrobloco para determinar o tipo de forma, ou seja o modo de codificação usado para a forma: transparente, opaca, *NoUpdate* ou *IntraCAE*. Se o VOP for rectangular não existe informação de forma, logo não é descodificado o cabeçalho da forma. Se um dado macrobloco é do tipo transparente, então não é para ele enviado mais nenhuma informação, e a sua descodificação termina. Se o macrobloco for opaco, não é necessário descodificar a forma. Caso contrário segue-se a descodificação da informação de forma. Descodificada a forma, passa-se à descodificação da textura. Se existir informação de textura codificada para este macrobloco, ou seja se o macrobloco não for *Skipped*, a informação de textura do macrobloco é finalmente descodificada. Este ciclo repete-se até terem sido descodificados todos os macroblocos do VOP (Figura 45).

Como se pode observar na Figura 45, no descodificador MoMuSys/IST o *padding* é efectuado no final da descodificação de cada VOP, sendo esse VOP depois utilizado para descodificar o VOP posterior. Para VOPs de forma arbitrária, o *padding* é efectuado, tal como descrito no capítulo 2, para os macroblocos de fronteira (*normal padding*) e para os macroblocos vizinhos destes (*extended padding*). Tanto para os VOPs de forma arbitrária como para os VOPs rectangulares é também aplicado *extended padding* numa faixa com a largura de um macrobloco para fora da *bounding box*, para que possa ser aplicada compensação de movimento sem restrições nos macroblocos junto aos limites da *bounding box*.

5.4.1.2 Descodificação de VOPs *Inter*

Um VOP *Inter* é codificado tendo como base outros VOPs, ou seja pode incluir macroblocos codificados em modo *Inter* (em termos de textura e de forma) para os quais pode existir compensação de movimento. Para além destes macroblocos, pode também incluir macroblocos *Intra*, ou seja macroblocos codificados de um modo independente em relação a outros VOPs. Como se pode observar na Figura 46, a descodificação da informação de forma é feita de um modo idêntico aos VOPs *Intra*. Se o macrobloco possuir informação de textura, são descodificados e armazenados os vectores de movimento (que podem ser nulos) se o macrobloco é *Inter*; este passo não é realizado se o macrobloco for *Intra*. Segue-se depois a descodificação da textura. Este processo é repetido para todos os macroblocos que constituem o VOP. No final da descodificação da informação codificada para todos macroblocos são somadas as diferenças descodificadas às respectivas predições obtidas a partir do VOP anterior ao qual foi aplicado *padding* (no fim da descodificação do VOP anterior) e compensação de movimento.

5.4.2 Medição do tempo de decodificação

A função utilizada para medir o tempo de decodificação dos macroblocos é a função `clock` disponível na linguagem de programação C; esta função devolve o tempo de processador que passou desde o início de um dado processo (neste caso, o início da decodificação). Para medir o tempo de execução de uma parte do código de decodificação, são introduzidos neste dois pontos de referência onde se mede o tempo (em relação ao início da decodificação), fazendo-se depois a diferença entre estes dois tempos. Esta função só apresenta resultados fiáveis no sistema operativo Linux, tendo uma precisão de 10 ms; na verdade, verificou-se que esta função era muito inexacta no sistema operativo Windows, não fornecendo resultados suficientemente precisos, já que variavam bastante quando se decodificava a mesma sequência várias vezes. Além disso, no Windows a função `clock` mostrou ter uma resolução muito baixa, na casa das dezenas de milissegundos.

O tempo de decodificação de um macrobloco é medido no ciclo de decodificação dos macroblocos (na Figura 45 e na Figura 46, este ciclo fica entre a decisão se o VOP tem forma arbitrária e a decisão se o macrobloco é o último do VOP). Como mostram os fluxogramas da Figura 45 e da Figura 46, é neste ciclo que é decodificada a forma, a textura e também, no caso dos macroblocos *Inter*, os vectores de movimento. Para medir com maior exactidão o tempo de decodificação, é necessário decodificar o mesmo macrobloco muitas vezes, dado que a precisão da função utilizada, 10 ms, não permite medir o tempo de decodificação de um só macrobloco com exactidão. Nesta tese optou-se por decodificar cada macrobloco 1000 vezes, pois assim os tempos obtidos já permitem obter uma estimativa credível para a complexidade relativa dos vários tipos de macroblocos, não sendo demasiado afectados pela precisão da função `clock`.

Ao tempo medido no ciclo de decodificação do macrobloco é necessário acrescentar os tempos gastos nos processos de *padding*, compensação de movimento e soma das diferenças decodificadas ao VOP anterior para os VOPs *Inter*, já que estes processos são efectuados fora do primeiro ciclo de decodificação de macrobloco. A compensação de movimento é feita ao nível do macrobloco, logo é simples adicionar o tempo gasto neste processo ao tempo de decodificação de cada macrobloco. O *padding* é uma técnica utilizada para melhorar o desempenho da estimação e compensação de movimento, sendo aplicada a um VOP para facilitar a predição do VOP seguinte. Por este motivo, a medição do tempo gasto no processo de *padding* pode ser feita de dois modos:

- **Inclusão do tempo do *padding* nos macroblocos do VOP corrente** – O tempo gasto no *padding* de cada macrobloco é incluído nos macroblocos do VOP corrente, ou seja naqueles sobre os quais é aplicado o *padding* caso pertençam à fronteira do objecto ou sejam vizinhos de macroblocos de fronteira. O decodificador MoMuSys/IST efectua o *padding* ao nível do macrobloco após a decodificação de cada VOP, para poder efectuar a compensação de movimento quando decodificar o VOP seguinte, sendo deste modo simples incluir nos macroblocos decodificados do VOP corrente sujeitos a *padding* o tempo gasto neste processo.
- **Inclusão do tempo do *padding* nos macroblocos do VOP seguinte** – O tempo gasto no *padding* é incluído nos macroblocos na mesma posição do VOP seguinte, que no fundo são os que beneficiam com este processo. Esta solução traz algumas dificuldades de

implementação, pois para além de ser necessário acumular entre VOPs os tempos medidos, pode acontecer que um macrobloco que pertença à *bounding box* do objecto e ao qual tenha sido aplicado *padding* e se tenha contado o tempo não pertença à *bounding box* do objecto no VOP seguinte, pois a *bounding box* pode variar bastante entre VOPs, nomeadamente se o objecto tiver muito movimento.

Dada a dificuldade de implementação do segundo método, foi feita a opção de incorporar o tempo gasto no processo de *padding* nos macroblocos do VOP corrente que são sujeitos a esse processo, já que o *padding* é feito ao nível do macrobloco. Após a aplicação do *padding* aos macroblocos de fronteira (*normal padding*) e aos seus vizinhos (*extended padding*) nos VOPs de forma arbitrária, tanto para estes como para os VOPs rectangulares é também aplicado *extended padding* para fora da sua *bounding box*, repetindo o valor dos *pixels* da fronteira do VOP numa faixa com a largura de um macrobloco em redor da *bounding box*. Este *padding* não é contabilizado pois afecta macroblocos que não são codificados, uma vez que estão fora da *bounding box* do objecto, pelo menos para o instante corrente, e portanto não têm um tipo definido. Uma vez que esta operação é feita na perspectiva de que possa ser útil para alguns macroblocos do VOP seguinte, é difícil atribuir o seu tempo a um dado tipo de macrobloco. Como esta operação é extremamente simples e rápida, o que significa que não traz um acréscimo significativo ao tempo de decodificação, não foi considerado este tempo em nenhum tipo de macrobloco.

Em resumo, a medição do tempo de decodificação de cada macrobloco, independentemente do seu tipo, consiste nas seguintes operações:

1. Descodificação de todos os VOPs da sequência.
2. Para cada VOP, decodifica-se cada macrobloco 1000 vezes (para rodear os problemas associados à medição de tempos demasiado pequenos) e mede-se o tempo de decodificação correspondente.
3. O tempo de decodificação de cada macrobloco corresponde ao tempo obtido no ponto anterior dividido por 1000.

5.5 Avaliação da complexidade: resultados

Para se avaliar a complexidade relativa dos vários tipos de macroblocos, foi medido o tempo de decodificação de cada macrobloco para um conjunto de sequências julgadas representativas, segundo a metodologia apresentada na secção 5.4.2. Para isso, foram utilizadas as sequências de teste referidas no capítulo 4: *Akiyo*, *Coastguard*, *News* e *Stefan*.

5.5.1 Avaliação da complexidade: objectos de vídeo rectangulares

Os objectos de vídeo rectangulares contêm exclusivamente macroblocos opacos. Os macroblocos opacos só possuem informação de textura e são considerados relevantes três tipos (não contando agora os *Skipped*), tal como especificado na Tabela 19: *Intra*, *Inter* e *Inter* com quatro vectores de movimento por macrobloco, um por cada bloco.

O principal factor que influencia o tempo de decodificação dos macroblocos opacos é o número de coeficientes da transformada DCT que são decodificados para cada macrobloco.

As figuras seguintes mostram, para todos os macroblocos das sequências de teste, a variação do tempo de decodificação de um macrobloco em função do número de coeficientes da DCT de luminância e crominância que o constituem. Os gráficos mostram o tempo de decodificação de todos os macroblocos da sequência em questão (determinado como especificado no fim da secção 5.4.2), sendo que cada ponto corresponde a um macrobloco decodificado. Esta avaliação foi feita utilizando as sequências rectangulares *Akiyo*, no formato espacial QCIF (sub-amostragem 4:2.0 para a crominância) e com uma frequência de trama de 30 tramas/s, codificada no perfil *Simple* e nível 2, com um débito binário de 128 kbit/s (Figura 47), e *News*, no formato espacial CIF e com uma frequência de trama de 30 tramas/s, codificada no perfil *Simple* e nível 3 (Figura 48), com um débito binário de 384 kbit/s. Para determinar a influência da escolha do perfil no tempo de decodificação, mostram-se também os tempos de decodificação para a sequência *News* no formato espacial CIF, codificada no perfil *Core* e nível 2, com um débito binário de 2000 kbit/s (Figura 49). É utilizada a mesma escala em todos os gráficos para que se possa fazer uma comparação directa entre eles.

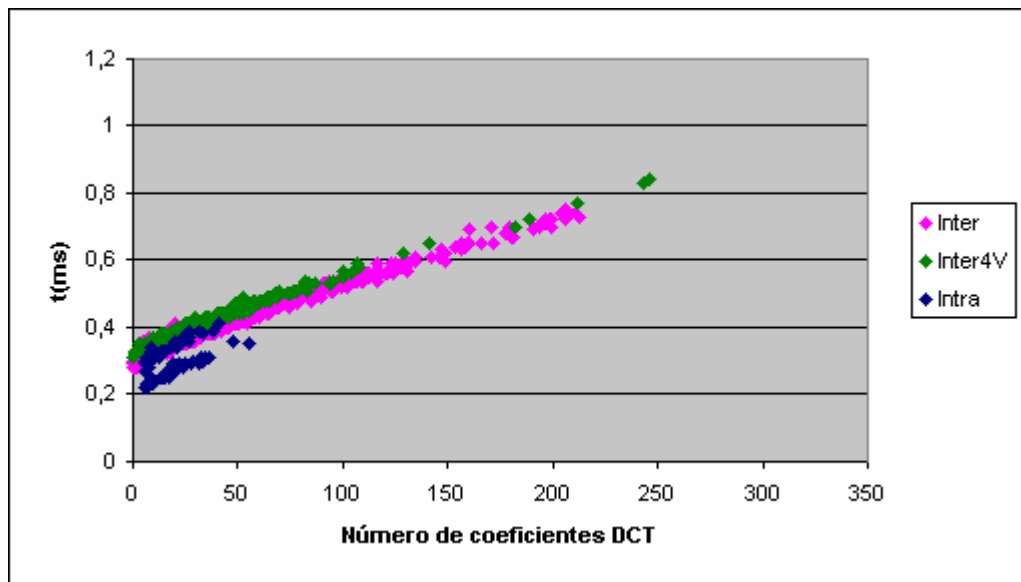


Figura 47 – Tempo de decodificação dos macroblocos para a sequência *Akiyo*: rectangular, QCIF, Simple@L2, 128 kbit/s (número de macroblocos usados: Intra – 99, Inter – 11287, Inter4V – 274).

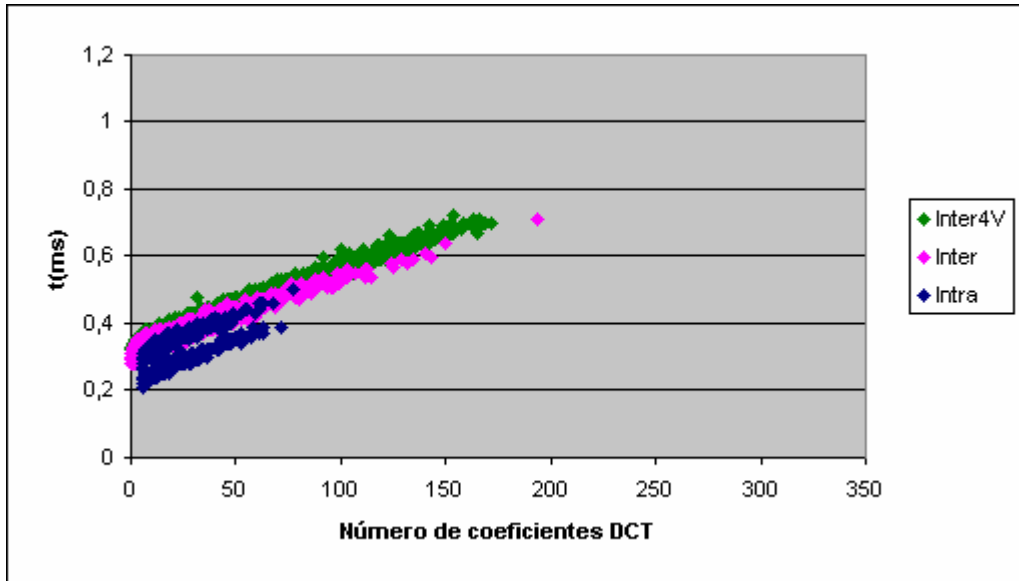


Figura 48 – Tempo de decodificação dos macroblocos para a sequência News: rectangular, CIF, Simple@L3, 384 kbit/s (número de macroblocos usados: Intra – 536, Inter – 35141, Inter4V – 6225).

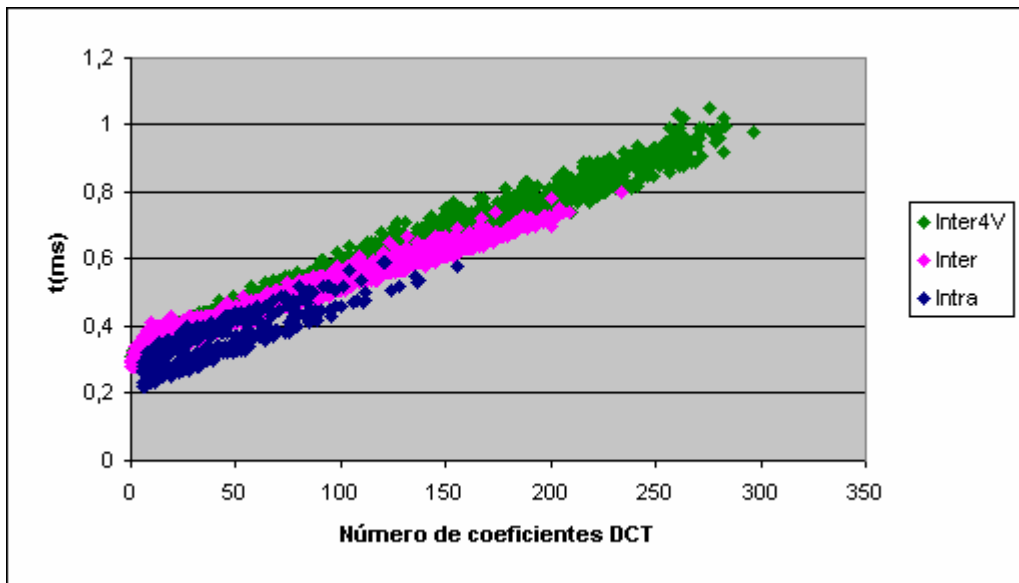


Figura 49 – Tempo de decodificação dos macroblocos para a sequência News: rectangular, CIF, Core@L2, 2000 kbit/s (número de macroblocos usados: Intra – 538, Inter – 89926, Inter4V – 5968).

Como se pode observar nas figuras anteriores, o tempo de decodificação dos macroblocos varia de um modo aproximadamente linear com o número de coeficientes da transformada DCT por macrobloco (incluindo luminância e crominâncias). O número de coeficientes da DCT que são decodificados pode variar entre 0 e 384 para o formato de sub-amostragem das crominâncias 4:2:0 (crominâncias com metade da resolução da luminância nas duas direcções). Os macroblocos *Inter* podem não ter coeficientes, mas os macroblocos *Intra* têm, pelo menos, o coeficiente DC para cada um dos 4 blocos de luminância e dos 2 blocos de crominância, ou seja os macroblocos *Intra* têm, no mínimo, 6 coeficientes DCT.

Por outro lado, se cada um dos 6 blocos tiver o número máximo de coeficientes ou seja 64 (8×8) coeficientes, cada macrobloco pode conter, no máximo, 384 coeficientes da DCT.

Os gráficos das figuras anteriores mostram que as curvas, para os três tipos relevantes de macroblocos em termos de codificação de textura – *Intra*, *Inter* e *Inter4V* – têm aproximadamente o mesmo declive. Os macroblocos *Intra* distribuem-se entre duas sub-curvas distintas (Figura 47, Figura 48 e Figura 49), devendo-se esta distinção ao uso ou não de predição para os coeficientes AC. Os macroblocos *Intra* que utilizam predição AC demoram mais tempo a ser decodificados, dado o acréscimo de tempo que esta ferramenta implica, como já foi atrás referido. A decodificação dos macroblocos *Inter* é um pouco mais demorada, em termos de valor máximo para um dado número de coeficientes DCT, que os macroblocos *Intra* com predição AC, enquanto que a decodificação dos macroblocos *Inter* demora aproximadamente o mesmo tempo, em valores máximos para um dado número de coeficientes DCT, que a decodificação dos macroblocos *Inter* que utilizam quatro vectores de movimento. Nas figuras não se distinguem outras zonas significativas, o que comprova, por exemplo, que a mudança de passo de quantificação não conduz a um acréscimo significativo de tempo em relação aos macrobloco em que não há mudança de passo.

A Figura 48 e a Figura 49 mostram o tempo de decodificação dos vários tipos de macroblocos para a mesma sequência de vídeo mas usando perfis e níveis diferentes. Os valores máximos para o mesmo número de coeficientes DCT e o declive das curvas mantêm-se em ambos os casos, mas o aumento de débito binário resultante da passagem do perfil *Simple@L3* (384 kbit/s) para o perfil *Core@L2* (2000 kbit/s) faz com que hajam mais macroblocos com um maior número de coeficientes DCT e menos macroblocos cuja textura não é codificada (*Skipped*). Isto significa que a “nuvem” de pontos tende a crescer para a direita dos gráficos mas os valores máximos do tempo de decodificação para um dado número de coeficientes tendem a manter-se.

5.5.2 Avaliação da complexidade: objectos de vídeo de forma arbitrária

Nos macroblocos que possuem informação de forma não transparente nem opaca (macroblocos fronteira), esta pode ser codificada dos seguintes modos: *IntraCAE*, *InterCAE* ou *NoUpdate*. Os tempos representados nas figuras seguintes correspondem ao tempo total de decodificação de cada macrobloco, ou seja à soma dos tempos de decodificação da forma, da textura, dos vectores de movimento da textura¹⁴ (para macroblocos *Inter* e *Inter4V*) e do *padding* para os macroblocos sujeitos a este processo.

A Figura 50, a Figura 51 e a Figura 52 mostram o tempo de decodificação dos macroblocos da sequência *Coastguard* codificada no perfil *Core* e nível 1, com um débito binário de 384 kbit/s, para macroblocos com a textura codificada em modo *Intra*, *Inter* e *Inter4V*, respectivamente; para cada figura mostra-se o tempo de decodificação para cada tipo de macrobloco em termos de codificação de forma para um tipo fixo de codificação de textura, em função do número de coeficientes da transformada DCT. Esta sequência tem o formato espacial QCIF, frequência de trama de 30 tramas/s e é composta por 4 objectos de

¹⁴ O tempo de decodificação dos vectores de movimento é referido explicitamente porque este tempo é medido individualmente. O tempo de decodificação dos vectores de movimento para a forma é incluído no tempo de decodificação da forma, devido à implementação do decodificador, e por esse motivo não é referido.

forma arbitrária como se pode ver no capítulo 4, onde as sequências de teste foram definidas. A Figura 53, a Figura 54 e a Figura 55 mostram o tempo de decodificação dos macroblocos da sequência *Stefan* codificada no perfil *Core* e nível 2, com um débito binário de 2000 kbit/s, para macroblocos com a textura codificada em modo *Intra*, *Inter* e *Inter4V*, respectivamente. Esta sequência é codificada no formato espacial CIF, com uma frequência de trama de 30 tramas/s e é composta por 2 objectos de forma arbitrária, como se pode ver no capítulo 4.

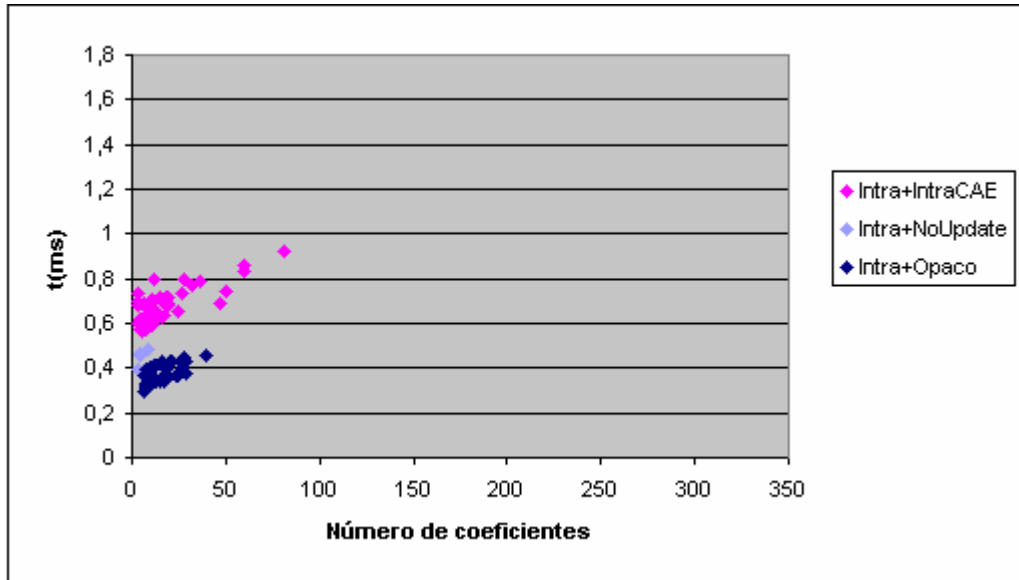


Figura 50 – Tempo de decodificação para vários tipos de codificação da forma: *Coastguard*, *QCIF*, *Core@L1*, 384 kbit/s, textura em modo *Intra* (número de macroblocos usados: *Opaco* – 62, *NoUpdate* – 4, *IntraCAE* – 63).

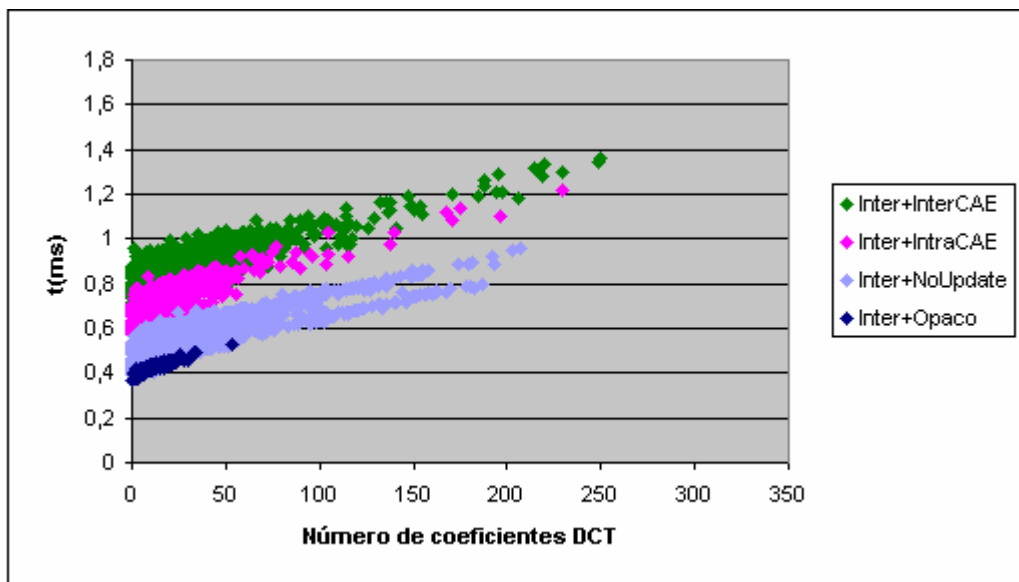


Figura 51 – Tempo de decodificação para vários tipos de codificação da forma: *Coastguard*, *QCIF*, *Core@L1*, 384 kbit/s, textura em modo *Inter* (número de macroblocos usados: *Opaco* – 290, *NoUpdate* – 28182, *IntraCAE* – 634, *InterCAE* – 5247).

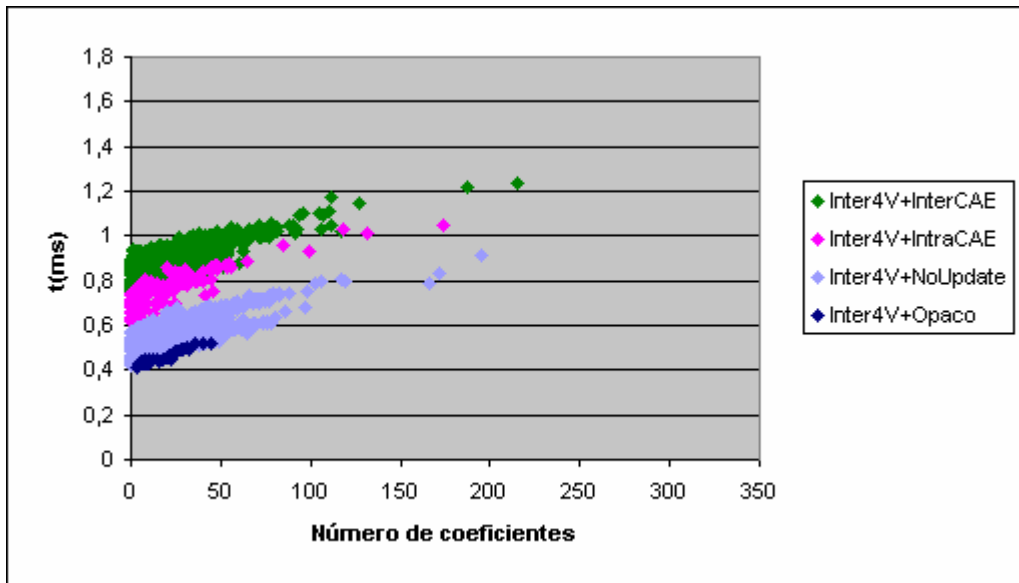


Figura 52 – Tempo de decodificação para vários tipos de codificação da forma: Coastguard, QCIF, Core@L1, 384 kbit/s, textura em modo Inter4V (número de macroblocos usados: Opaco – 38, NoUpdate – 2329, IntraCAE – 138, InterCAE – 1207).

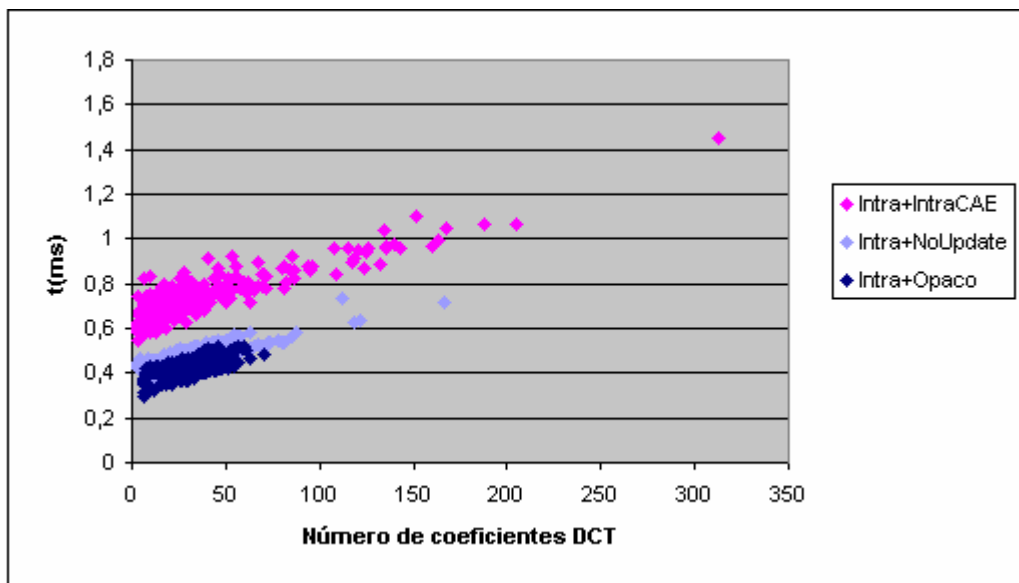


Figura 53 – Tempo de decodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Intra (número de macroblocos usados: Opaco – 356, NoUpdate – 485, IntraCAE – 264).

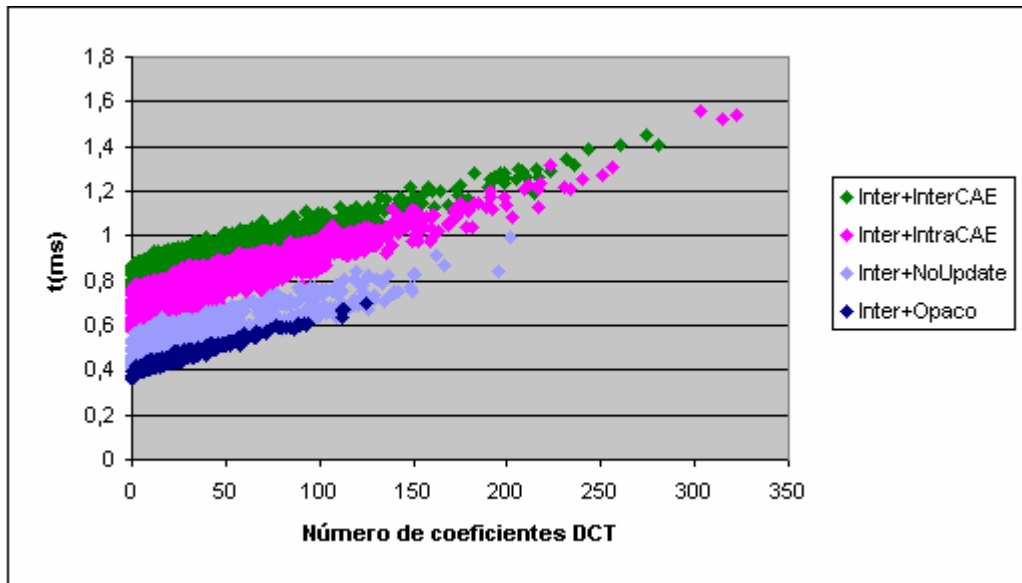


Figura 54 – Tempo de decodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Inter (número de macroblocos usados: Opaco – 1175, NoUpdate – 74314, IntraCAE – 5334, InterCAE – 2364).

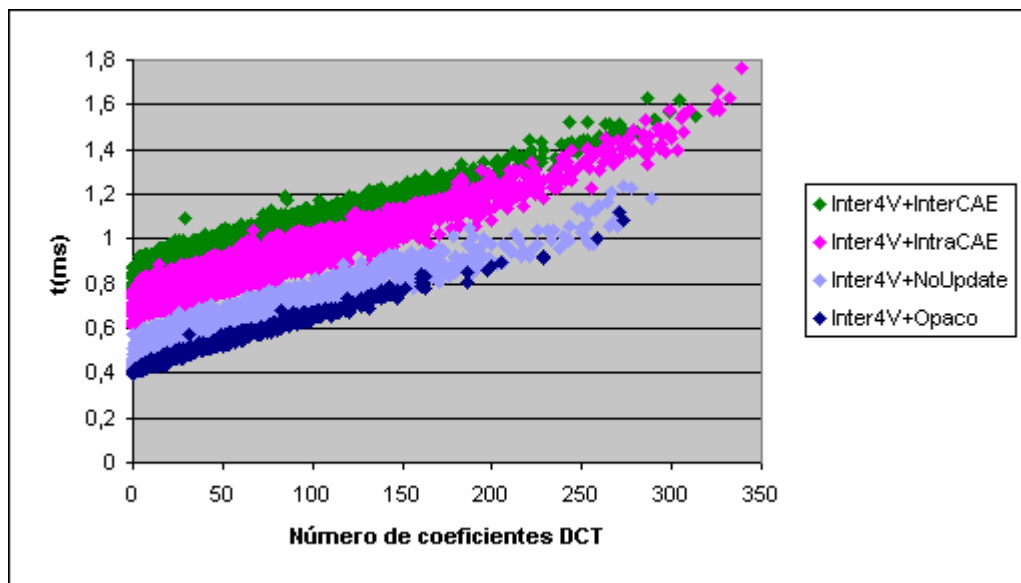


Figura 55 – Tempo de decodificação para vários tipos de codificação da forma: Stefan, CIF, Core@L2, 2000 kbit/s, textura em modo Inter4V (número de macroblocos usados: Opaco – 1061, NoUpdate – 25207, IntraCAE – 6056, InterCAE – 2351).

Como se pode observar nas figuras anteriores, para um dado modo de codificação da textura e para o mesmo número de coeficientes DCT, o tempo de decodificação total (textura+forma) aumenta na seguinte ordem, em termos de tipos de codificação de forma: Opaco, NoUpdate, IntraCAE e InterCAE.

O número de coeficientes da DCT que são decodificados varia entre 0 e 364, para os macroblocos opacos, como já foi referido atrás. Porém, quando existe informação de forma, o número mínimo de coeficientes para macroblocos Intra é 3 e não 6, como acontece com os

macroblocos opacos. Este caso corresponde a um macrobloco de fronteira onde apenas um dos seus quatro blocos de luminância contém informação de textura (1 coeficiente DC), sendo neste caso os três restantes blocos de luminância transparentes. Assim, no mínimo, existe um coeficiente DC da transformada DCT para a luminância desse bloco e dois coeficientes DC para as crominâncias do macrobloco (ou bloco, neste caso). Os coeficientes DC para as crominâncias estão sempre presentes em macroblocos *Intra*, já que os valores das duas crominâncias abrangem os quatro blocos do macrobloco.

A descodificação dos macroblocos opacos é a mais rápida, já que estes macroblocos não contêm informação de forma e basta colocar todos os *shapels* do macrobloco respectivo a 255. Os macroblocos cuja forma é codificada em modo *NoUpdate* são mais demorados no processo de descodificação que os macroblocos opacos, já que têm de se descodificar os vectores de movimento e tem de ser aplicada compensação de movimento em relação à forma do VOP anterior. É possível distinguir nos gráficos referentes a este tipo de macroblocos (*NoUpdate*) duas sub-curvas, mais evidentes para números elevados de coeficientes DCT, onde os pontos estão menos aglomerados. A sub-curva cujo tempo de descodificação é menor corresponde a macroblocos não sujeitos a *padding* (por estarem no interior do VOP) e a zona com o tempo de descodificação maior corresponde a macroblocos sujeitos ao processo de *padding* (por estarem na fronteira do VOP).

Pode também verificar-se nas figuras anteriores que os macroblocos cuja forma é do tipo *InterCAE* são mais demorados no processo de descodificação que os macroblocos do tipo *IntraCAE*, em termos dos valores máximos do tempo para o mesmo número de coeficientes DCT. O acréscimo de tempo de descodificação da forma *InterCAE* relativamente à forma do tipo *IntraCAE* deve-se principalmente ao facto de a descodificação da forma *InterCAE* considerar o VOP anterior ao qual é aplicada compensação de movimento.

A variação existente no tempo de descodificação para um mesmo número de coeficientes DCT que se pode observar nos gráficos está essencialmente relacionada com a complexidade da forma dos objectos, pois formas mais complicadas demoram mais tempo a serem descodificadas (para um mesmo número de coeficientes DCT envolvidos).

5.5.3 Avaliação da complexidade: macroblocos transparentes

Um macrobloco transparente não contém informação de textura nem de forma, pois nenhum dos *pixels* desse macrobloco pertence ao objecto, embora os *pixels* pertençam à sua *bounding box*. Na Figura 56 e na Figura 57 estão representados os histogramas do tempo de descodificação dos macroblocos transparentes para as sequências de teste utilizadas que incluem objectos de forma arbitrária.

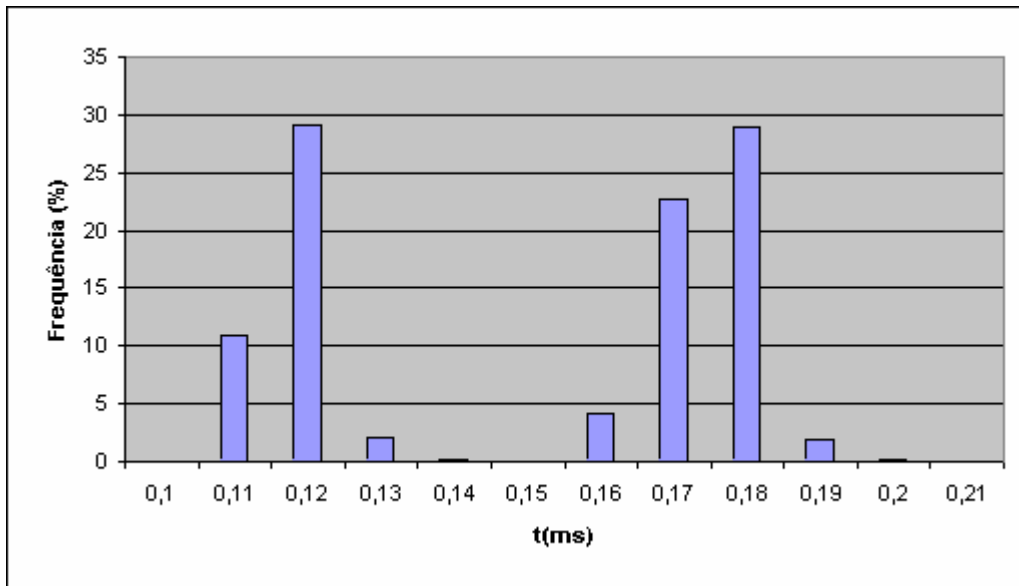


Figura 56 – Histograma para o tempo de decodificação de macroblocos transparentes: Coastguard, QCIF, Core@L1, 384 kbit/s (número de macroblocos usados: 7673).

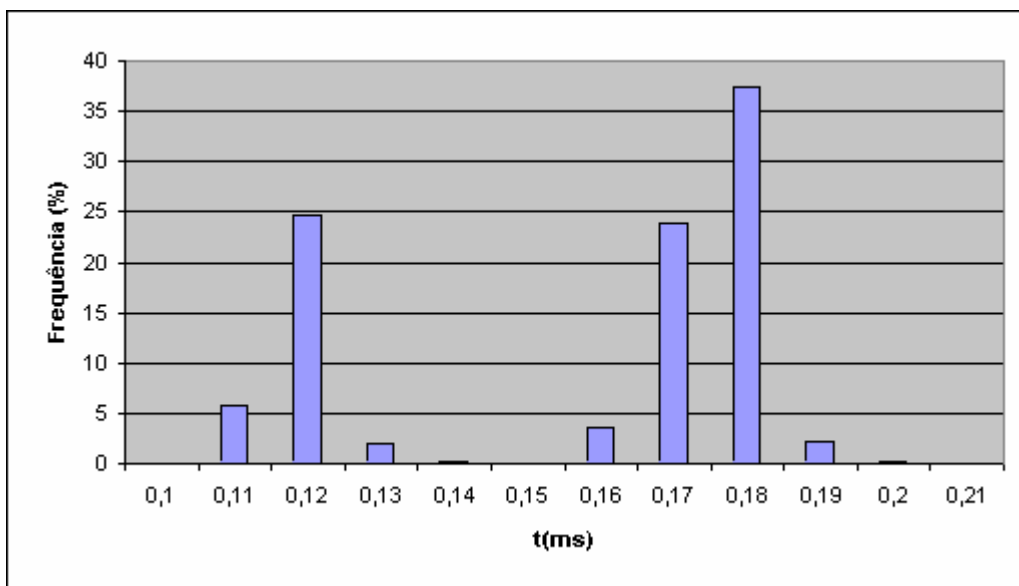


Figura 57 – Histograma para o tempo de decodificação de macroblocos transparentes: Stefan, CIF, Core@L2, 2000 kbit/s (número de macroblocos usados: 10500).

As funções utilizadas para medir o tempo de decodificação no sistema operativo Linux têm uma resolução de 10 ms, o que leva à existência de alguma dispersão nos histogramas em torno dos valores onde a frequência é maior. Os histogramas mostram claramente a existência de duas zonas distintas que correspondem a dois tipos de macroblocos transparentes: os que estão junto da fronteira do objecto, e logo são sujeitos a *padding* e consequentemente mais demorados na decodificação, e os restantes que se encontram mais afastados da fronteira do objecto e que não sofrem qualquer processamento adicional. Como foi referido no capítulo 2, os macroblocos transparentes que estão junto de macroblocos de fronteira do objecto em questão estão sujeitos ao processo de *padding* (*extended padding*); este processo consiste na colocação dos *pixels* deste macrobloco com o valor dos *pixels* na fronteira dos macroblocos

adjacentes (por sua vez também já sujeitos a *padding*) para facilitar a predição. Estes macroblocos transparentes demoram mais tempo a ser decodificados do que os macroblocos transparentes não sujeitos a *padding*, pois este processo origina um incremento de processamento.

5.5.4 Comparação da complexidade para os vários tipos de macroblocos

Para comparar a complexidade, ou seja o tempo de decodificação entre os vários tipos de macroblocos, apresentam-se de seguida gráficos com os tempos de decodificação dos macroblocos para as sequências de teste e outras sequências. Com este fim, foram codificadas as sequências de teste atrás referidas e ainda outras sequências para complementar os resultados alcançados, nomeadamente: i) *Akiyo*, rectangular, QCIF, *Core@L1*; ii) *Children*, QCIF, *Core@L1*; iii) *Coastguard*, QCIF, *Core@L1*, só com tramas *Intra*; iv) *News*, rectangular, QCIF, *Simple@L3*; v) *News*, rectangular, QCIF, *Core@L2*; vi) *Stefan*, rectangular, QCIF, *Simple@L3*; vii) *Stefan*, rectangular, QCIF, *Core@L2*, só com tramas *Intra* e *Weather*, QCIF, *Core@L1*. Com este conjunto adicional de sequências conseguem-se resultados para um número bastante maior de macroblocos codificados e com uma maior variação em termos do número de coeficientes DCT.

5.5.4.1 Comparação entre os tipos de macroblocos para objectos rectangulares

A Figura 58 mostra o tempo de decodificação para os vários tipos de macroblocos para sequências rectangulares, ou seja sem informação de forma. Em valores máximos, e para o mesmo número de coeficientes DCT, os macroblocos *Inter4V* demoram ligeiramente mais tempo a ser decodificados que os macroblocos *Inter*. Nas sequências utilizadas, este facto é tanto mais significativo quanto maior é o número de coeficientes DCT codificados. Os macroblocos *Intra* são decodificados um pouco mais rapidamente que os macroblocos *Inter* e *Inter4V*, nomeadamente em termos de valores máximos de tempo para um dado número de coeficientes DCT.

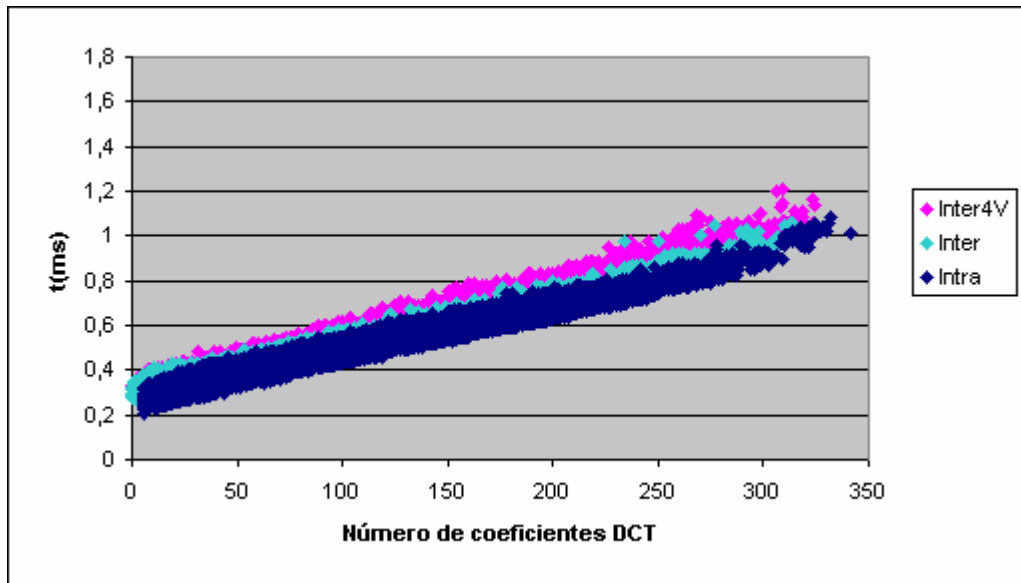


Figura 58 – Tempo de decodificação para os macroblocos (só com informação de textura) em seqüências rectangulares (número de macroblocos usados: Intra – 61653, Inter – 252982, Inter4V – 30317).

O tempo de decodificação dos macroblocos pertencentes a objectos rectangulares cuja textura é codificada através de macroblocos *Skipped* está representado na Figura 59. Para estes macroblocos não existem coeficientes DCT, utilizando-se por isso um histograma com a percentagem do número de macroblocos em função do seu tempo de decodificação. Os macroblocos deste tipo são decodificados muito rapidamente, demorando um tempo semelhante aos macroblocos transparentes sem *padding*, pois não têm informação de textura nem de forma a processar. O tempo de processamento corresponde à decodificação do cabeçalho da textura, e a dispersão que se observa no histograma deve-se à precisão da função utilizada para medir o tempo de decodificação.

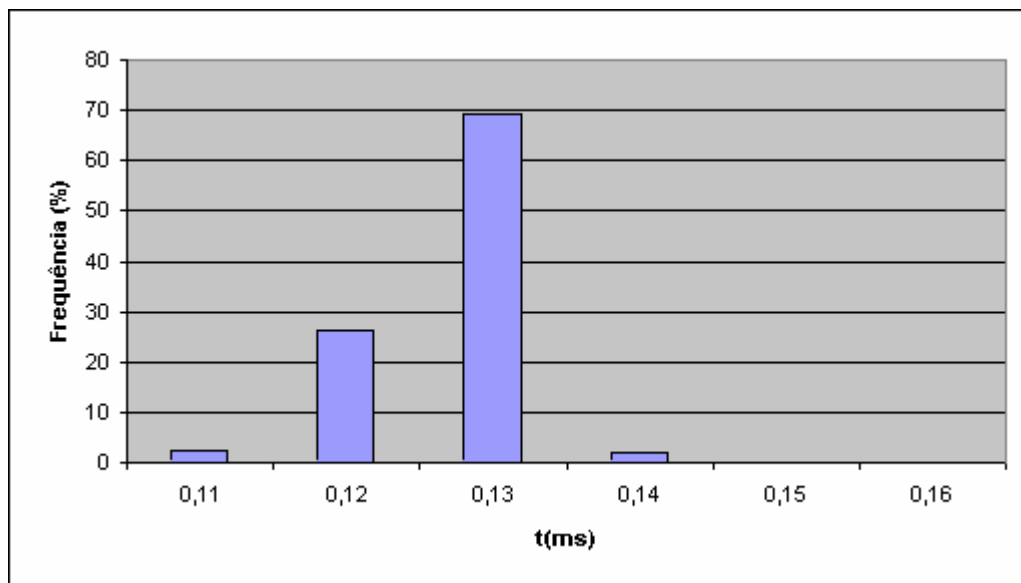


Figura 59 – Histograma do tempo de decodificação dos macroblocos *Skipped* para objectos rectangulares (número de macroblocos usados: 114660).

5.5.4.2 Comparação entre os tipos de macroblocos para objectos com forma arbitrária

Esta secção apresenta uma comparação entre os vários tipos de macroblocos que servem para codificar os objectos de vídeo com forma arbitrária.

- **Forma opaca versus textura**

Na Figura 60 está ilustrado o tempo de decodificação para os macroblocos opacos que constituem os objectos de vídeo de forma arbitrária. Relativamente aos macroblocos opacos de objectos rectangulares mantém-se a hierarquia referida para a textura, mas estes macroblocos têm uma decodificação ligeiramente mais demorada para o mesmo número de coeficientes DCT. Isto acontece porque nos objectos de forma arbitrária, contrariamente aos rectangulares, é necessário decodificar o cabeçalho da forma (o que implica leitura do disco) para determinar o seu tipo e colocar todos os *shapels* a 255, o que corresponde a um incremento no tempo de decodificação.

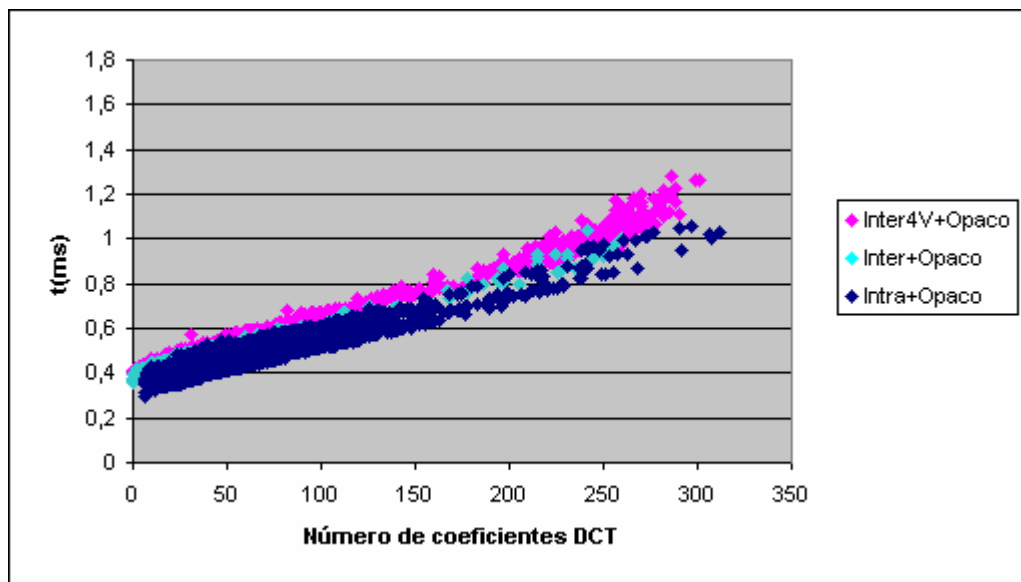


Figura 60 – Tempo de decodificação para os macroblocos opacos em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra – 19272, Inter – 1544, Inter4V – 1391).

- **Forma NoUpdate versus textura**

A Figura 61 mostra o tempo de decodificação para os macroblocos cuja forma é codificada com o tipo *NoUpdate* em função do número de coeficientes DCT que são decodificados. Para o mesmo número de coeficientes DCT, estes macroblocos demoram um pouco mais de tempo a serem decodificados que os macroblocos opacos, já que tem de ser aplicada compensação de movimento à forma do VOP anterior. O tempo de decodificação aumenta, em termos de valor máximo para o mesmo número de coeficientes DCT, com o tipo de codificação da textura tal como nos outros tipos de macroblocos: *Intra*, *Inter*, *Inter4V*.

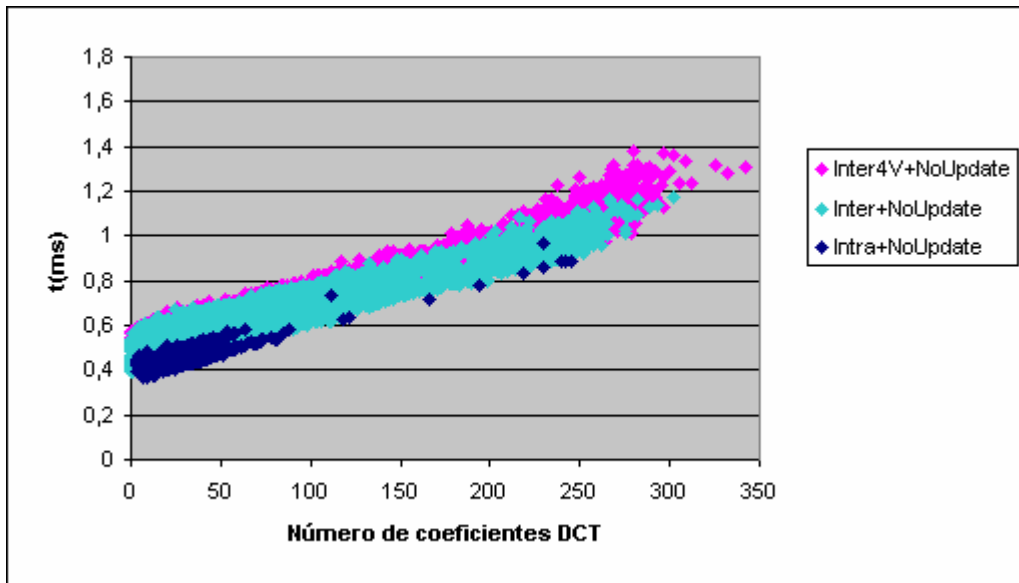


Figura 61 – Tempo de decodificação para os macroblocos com codificação de forma NoUpdate em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra – 500, Inter – 73256, Inter4V – 29913).

- **Forma IntraCAE e InterCAE versus textura**

A Figura 62 ilustra o tempo de decodificação dos macroblocos cuja informação de forma é codificada recorrendo a *Context-based Arithmetic Encoding* (CAE), ou seja macroblocos com forma codificada com os tipos *IntraCAE* e *InterCAE*. Relativamente aos macroblocos referidos anteriormente (com forma codificada Opaca e NoUpdate), os macroblocos com forma codificada *IntraCAE* e *InterCAE* são claramente os que demoram mais tempo a serem decodificados, considerando os valores máximos para o mesmo número de coeficientes DCT. Da Figura 62 pode concluir-se que os macroblocos com informação de forma *InterCAE* têm uma decodificação ligeiramente mais demorada que os macroblocos correspondentes com informação de forma *IntraCAE*.

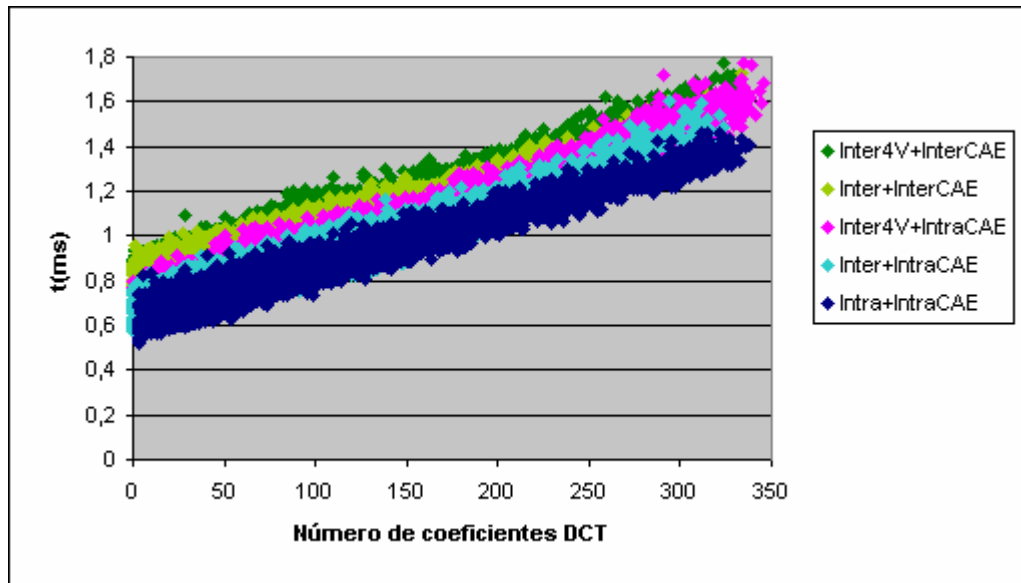


Figura 62 – Tempo de decodificação para os macroblocos com codificação de forma IntraCAE e InterCAE em sequências com objectos de forma arbitrária (número de macroblocos usados: Intra+IntraCAE – 6997, Inter+IntraCAE – 8583, Inter4V+IntraCAE – 12131, Inter+InterCAE – 16251, Inter4V+InterCAE – 11284).

- **Textura *Skipped* versus forma**

O tempo de decodificação dos macroblocos do tipo *Skipped* está representado da Figura 63 à Figura 66 para os vários tipos de macroblocos em termos de codificação de forma. Para estes macroblocos não existem coeficientes DCT (não podendo por isso ser estudados como os tipos de macroblocos anteriores em função do número de coeficientes DCT codificados), utilizando-se por isso histogramas com o número de macroblocos em função do seu tempo de decodificação. Os macroblocos deste tipo com decodificação mais rápida são os que têm forma opaca (Figura 63). Relativamente aos macroblocos opacos de objectos rectangulares, estes macroblocos têm uma decodificação um pouco mais demorada, já que é necessário decodificar a informação de forma.

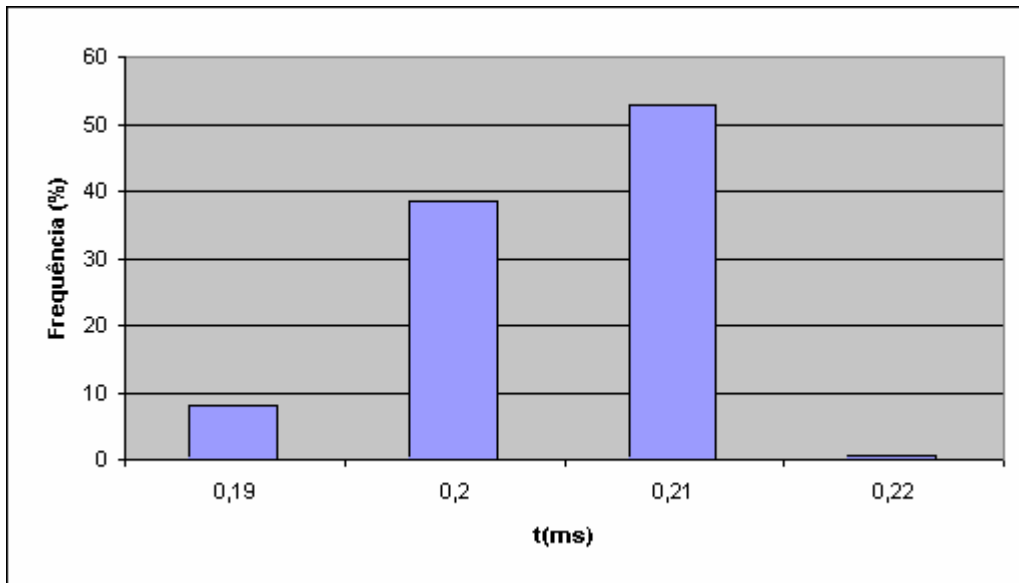


Figura 63 – Histograma do tempo de decodificação dos macroblocos *Skipped* para forma opaca (número de macroblocos usados: 184).

Os macroblocos com informação de forma codificada com o tipo *NoUpdate* (Figura 64) são mais demorados do que os macroblocos opacos em termos de decodificação devido à necessidade de decodificar o cabeçalho da forma e aplicar a compensação de movimento. As duas regiões que se observam no histograma da Figura 64, em torno de 0,23 s e de 0,33 s, correspondem, respectivamente, aos macroblocos aos quais não é aplicado *padding* (decodificação mais rápida) e aos macroblocos sujeitos ao processo de *padding* (decodificação mais demorada). Este histograma mostra que a frequência de ocorrência de macroblocos com forma codificada segundo o tipo *NoUpdate* e não sujeitos a *padding* é bastante superior à frequência de ocorrência de macroblocos do mesmo tipo mas sujeitos a *padding*, isto para as sequências utilizadas, o que é natural uma vez que a grande maioria dos macroblocos *Skipped* aparece no interior, mais uniforme, de alguns objectos e logo “longe” das zonas onde o *padding* é aplicado.

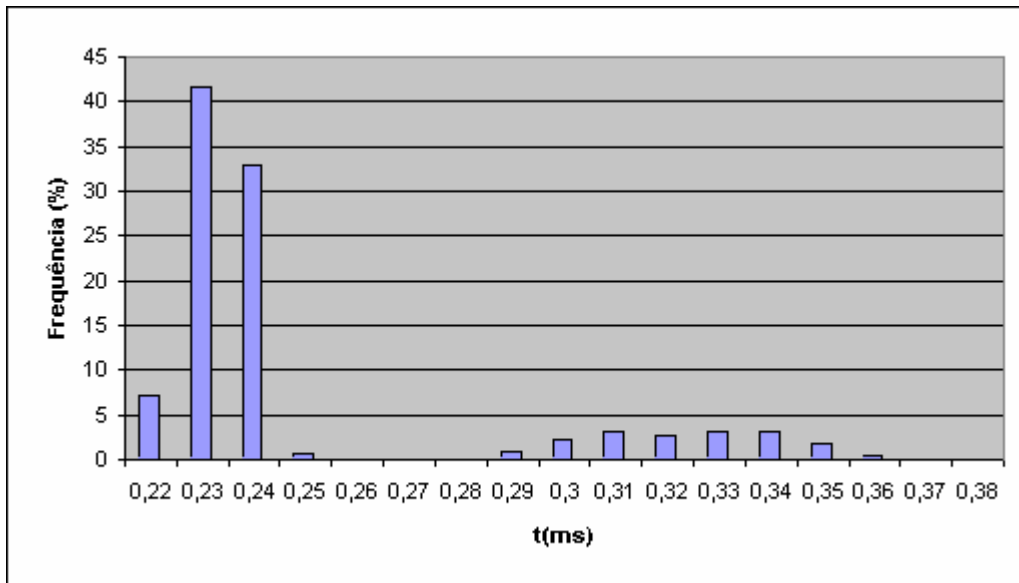


Figura 64 – Histograma do tempo de decodificação dos macroblocos *Skipped* para forma *NoUpdate* (número de macroblocos usados: 10914).

A Figura 65 e a Figura 66 mostram os histogramas do tempo de decodificação dos macroblocos cuja textura é codificada com o modo *Skipped* e a forma é codificada com os tipos *IntraCAE* e *InterCAE*, respectivamente. Ambos os tipos de macroblocos demoram mais tempo a serem decodificados que os macroblocos *Skipped* referidos anteriormente. Os macroblocos *Skipped InterCAE* são mais morosos no processo de decodificação que os *Skipped IntraCAE* em virtude de a decodificação da informação de forma *InterCAE* ser mais demorada que a *IntraCAE*. A dispersão verificada em ambos os histogramas deve-se ao tempo gasto na decodificação da informação de forma que depende da sua complexidade (dentro dum macrobloco), o que leva a que o tempo total de decodificação dos macroblocos varie bastante para macroblocos do mesmo tipo.

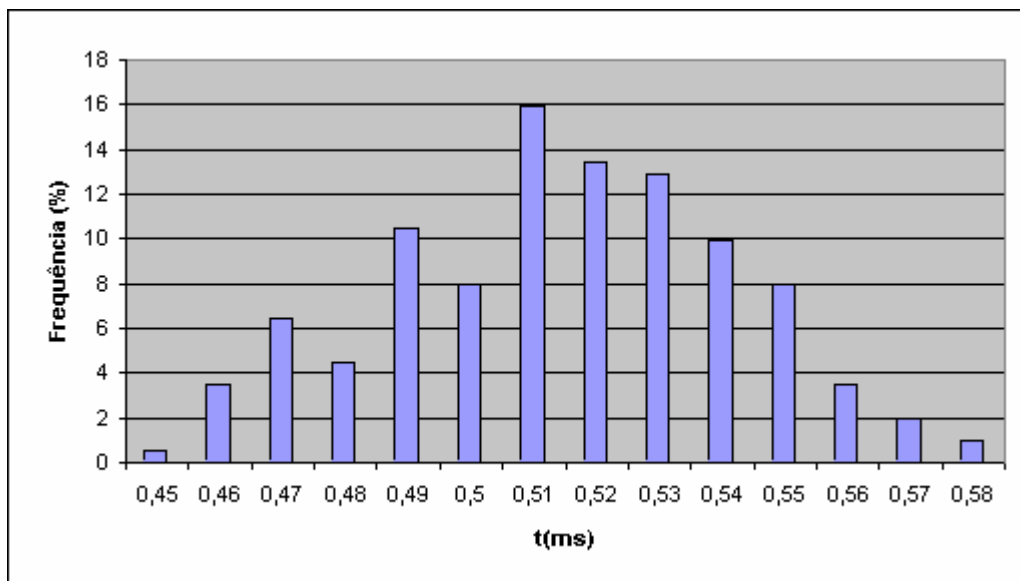


Figura 65 – Histograma do tempo de decodificação dos macroblocos *Skipped* para forma *IntraCAE* (número de macroblocos usados: 201).

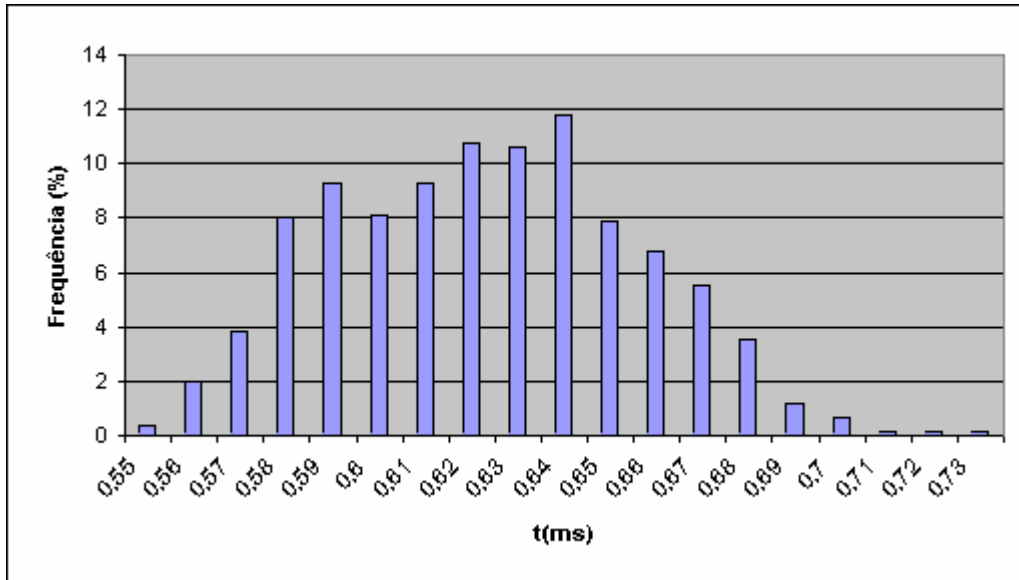


Figura 66 – Histograma do tempo de decodificação dos macroblocos *Skipped* para forma *InterCAE* (número de macroblocos usados: 763).

• Comparando tipos de macroblocos

As figuras apresentadas nesta secção permitem estabelecer uma hierarquia em termos da complexidade de decodificação dos vários tipos de macroblocos. Senão veja-se:

- Os macroblocos transparentes são os que demoram menos tempo a ser decodificados, já que não contêm informação de textura nem de forma. De notar que existem dois tipos de macroblocos transparentes: os que estão afastados da fronteira do objecto e os que estão junto da fronteira do objecto, sujeitos por isso a *padding* e logo com um consequente acréscimo no tempo de decodificação.
- Os macroblocos cuja textura é codificada em modo *Skipped* em objectos rectangulares têm, aproximadamente, o mesmo tempo de decodificação que os macroblocos transparentes mais rápidos, já que a sua decodificação é semelhante – nenhum deles contém informação de textura nem de forma a processar. Os macroblocos com textura *Skipped* mas que pertencem a objectos de forma arbitrária precisam de mais tempo de decodificação que os anteriores, já que é necessário decodificar pelo menos o cabeçalho da forma. Para decodificar os macroblocos *Skipped* cuja informação de forma é do tipo *NoUpdate*, é necessário utilizar a forma do VOP anterior para além da decodificação do cabeçalho da forma. Dentro deste tipo de macroblocos existem duas classes distintas para o tempo de decodificação: macroblocos sujeitos ou não ao processo de *padding*. Como seria de esperar, o tempo de decodificação aumenta quando a informação de forma CAE necessita de ser decodificada, ou seja quando se utilizam os modos *InterCAE* e *IntraCAE*. Neste caso, a forma no modo *InterCAE* demora mais tempo a ser decodificada que a forma no modo *IntraCAE*.
- O tempo de decodificação dos macroblocos cuja textura é codificada usando coeficientes DCT depende do número de coeficientes da transformada DCT envolvidos. Para o mesmo tipo de codificação da informação de forma e considerando

o mesmo número de coeficientes DCT, o tempo máximo de decodificação aumenta na seguinte ordem: *Intra*, *Inter* e *Inter4V*. No entanto, as diferenças são muito pequenas e é difícil estabelecer uma relação clara para toda a gama de coeficientes DCT.

- Considerando o mesmo tipo de codificação da textura, o tempo de decodificação aumenta da seguinte maneira com o modo de codificação da forma: *Opaco*, *NoUpdate*, *IntraCAE*, *InterCAE*.

A Tabela 20 mostra os valores da média, desvio padrão, moda e mediana para o tempo de decodificação dos vários tipos de macroblocos para todas as sequências utilizadas, ou seja as sequências de teste definidas no capítulo 4 e as sequências adicionais descritas em 5.5.4. Estes valores comprovam as relações em termos de tempo de decodificação referidas anteriormente, dando uma ideia da complexidade relativa dos vários tipos de macroblocos. Para dar uma ideia do significado estatístico das conclusões aqui obtidas, a Tabela 20 mostra também o número de macroblocos decodificados para cada tipo de macrobloco considerado.

	Nº de MB codificados	Média (ms)	Desvio padrão (ms)	Moda (ms)	Mediana (ms)
Transparente	53232	0.15	0.03	0.18	0.16
<i>Skipped</i> (VO rect.)	114660	0.13	0.01	0.13	0.13
<i>Intra</i> (VO rect.)	61653	0.43	0.15	0.29	0.38
<i>Inter</i> (VO rect.)	252982	0.37	0.08	0.32	0.34
<i>Inter4V</i> (VO rect.)	30317	0.47	0.14	0.38	0.43
<i>Skipped+Opaco</i>	184	0.20	0.01	0.21	0.21
<i>Intra+Opaco</i>	19272	0.47	0.06	0.48	0.47
<i>Inter+Opaco</i>	1544	0.44	0.08	0.40	0.42
<i>Inter4V+Opaco</i>	1391	0.63	0.20	0.49	0.55
<i>Skipped+NoUpdate</i>	10914	0.25	0.04	0.23	0.24
<i>Intra+NoUpdate</i>	500	0.46	0.07	0.44	0.45
<i>Inter+NoUpdate</i>	73256	0.50	0.08	0.45	0.47
<i>Inter4V+NoUpdate</i>	29913	0.57	0.13	0.52	0.53
<i>Skipped+IntraCAE</i>	201	0.52	0.03	0.51	0.52
<i>Intra+IntraCAE</i>	6997	0.89	0.19	0.81	0.85
<i>Inter+IntraCAE</i>	8583	0.80	0.16	0.74	0.75
<i>Inter4V+IntraCAE</i>	12131	1.02	0.23	0.82	0.97
<i>Skipped+InterCAE</i>	763	0.62	0.03	0.64	0.62
<i>Inter+InterCAE</i>	16251	0.92	0.14	0.88	0.88
<i>Inter4V+InterCAE</i>	11284	1.05	0.20	0.92	1.01

Tabela 20 – Valores estatísticos para o tempo de decodificação dos vários tipos de macroblocos para as sequências utilizadas: média, desvio padrão, moda e mediana.

5.5.5 Determinação da complexidade relativa dos vários tipos de macroblocos de vídeo MPEG-4

O modelo VCV especificado na norma MPEG-4 Visual [3] define, para um determinado perfil@nível, um conjunto de regras e limites para verificar se a capacidade computacional pedida ao decodificador para decodificar um dado conjunto de fluxos binários não excede esses limites. A capacidade computacional é definida em termos do número máximo de macroblocos por segundo que o decodificador pode decodificar, como se viu no capítulo 2. O modelo VCV MPEG-4 contempla duas memórias, uma para os macroblocos fronteira (B-VCV) e outra (VCV) para todos os macroblocos sem distinção, definindo para estas duas memórias a sua capacidade máxima e o ritmo de decodificação (em macroblocos/s). Uma vez que a capacidade e o ritmo de decodificação para a memória VCV são apenas definidos em termos de macroblocos e macroblocos/s sem qualquer distinção entre macroblocos, assume-se implicitamente que o decodificador deverá ser capaz de decodificar qualquer tipo de macroblocos desde que a capacidade e ritmo definidos para o perfil@nível em questão não sejam excedidos, o que em termos de dimensionamento implica que o decodificador deve estar preparado para o pior caso possível, ou seja que os macroblocos em questão sejam todos simultaneamente do tipo mais complexo.

Com o modelo VCV alternativo a propor nesta tese, pretende-se fazer um uso mais eficiente dos recursos de decodificação disponíveis, ou seja aqueles referidos no parágrafo anterior para uma dada combinação perfil@nível, o que significa que não se alteram os recursos de decodificação para cada combinação perfil@nível. Mais precisamente, isto significa que se mantêm as capacidades das memórias e os ritmos de decodificação actualmente definidos na norma para cada perfil@nível. Deste modo, qualquer decodificador em conformidade com a norma MPEG-4 deverá conseguir decodificar os fluxos binários gerados por um codificador que implemente o novo modelo VCV, para um dado perfil@nível.

Atendendo a que, como já se viu, não é verdade que a complexidade de decodificação seja semelhante para todos os tipos de macroblocos de vídeo com excepção dos macroblocos fronteira (o que é manifestamente insuficiente para caracterizar eficazmente a complexidade de uma cena de vídeo), como assume o modelo VCV MPEG-4, o novo modelo VCV pretende melhorar o uso dos recursos de decodificação atribuindo pesos aos vários tipos de macroblocos, reflectindo a sua complexidade efectiva em termos de decodificação. Este objectivo não é trivial, pois como se demonstrou ao longo deste capítulo existe uma grande variação em termos dos tempos de decodificação para cada tipo de macrobloco.

Atendendo a que os tempos absolutos de decodificação variam de plataforma para plataforma de implementação, interessa aqui determinar qual a relação de tempo/complexidade entre os vários tipos de macroblocos, ou seja pesos relativos em relação a uma dada referência. Considerando uma determinada plataforma de implementação do decodificador, esta relação de tempos deve corresponder a uma relação equivalente em termos de complexidade se os tempos de decodificação forem suficientemente representativos da complexidade efectiva, ou seja obtidos através de decodificadores suficientemente optimizados. Espera-se ainda que, mesmo para diferentes plataformas de implementação (optimizadas), os pesos de complexidade sejam suficientemente estáveis para que os valores aqui determinados continuem a ser úteis.

Uma vez que, como se disse acima, se pretendem manter os recursos de descodificação e o modelo VCV MPEG-4 está implicitamente dimensionado para o tipo de macroblocos mais complexo, os pesos a atribuir aos vários tipos de macroblocos devem ser relativos ao tipo de macrobloco mais complexo entre os tipos estudados, ou seja o peso máximo será 1 para esse tipo de macrobloco e todos os outros tipos de macroblocos terão pesos inferiores a 1. Esta solução permite encarar o novo modelo VCV como um modelo onde a complexidade usada é gerida em “regime de trocas”, ou seja onde é possível trocar a descodificação de um macrobloco do tipo mais complexo pela descodificação de dois macroblocos com metade da complexidade e assim sucessivamente.

Os pesos relativos de complexidade para cada tipo de macroblocos podem ser determinados a partir das complexidades (tempos) máximas ou médias atrás determinadas. Analisem-se então os dois casos:

- **Pesos relativos de complexidade determinados a partir das complexidades máximas** – O peso relativo para cada um dos tipos de macroblocos é obtido como a razão entre o tempo máximo de descodificação respectivo (para esse tipo de macrobloco, muito provavelmente para o máximo do número de coeficientes DCT) e o maior tempo máximo de descodificação entre todos os tipos de macroblocos considerados, ou seja o tempo de descodificação para o tipo *Inter4V+InterCAE*, como se pode observar na Tabela 21, que mostra os tempos máximos e médios de descodificação para os vários tipos de macroblocos:

$$Peso[MB] = \frac{t_{\max}[MB]}{t_{\max}[Inter4V + InterCAE]} \quad (1)$$

A adopção da relação de máximos para determinar os pesos relativos de complexidade implica a validade do “regime de trocas” atrás referido para o caso crítico em termos de complexidade, ou seja quando os macroblocos têm a máxima complexidade. Obviamente, esta validade está associada ao facto de o modelo VCV MPEG-4 especificar implicitamente que o descodificador deve ser capaz de descodificar um certo número de macroblocos por segundo do tipo de macrobloco com a maior complexidade, para um certo perfil@nível. Deste modo, é possível implementar um verdadeiro “regime de trocas”, onde se substitui um macrobloco de complexidade máxima por outros menos complexos, existindo a garantia que o fluxo binário é descodificado por um descodificador conforme, ou seja sem aumento dos recursos de descodificação disponíveis.

- **Pesos relativos de complexidade determinados a partir das complexidades médias** – O peso relativo para cada um dos tipos de macroblocos é obtido como a razão entre o tempo médio de descodificação respectivo e o maior tempo médio de descodificação entre todos os tipos de macroblocos considerados, ou seja o tempo de descodificação para o tipo *Inter4V+InterCAE*, como se pode observar na Tabela 21:

$$Peso[MB] = \frac{t_{\text{med}}[MB]}{t_{\text{med}}[Inter4V + InterCAE]} \quad (2)$$

A relação entre os tempos médios de descodificação pode ser maior ou menor que a relação entre os tempos máximos apresentada no ponto anterior, consoante os tipos de macroblocos, como se pode observar na Tabela 21, dependendo da distribuição dos tempos de descodificação para cada tipo de macrobloco. Quando a relação de médias é maior que a relação de máximos, os pesos são mais conservadores que os pesos obtidos através dos tempos máximos, pois faz-se uma “troca” de um macrobloco do tipo mais complexo por menos macroblocos de tipos menos complexos, permitindo que a descodificação se efectue normalmente embora desperdiçando recursos. Porém, na situação contrária, ou seja quando a relação de médias é menor que a relação de máximos (menor peso), pode haver problemas com o “regime de trocas” porque no caso de estar em presença dos macroblocos mais complexos para cada tipo, far-se-ia uma troca excedendo as capacidades máximas de descodificação se o peso da troca fosse determinado pela relação de médias, neste caso mais favorável no sentido de que os macroblocos a trocar são considerados com menor peso do que efectivamente têm. Isto significa que pode acontecer que se subestime a complexidade de uma cena, pois é feita uma “troca” de um macrobloco do tipo mais complexo por mais macroblocos de tipos menos complexos, relativamente ao caso em que se consideram complexidades máximas. Como se pretendem manter os recursos de descodificação, a adopção dos pesos baseados nos tempos médios pode fazer com que os fluxos binários não possam ser descodificados, dado o número exagerado de macroblocos que foram codificados devido às trocas serem tão favoráveis.

Pelo que foi acima referido, conclui-se que o modelo alternativo de complexidade deve basear-se em pesos relativos de complexidade determinados a partir dos tempos de descodificação máximos dos vários tipos de macroblocos, obtendo-se os resultados apresentados na Tabela 21.

	$t_{med}(ms)$	Relação méd.	$t_{max}(ms)$	Relação máx.
Transparente	0.15	0.14	0.21	0.12
<i>Skipped</i> (VO rect.)	0.13	0.12	0.16	0.09
<i>Intra</i> (VO rect.)	0.43	0.41	1.08	0.59
<i>Inter</i> (VO rect.)	0.37	0.35	1.06	0.58
<i>Inter4V</i> (VO rect.)	0.47	0.45	1.21	0.66
<i>Skipped+Opaco</i>	0.20	0.19	0.22	0.12
<i>Intra+Opaco</i>	0.47	0.45	1.06	0.58
<i>Inter+Opaco</i>	0.44	0.42	1.04	0.57
<i>Inter4V+Opaco</i>	0.63	0.60	1.28	0.70
<i>Skipped+NoUpdate</i>	0.25	0.24	0.38	0.21
<i>Intra+NoUpdate</i>	0.46	0.44	0.97	0.53
<i>Inter+NoUpdate</i>	0.50	0.48	1.17	0.64
<i>Inter4V+NoUpdate</i>	0.57	0.54	1.38	0.76
<i>Skipped+IntraCAE</i>	0.52	0.50	0.58	0.32
<i>Intra+IntraCAE</i>	0.89	0.85	1.46	0.80
<i>Inter+IntraCAE</i>	0.80	0.76	1.60	0.88
<i>Inter4V+IntraCAE</i>	1.02	0.97	1.77	0.97
<i>Skipped+InterCAE</i>	0.62	0.59	0.73	0.40
<i>Inter+InterCAE</i>	0.92	0.88	1.73	0.95
<i>Inter4V+InterCAE</i>	1.05	1.00	1.82	1.00

Tabela 21 – Tempos de decodificação médio e máximo para os vários tipos de macroblocos e respectivas relações de tempos.

Existem alguns tipos de macroblocos cujos valores máximos do tempo de decodificação são muito próximos e cujos tempos de decodificação, mesmo fora dos máximos, são sempre muito parecidos. De forma a limitar a complexidade do novo modelo VCV, evitando um número demasiado elevado e desnecessário de classes de complexidade, estes tipos de macroblocos foram agrupados em classes, já que as diferenças de complexidade não são muito grandes. De qualquer forma, a cada classe é atribuído o peso correspondente ao macrobloco mais complexo de todos os tipos de macroblocos que a classe em questão abrange para que o modelo nunca corra o risco de subavaliar a complexidade da cena a codificar.

Tendo em conta os factores acima mencionados, optou-se por agrupar os vários tipos de macroblocos nas classes de complexidade apresentadas na Tabela 22, e que serão utilizadas no novo modelo do VCV a propor no capítulo 6. A cada classe de complexidade foi atribuído um peso relativo que corresponde à relação entre o tempo de decodificação máximo do tipo de macrobloco mais complexo dentro da classe e o tempo de decodificação máximo do tipo de macroblocos mais complexo de todos (*Inter4V+InterCAE*), utilizando a expressão (1). O tempo de decodificação máximo para cada tipo de macroblocos foi retirado da Tabela 21.

Tipos de macroblocos	Peso relativo
<i>Inter4V+InterCAE</i> <i>Inter+InterCAE</i> <i>Inter4V+IntraCAE</i>	1.00
<i>Inter+IntraCAE</i> <i>Intra+IntraCAE</i>	0.88
<i>Inter4V+NoUpdate</i> <i>Inter+NoUpdate</i> <i>Intra+NoUpdate</i>	0.77
<i>Inter4V+Opaco</i> <i>Inter+Opaco</i> <i>Intra+Opaco</i>	0.70
<i>Skipped+InterCAE</i>	0.40
<i>Skipped+IntraCAE</i>	0.32
<i>Skipped+NoUpdate</i>	0.21
<i>Skipped+Opaco</i>	0.12
Transparente	0.12

Tabela 22 – Pesos relativos de complexidade atribuídos às várias classes de macroblocos.

Tanto quanto se sabe, estudos semelhantes de complexidade não estão ainda disponíveis na literatura, representando assim o estudo de complexidade apresentado neste capítulo uma das primeiras abordagens a nível mundial no sentido de estimar a complexidade efectiva de uma cena de vídeo MPEG-4.

5.6 Conclusões

Neste capítulo foi estudada a complexidade relativa de descodificação dos vários tipos de macroblocos usados na norma MPEG-4 Visual. O modelo de complexidade sugerido nesta tese é baseado no número de macroblocos por tipo de codificação (textura e forma), já que este modelo parece levar em conta os principais factores que determinam as diferenças de complexidade entre macroblocos, permitindo modelar a complexidade de uma cena de vídeo de um modo mais eficiente que o modelo especificado na norma MPEG-4 Visual [3].

O tempo de descodificação dos macroblocos que contêm informação de textura varia linearmente com o número de coeficientes da transformada DCT que são descodificados. Em termos de codificação de forma, verificou-se que a complexidade em termos de tempos de descodificação cresce pela seguinte ordem para os vários tipos de macroblocos: transparentes, textura codificada no modo *Skipped*, forma opaca, forma *NoUpdate*, forma *IntraCAE* e forma *InterCAE*. Em termos de codificação de textura, e embora as diferenças sejam pequenas e difíceis de quantificar de um modo preciso, a complexidade aumenta na seguinte ordem: *Intra*, *Inter* e *Inter4V*.

Os pesos obtidos neste capítulo para modelar a complexidade relativa das várias classes de macroblocos são muito importantes no contexto desta tese, já que permitem a obtenção de um novo modelo VCV mais eficiente do que o actualmente implementado na norma MPEG-4, que será descrito no capítulo seguinte. Estes pesos relativos foram determinados com base

nos tempos máximos de decodificação dos vários tipos de macroblocos, utilizando como referência o tipo de macrobloco mais complexo: *Inter4V+InterCAE*.

Capítulo 6

Modelo Alternativo para o VCV

6.1 Introdução

O modelo VCV (*Video Complexity Verifier*) actualmente especificado na norma MPEG-4 Visual [3] apenas faz a distinção entre a complexidade dos macroblocos de fronteira e os restantes macroblocos de vídeo, não distinguindo a complexidade inerente aos vários tipos de macroblocos que não de fronteira. Este facto leva a que macroblocos com vários graus de complexidade de descodificação sejam tratados do mesmo modo pelo modelo, por exemplo macroblocos transparentes e macroblocos opacos. Neste capítulo é proposto um modelo alternativo para o VCV que explora de um modo mais eficiente a variação de complexidade existente entre os vários tipos de macroblocos de vídeo, permitindo que a complexidade de descodificação estimada para os objectos codificados que constituem uma cena se aproxime mais da sua complexidade de descodificação real. Ao longo deste capítulo mostra-se que fluxos binários que seriam considerados não conformes para um dado perfil@nível, usando o modelo MPEG-4, passam a ser conformes utilizando o novo modelo proposto, isto para os mesmos recursos de descodificação. Este facto consubstancia um uso mais eficiente dos recursos disponíveis, propriedade que é sempre desejável.

6.2 Modelo VCV especificado na norma MPEG-4 Visual

O modelo VCV especificado na norma MPEG-4 Visual [3] define um conjunto de regras e limites para verificar se a capacidade computacional (medida em macroblocos por segundo) necessária no decodificador para decodificar um dado conjunto de fluxos binários não excede os valores especificados para o perfil e nível escolhido. Este modelo é aplicado, conjuntamente, a todos os macroblocos de todos os objectos de vídeo que constituem a cena.

O modelo VCV é aplicado aos objectos de vídeo codificados como uma combinação de VOPs dos tipos I, P, B e S (*sprite*). Para objectos do tipo *Still Scalable Texture* é aplicado um modelo VCV separado. O modelo VCV não se aplica a objectos visuais sintéticos, nomeadamente os que utilizam malhas animadas 2D e modelos 3D para animação facial.

6.2.1 Parâmetros do modelo VCV

O modelo VCV consiste em duas memórias virtuais que acumulam o número de macroblocos que são codificados:

- **Memória VCV** – Acumula todos os macroblocos de todas as camadas de todos os objectos presentes na cena.
- **Memória VCV para macroblocos de fronteira, B-VCV** – Acumula apenas os macroblocos de fronteira de todas as camadas de todos os objectos presentes na cena; o B-VCV só está definido para perfis que suportem objectos de vídeo com forma arbitrária.

Os macroblocos de fronteira, ou seja macroblocos que incluem informação de forma e que não são transparentes nem opacos, são contabilizados em ambas as memórias VCV e B-VCV.

O modelo VCV da norma MPEG-4 Visual é definido através de cinco parâmetros [3]:

1. *vcv_buffer_size* – Dimensão da memória VCV.
2. *boundary_vcv_buffer_size* – Dimensão da memória B-VCV.
3. *H* – Ritmo de esvaziamento da memória VCV.
4. *H_B* – Ritmo de esvaziamento da memória B-VCV.
5. *L* – Latência do VCV.

Apresentam-se de seguida as principais características e regras referentes a estes parâmetros.

Dimensões das memórias VCV e ritmos de esvaziamento

A memória VCV é cheia instantaneamente com todos os macroblocos de cada VOP no instante de descodificação do VOP e esvaziada a um ritmo constante à medida que os macroblocos codificados são entregues ao processo de descodificação. O instante de descodificação de um VOP indica o instante em que os dados codificados são removidos da memória VCV para posterior descodificação, sendo calculado pelo codificador a partir do instante de composição. Na norma MPEG-4 Visual [3], o instante em que um VOP deve estar disponível para composição corresponde ao instante de composição somado a um atraso fixo (latência do VCV), como se verá mais adiante. Este atraso indica a latência mínima associada ao processo de descodificação.

O tamanho de cada memória, *vcv_buffer_size* e *boundary_vcv_buffer_size*, define o número máximo de macroblocos que cada memória pode conter em cada instante, ou seja a ocupação máxima das memórias VCV em unidades de macrobloco. A norma MPEG-4 especifica a mesma capacidade máxima para as duas memórias VCV [3].

Os macroblocos são processados pelo decodificador, a partir de cada memória VCV, a um dado ritmo expresso em MB/s, especificado para cada perfil@nível. O ritmo de descodificação da memória VCV, H , especifica o ritmo de esvaziamento da memória VCV, enquanto que o ritmo de descodificação da memória B-VCV, H_B , especifica o ritmo de esvaziamento da memória B-VCV. Estes dois ritmos em conjunto definem o ritmo máximo do processo de descodificação. Para todos os perfis e níveis especificados na norma MPEG-4 Visual [3], o ritmo de descodificação da memória B-VCV, H_B , é metade do ritmo de descodificação da memória VCV, H .

Latência do VCV

A latência do VCV, L , é definida como o tempo necessário para descodificar o conteúdo da memória VCV cheia, e é dada pela seguinte expressão

$$L = \frac{vcv_buffer_size}{H} \quad (3)$$

Este parâmetro impõe a latência mínima que deve ser respeitada no processo de descodificação. Por definição, a latência do modelo VCV é imposta pela memória VCV e não pela memória B-VCV. O ritmo de descodificação da memória B-VCV, H_B , é metade do ritmo de descodificação da memória VCV, H , o que significa que não é possível descodificar o conteúdo da memória B-VCV completa durante o intervalo de tempo L (a memória B-VCV é do mesmo tamanho mas é esvaziada a metade do ritmo). Tudo isto implica que, para cada VOP, o número de macroblocos de fronteira não pode exceder 50% do número total de macroblocos que a memória B-VCV suporta.

6.2.2 Dinâmica de ocupação do VCV

A dinâmica de ocupação do VCV depende directamente do processo de descodificação dos VOPs. Nos instantes de descodificação de cada VOP (atrás definidos), os macroblocos codificados são adicionados às memórias VCV e B-VCV e são posteriormente removidos destas memórias à medida que o processo de descodificação avança. O instante de tempo para o qual um dado VOP fica completamente descodificado depende da quantidade e do tipo de macroblocos que têm de ser descodificados, da ocupação das memórias no instante de descodificação do VOP e da velocidade máxima de descodificação especificada pelos ritmos de descodificação do VCV.

Enchimento da memória VCV

Seja M_i o número total de macroblocos do VOP i e M_{Bi} o número de macroblocos de fronteira do mesmo VOP. As memórias do modelo VCV estão vazias no início da descodificação e são, instantaneamente, ocupadas com macroblocos codificados nos instantes de descodificação de cada VOP à medida que o processo de descodificação avança. Assim, em cada instante de descodificação do VOP i , t_i , são adicionados M_i macroblocos à ocupação da memória VCV, $VCV(t)$, e M_{Bi} macroblocos à ocupação da memória B-VCV, $B-VCV(t)$.

Esvaziamento da memória VCV

A ocupação das memórias decresce linearmente aos ritmos H e H_B , respectivamente para a memória VCV e B-VCV, até a sua ocupação chegar a zero ou até ao próximo instante de

descodificação, t_{next} , sendo t_{next} o instante de descodificação mais próximo depois de t_i para qualquer VOP de qualquer objecto presente na cena. Se as duas memórias VCV ficarem vazias, o modelo VCV fica inactivo e permanece inactivo até ao próximo instante de descodificação de qualquer VOP, t_{next} , tal como exemplificado na Figura 67 para a memória VCV. A dinâmica da ocupação da memória B-VCV é idêntica à da memória VCV.

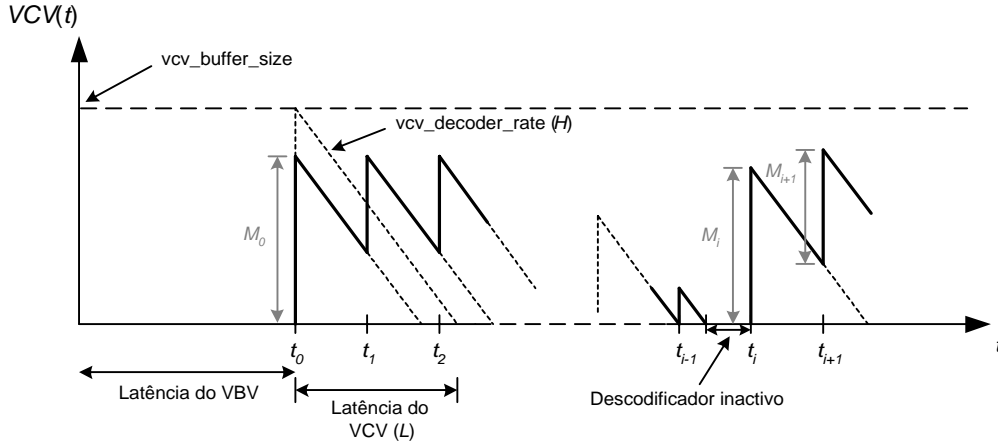


Figura 67 – Dinâmica da ocupação da memória VCV.

Duração da descodificação de um VOP

O intervalo de tempo durante o qual o VOP i está a ser descodificado estende-se entre os instantes s_i e e_i que são definidos pelas seguintes equações

$$\begin{aligned} s_i &= t_i + \max \left[\frac{VCV(t_i)}{H}, \frac{B - VCV(t_i)}{H_B} \right] \\ e_i &= t_i + \max \left[\frac{VCV(t_i) + M_i}{H}, \frac{B - VCV(t_i) + M_{Bi}}{H_B} \right] \end{aligned} \quad (4)$$

onde t_i é o instante de descodificação do VOP i , $VCV(t_i)$ é a ocupação da memória VCV antes de os macroblocos do VOP i , M_i , serem adicionados a $VCV(t)$, H é o ritmo de descodificação do VCV, $B - VCV(t_i)$ é a ocupação da memória B-VCV antes de os macroblocos de fronteira do VOP i , M_{Bi} , serem adicionados a $B - VCV(t)$ e H_B é o ritmo de descodificação do B-VCV.

O VOP só começa efectivamente a ser descodificado quando o VOP anterior já foi descodificado, ou seja quando o conteúdo relativo a esse VOP saiu completamente das duas memórias. Por isso, s_i corresponde ao instante de descodificação do VOP i mais o maior dos tempos que demora a descodificação dos macroblocos do VOP anterior que ainda ocupam as memórias VCV e B-VCV. A descodificação do VOP i termina quando todos os seus macroblocos saíram das memórias VCV, ou seja no instante que resulta somando ao instante de descodificação t_i o maior dos tempos de esvaziamento das duas memórias VCV; cada um destes tempos de esvaziamento corresponde ao tempo de descodificação dos macroblocos do VOP anterior que estavam em cada uma das memórias mais os macroblocos respectivos do VOP i .

6.2.3 Restrições impostas pelo modelo VCV

A conformidade dos fluxos binários em termos do modelo VCV só pode ser garantida se o conjunto de fluxos binários que correspondem a uma cena codificada está de acordo com as restrições impostas pelo modelo VCV, relativamente à ocupação das memórias VCV e à duração da descodificação de cada VOP.

Restrições impostas na ocupação das memórias VCV

Um conjunto de fluxos binários que constituem uma cena está em conformidade em termos do modelo VCV, para uma determinada combinação perfil@nível, se a ocupação das memórias VCV nunca excede a sua capacidade máxima.

Quando as memórias VCV ficam vazias, o descodificador permanece inactivo e as ocupações das memórias VCV, $VCV(t)$ e $B-VCV(t)$, mantêm-se inalteradas durante o período de inactividade. Esta situação é ilustrada na Figura 67, que mostra a ocupação de uma memória VCV, $VCV(t)$, em função do tempo.

Restrições impostas na duração da descodificação de um VOP

Para além de a capacidade das memórias VCV não poder ser excedida, a descodificação de cada VOP i tem de estar terminada até ao instante $\tau_i + L$ (instante de composição mais a latência do VCV). É esta condição que determina que certos recursos computacionais mínimos sejam necessários quando se está a fazer descodificação em tempo real e logo com instantes de composição rigidamente determinados. A latência do VCV é constante para todos os VOPs.

6.3 Abordagens alternativas para o modelo VCV

O controlo da capacidade computacional exigida ao descodificador é uma tarefa essencial que o codificador deve desempenhar para garantir que os limites estabelecidos em termos dessa carga computacional não são ultrapassados. Existem várias abordagens alternativas para modelar a capacidade computacional exigida ao descodificador; essas abordagens diferenciam-se em termos do número de memórias que usam e dos respectivos ritmos de esvaziamento. Esta secção apresenta várias abordagens alternativas para o modelo VCV [25] e compara-as com o modelo VCV especificado pela norma MPEG-4 Visual [3].

6.3.1 Memória única com ritmo de descodificação único

O modelo VCV mais simples que utiliza a abordagem de memória virtual consiste numa única memória que acumula a complexidade dos dados codificados e que esvazia com um único ritmo de descodificação; este ritmo define a velocidade a que o descodificador pode descodificar dados de vídeo MPEG-4. Este modelo assume que a descodificação se faz de acordo com a estratégia *first in, first out* (FIFO). Existem duas variações possíveis para esta abordagem: com e sem pesos relativos atribuídos aos vários tipos de macroblocos.

- **Sem pesos atribuídos aos vários tipos de macroblocos**

Nesta abordagem, todos os macroblocos têm o mesmo peso em termos da complexidade de descodificação. Este modelo é bastante simples de implementar, mas tem como desvantagem o facto de os descodificadores terem de ser desenhados para lidar com o tipo de macroblocos mais pesado computacionalmente, uma vez que não se exclui a possibilidade de todos os macroblocos serem desse tipo. De facto, a não distinção entre tipos de macroblocos leva a que os descodificadores tenham de estar preparados para descodificar cenas que podem ter grandes variações de complexidade ao nível dos macroblocos e das ferramentas de codificação utilizadas, ou seja casos demasiado críticos que dificilmente ocorrerão na realidade.

O tempo que demora a descodificação de um determinado VOP i é dado por

$$td_i = \frac{M_i}{H} \quad (5)$$

onde M_i é o número de macroblocos do VOP i e H é o ritmo de descodificação do VCV.

- **Com pesos relativos atribuídos aos vários tipos de macroblocos**

Esta abordagem é semelhante à anterior, mas os macroblocos são agora divididos em classes, por exemplo de acordo com o seu tipo de codificação da textura e da forma, e cada classe tem associado um peso relativo que expressa a sua complexidade de descodificação face a uma dada referência. Assim, a ocupação da memória VCV para um dado VOP corresponde à soma pesada dos seus macroblocos. Tal como no caso anterior, a memória VCV é esvaziada a um dado ritmo constante. A atribuição de pesos relativos aos vários tipos de macroblocos permite modelar de uma forma mais real a complexidade de descodificação de uma cena. Contudo, a definição dos pesos não é trivial, pois as implementações dos descodificadores podem ser muito distintas, podendo ir da implementação em *software* até *hardware* dedicado.

O tempo que demora um VOP i a ser descodificado é dado por

$$td_i = \frac{\sum_{j=1}^N \alpha_j M_{ij}}{H} \quad (6)$$

onde M_{ij} representa o número de macroblocos do VOP i da classe j , α_j representa o peso correspondente à complexidade de descodificação dessa classe e N o número de classes de macroblocos consideradas.

6.3.2 Memória única com vários ritmos de descodificação

Tal como no caso anterior, esta abordagem assume uma descodificação do tipo FIFO, mas cada classe de macroblocos tem um ritmo de descodificação diferente. Esta abordagem é equivalente à anterior, se se considerar que os pesos dos macroblocos correspondem aos quocientes entre um ritmo de descodificação de referência e os ritmos de descodificação de cada classe.

Sendo H_j o ritmo de descodificação para a classe j , então o tempo que o descodificador demora a descodificar o VOP i é dado por

$$td_i = \sum_{j=1}^N \frac{M_{ij}}{H_j} \quad (7)$$

ou equivalentemente por

$$td_i = \frac{1}{H_k} \left(M_{ki} + \sum_{j=1, j \neq k}^N \frac{H_k}{H_j} M_{ji} \right) = \frac{1}{H} \left(\sum_{j=1}^N \alpha_j M_{ji} \right) \quad (8)$$

onde $H = H_k$ é o ritmo de referência e $\alpha_j = \begin{cases} 1, & j = k \\ \frac{H_k}{H_j}, & j \neq k \end{cases}$.

Esta abordagem é muito semelhante à anterior e produz resultados idênticos. É contudo mais difícil de implementar, pois é mais fácil adicionar à memória VCV o número de macroblocos pesados segundo a sua complexidade do que variar o ritmo de descodificação de acordo com a classe dos macroblocos.

6.3.3 Várias memórias com vários ritmos de descodificação

Esta abordagem para o modelo VCV assume um grau de paralelismo no descodificador que pode existir em descodificadores implementados em *hardware*, com componentes dedicados para vários tipos de macroblocos. Por exemplo, o descodificador pode conter um módulo para descodificar macroblocos sem informação de forma (macroblocos opacos) e outro módulo para lidar com os macroblocos com informação de forma (macroblocos de fronteira).

Neste caso, o modelo VCV consiste em N memórias, uma para cada classe de macroblocos, com os ritmos de descodificação associados H_j . O tempo que demora a descodificação de um determinado VOP é dado por

$$td_i = \max_j \left(\frac{M_{ji}}{H_j} \right). \quad (9)$$

Neste modelo, o tempo de descodificação é determinado pela última memória a ser esvaziada.

Esta abordagem para o modelo VCV faz sentido em descodificadores que descodifiquem as várias classes de macroblocos de um modo paralelo, possuindo ferramentas optimizadas para cada classe. Relativamente aos modelos anteriores que seguem uma abordagem FIFO, este modelo apresenta tempos de descodificação menores, já que a descodificação dos macroblocos é feita em paralelo, ou seja à custa de maiores recursos e logo de maior custo.

6.3.4 Modelo VCV MPEG-4

O modelo VCV especificado na norma MPEG-4 segue uma abordagem de várias memórias com vários ritmos de decodificação. São utilizadas duas memórias: VCV, que acumula todos os macroblocos, e B-VCV, que acumula só os macroblocos de fronteira. Note-se que os macroblocos de fronteira são incluídos em ambas as memórias. A medida da complexidade de decodificação dos macroblocos baseada só no seu tipo de forma (neste caso são distinguidos macroblocos de fronteira dos restantes) reflecte um compromisso entre a simplicidade do modelo e a sua eficiência na caracterização da complexidade da cena.

Para um decodificador ser considerado em conformidade com a norma MPEG-4, para uma dada combinação perfil@nível, deve implementar o mecanismo de verificação de vídeo especificado na norma MPEG-4 Visual [3] e deve também conseguir decodificar todos os VOPs dos fluxos binários de teste definidos na parte 4 da norma MPEG-4 [10], denominada por *MPEG-4: Conformance*, dentro dos instantes de composição de cada VOP, verificando ao mesmo tempo se o mecanismo de verificação de vídeo (VBV, VCV e VMV) não é violado quando os bits são entregues ao decodificador ao débito correcto. O decodificador deve reconstruir os VOPs I, P, B e S (*sprites*) com uma diferença de ± 1 para cada *pixel* em comparação com os VOPs gerados pelo decodificador de referência, definido na parte 5 da norma MPEG-4 [11]. Adicionalmente, a precisão aritmética excluindo a IDCT deve ser idêntica à do decodificador de referência, não considerando a função de deformação de perspectiva usada para decodificar S-VOPs.

Os fluxos binários de teste destinam-se a testar o decodificador em situações críticas dos parâmetros definidos no perfil@nível, por exemplo o débito binário máximo, o número máximo de objectos, a ocupação máxima das memórias VBV, VCV e VMV entre muitos outros¹⁵. Um decodificador só pode reclamar conformidade para uma dada combinação perfil@nível se conseguir decodificar os fluxos binários de teste normativamente definidos, e ainda todos os fluxos binários com quaisquer parâmetros ou opções permitidos por esse perfil@nível. Ao não considerar as diferenças de complexidade existentes entre os vários tipos de macroblocos de vídeo, o modelo VCV especificado na norma MPEG-4 leva a que os decodificadores tenham de ser desenhados para lidar com os casos mais críticos, nomeadamente para cumprir o mecanismo de verificação de vídeo e para conseguir decodificar os VOPs cumprindo os requisitos de tempo exigidos. Isto significa que especificando-se apenas que um dado número de macroblocos por segundo têm de ser decodificados, esses macroblocos têm de poder ser todos simultaneamente do tipo de macroblocos mais complexo.

Para os perfis e níveis definidos na norma MPEG-4 Visual, a memória VCV tem o dobro do ritmo de decodificação da memória B-VCV, como se pode observar na Tabela 23, o que tenta expressar a maior complexidade de decodificação dos macroblocos de fronteira [3]. Esta divisão dos macroblocos em duas classes (macroblocos de fronteira e todos os macroblocos) leva a que macroblocos opacos e transparentes sejam tratados exactamente da mesma forma em termos de complexidade, o que não é uma boa solução como já se mostrou no capítulo anterior.

¹⁵ Ainda que estes sejam os objectivos que determinam a inclusão dos fluxos binários na parte 4 da norma MPEG-4, verifica-se infelizmente que os fluxos binários actualmente incluídos não estão associados a situações suficientemente críticas.

Perfil@Nível	Capacidade VMV (MB)	Capacidade VCV (MB)	Ritmo de decodificação VCV (MB/s)	Ritmo de decodificação B-VCV (MB/s)
<i>Simple@L1</i>	198	99	1485	–
<i>Simple@L2</i>	792	396	5940	–
<i>Simple@L3</i>	792	396	11880	–
<i>Simple Scalable@L1</i>	1782	495	7425	–
<i>Simple Scalable@L2</i>	3168	792	23760	–
<i>Core@L1</i>	594	198	5940	2970
<i>Core@L2</i>	2376	792	23760	11880
<i>N-Bit@L2</i>	2376	792	23760	11880
<i>Main@L2</i>	2376	1188	23760	11880
<i>Main@L3</i>	9720	3240	97200	48600
<i>Main@L4</i>	48960	16320	489600	244800

Tabela 23 – Dimensão das memórias VMV e VCV/B-VCV e respectivos ritmos de decodificação/esvaziamento para cada perfil@nível visual.

Para os perfis que não permitem objectos de vídeo de forma arbitrária, o modelo VCV torna-se um modelo de uma memória com um único ritmo de decodificação, ou seja não existe diferença entre a complexidade de decodificação dos macroblocos. Nos perfis que suportam objectos de forma arbitrária, o B-VCV impõe limites ao número de macroblocos codificados com forma arbitrária que podem ser decodificados para um dado perfil@nível.

O ritmo de decodificação especificado na norma MPEG-4 para a memória VCV é o dobro do ritmo de decodificação da memória B-VCV, ou seja $H = 2 \times H_B$. Este facto determina que o tempo de decodificação de um VOP é determinado pela memória VCV se a percentagem do número de macroblocos de fronteira no número total de macroblocos for inferior a 50% e pela memória B-VCV no caso contrário. Deste modo, o tempo de decodificação de um VOP i é dado pela expressão

$$td_i = \begin{cases} \frac{M_{Bi}}{H_B}, & M_{Bi} \geq \frac{1}{2} M_i \\ \frac{M_i}{H}, & M_{Bi} < \frac{1}{2} M_i \end{cases} \quad (10)$$

onde M_{Bi} e M_i representam o número de macroblocos de fronteira e o número total de macroblocos no VOP i , respectivamente.

É importante notar que no modelo VCV da norma MPEG-4 Visual o mesmo número de macroblocos presentes numa cena podem conduzir a tempos de decodificação diferentes, dependendo da sua distribuição pelos objectos de vídeo presentes na cena. Por exemplo, um único VOP com 198 macroblocos (99 MB de fronteira + 99 MB não fronteira) demora aproximadamente 33 ms a ser decodificado no nível 1 do perfil *Core*:

$$td = \max\left(\frac{198}{5940}, \frac{99}{2970}\right) \approx 33 \text{ ms}.$$

O mesmo número de macroblocos organizados em dois VOPs (VOP₁: 69 MB não fronteira + 30 MB de fronteira, VOP₂: 30 MB não fronteira + 69 MB de fronteira) que

pertencam a dois objectos de vídeo diferentes demoram, aproximadamente, 40 ms a serem decodificados, para o mesmo perfil e nível:

$$td = \max\left(\frac{99}{5940}, \frac{30}{2970}\right) + \max\left(\frac{99}{5940}, \frac{69}{2970}\right) \approx 40 \text{ ms}.$$

Esta diferença de tempos é uma consequência directa da equação (4), que impõe que a decodificação de um dado VOP só pode começar depois de o VOP anterior ter sido completamente removido das duas memórias VCV, mesmo que uma delas fique, entretanto, completamente vazia.

Uma das principais limitações do modelo VCV especificado na norma MPEG-4 é o facto de os macroblocos transparentes não serem distinguidos dos macroblocos opacos e de fronteira, mesmo não tendo informação de textura e forma a processar. Isto leva muitas vezes a uma forte sobre-estimação da complexidade efectiva da cena, não sendo por isso possível codificá-la de um modo conforme de acordo com um certo perfil@nível para o qual era esperado que essa cena pudesse ser codificada, em comparação com cenas semelhantes em termos de complexidade e que podem ser codificadas nesse perfil@nível. Este facto é particularmente relevante para objectos com *bounding boxes* constituídas por uma grande percentagem de macroblocos transparentes (Figura 68) ou para cenas constituídas por vários objectos de vídeo cujas *bounding boxes* se sobrepõem (Figura 69), contribuindo então alguns macroblocos mais do que uma vez (mesmo só como transparentes) para a ocupação do VCV e do VMV.



Figura 68 – Sequência Container: objecto composto principalmente por macroblocos transparentes (82%).



Figura 69 – Sequência News: os macroblocos dentro dos rectângulos são contados três vezes para o VCV e VMV, duas como transparentes e uma como opacos ou de fronteira.

A Figura 70 mostra a percentagem do número de macroblocos transparentes, opacos e de fronteira e a ocupação das memórias VBV, VCV, B-VCV e VMV em percentagem dos valores máximos permitidos para a sequência *News*, composta por 4 objectos de vídeo, codificada em formato QCIF, a 15 tramas por segundo, no nível 1 do perfil *Core*, a 384 kbit/s. A Figura 71 mostra os mesmos parâmetros para a sequência *Coastguard* composta por 4

objectos de vídeo, codificada em formato QCIF, a 30 tramas por segundo, no nível 1 do perfil *Core*, a 384 kbit/s.

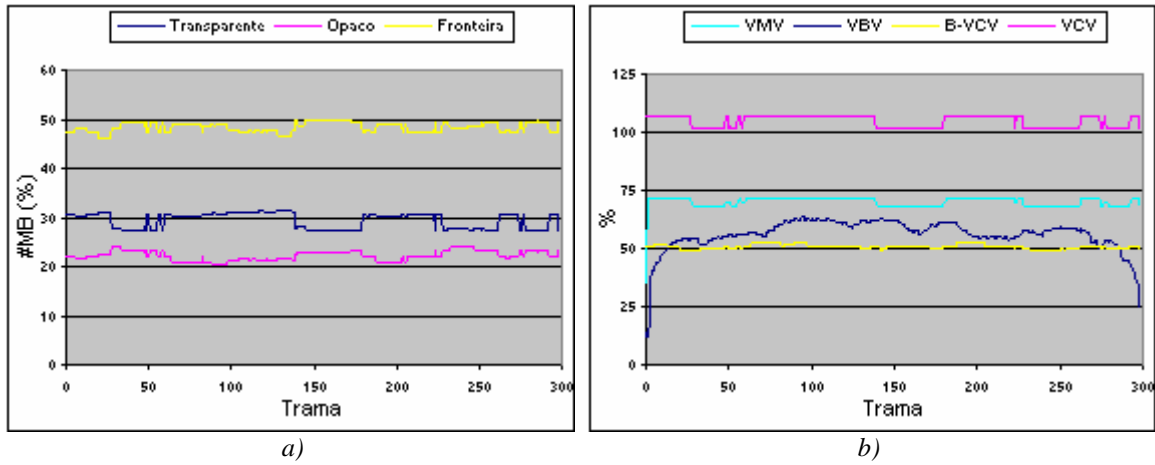


Figura 70 – Sequência News: a) número de macroblocos; b) ocupação das memórias.

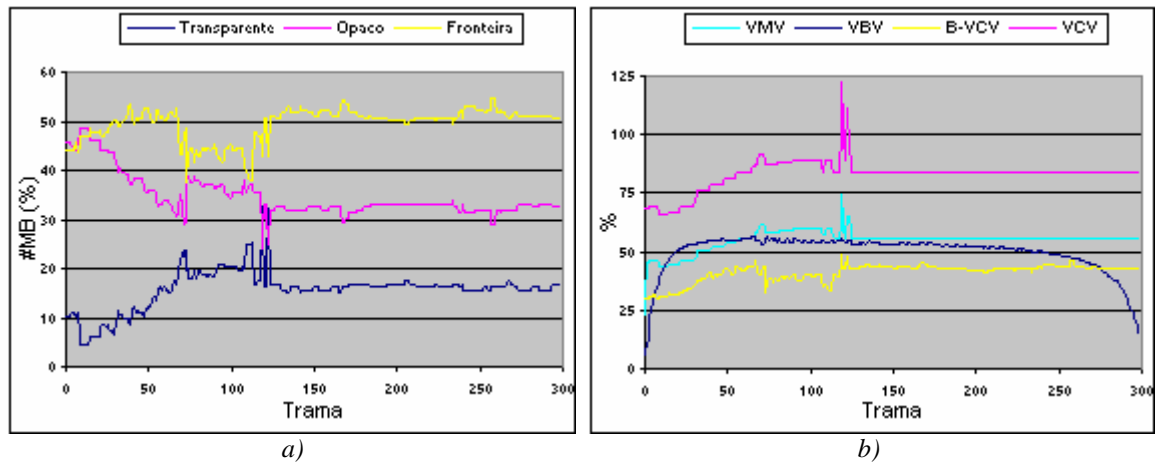


Figura 71 – Sequência Coastguard: a) número de macroblocos; b) ocupação das memórias.

Como se pode observar nos gráficos anteriores, em ambos os casos, mas de forma mais evidente no segundo caso, os picos de ocorrência de macroblocos transparentes correspondem aos picos da ocupação do VCV. Estes exemplos mostram a grande influência que os macroblocos transparentes têm no modelo VCV especificado na norma MPEG-4. Na sequência dos resultados da avaliação da complexidade relativa dos vários tipos de macroblocos feita no capítulo 5, pensa-se que esta influência é incorrecta e exagerada, uma vez que não corresponde a qualquer uso efectivo de recursos computacionais mas sim ao seu desperdício, em virtude da sobreavaliação da complexidade dos macroblocos transparentes impedir a codificação e correspondente descodificação de cenas de vídeo para as quais existiriam efectivamente suficientes recursos de descodificação.

6.4 Modelo VCV IST: uma solução mais eficiente

Conforme se tem vindo a referir ao longo desta tese, o seu principal objectivo é a proposta de um modelo alternativo para o VCV, denominado por modelo VCV IST, que não apresente as principais limitações identificadas no modelo VCV MPEG-4.

O modelo VCV IST proposto baseia-se nos seguintes princípios:

- **Modelo de complexidade baseado no tipo de codificação dos macroblocos** – O modelo proposto faz a distinção da complexidade entre os vários macroblocos com base no tipo de codificação da textura e da forma do macrobloco. Este modelo leva em conta os factores que efectivamente determinam as diferenças de complexidade entre os vários tipos de macroblocos.
- **Memória única com pesos relativos atribuídos às várias classes de macroblocos** – A memória única acumula todos os macroblocos codificados, sendo cada macrobloco pesado de acordo com a classe de complexidade a que pertence. Deste modo, a ocupação da memória VCV corresponde à soma pesada dos macroblocos que são codificados. Esta solução revela-se adequada para modelar de um modo eficaz, e relativamente simples (por exemplo, em relação à solução com várias memórias), a complexidade de uma cena, já que permite distinguir e ponderar em termos de complexidade as várias classes de macroblocos. A utilização de várias memórias, uma para cada classe de macroblocos, poderia ser útil se o descodificador funcionasse com algum grau de paralelismo, o que não acontece com o descodificador MoMuSys/IST utilizado nesta tese, não acontece com os dois codecs de referência da norma MPEG-4 Visual incluídos na parte 5 da norma MPEG-4 [11], nem deverá acontecer com muitos (ou até mesmo a maioria) dos descodificadores MPEG-4 vídeo a aparecer no mercado.
- **Ritmo de descodificação único** – A utilização de uma memória única com pesos atribuídos às várias classes de macroblocos implica um ritmo de descodificação único. A opção por ritmos de descodificação diferentes para as várias classes de macroblocos só faz sentido quando não se utilizam pesos, já que os ritmos de descodificação diferentes são destinados precisamente a diferenciar as classes de macroblocos. O ritmo de descodificação utilizado no modelo VCV IST é igual ao ritmo de descodificação da memória VCV do modelo VCV MPEG-4. Assim é possível realizar de um modo simples a comparação entre os dois modelos, uma vez que se mantêm constantes os recursos computacionais disponíveis.

O modelo VCV IST permite distinguir de um modo mais preciso do que o modelo VCV MPEG-4 as diferenças de complexidade existentes entre os vários tipos de macroblocos e permite modelar mais eficazmente a complexidade das cenas codificadas.

Assim, o modelo VCV IST modela o número de macroblocos de um VOP i , M_i , que são adicionados à memória VCV no instante de descodificação t_i através da seguinte expressão:

$$M_i = \sum_{j=1}^9 k_j \times Mc_j \quad (11)$$

onde k_j é o peso associado à classe j e Mc_j é o número de macroblocos do VOP i pertencentes à classe j .

O tempo que demora a descodificação de um VOP i é obtido a partir da expressão

$$td_i = \frac{M_i}{H} \quad (12)$$

onde M_i representa o número de macroblocos do VOP i , obtido através da expressão (11), e H é o ritmo de descodificação do VCV.

O instante inicial, s_i , e final, e_i , da descodificação de um VOP i são agora dados pelas equações

$$\begin{aligned} s_i &= t_i + \frac{VCV(t_i)}{H} \\ e_i &= t_i + \frac{VCV(t_i) + M_i}{H} \end{aligned} \quad (13)$$

onde t_i é o instante de descodificação do VOP i , $VCV(t_i)$ é a ocupação da memória VCV antes de os macroblocos do VOP i , M_i , serem adicionados a $VCV(t)$ e H é o ritmo de descodificação do VCV.

Atendendo a que as vantagens do modelo VCV IST estão fortemente dependentes da disponibilidade de pesos de complexidade credíveis para os vários tipos de macroblocos relevantes é essencial que um estudo exaustivo da complexidade dos macroblocos tenha sido realizado de forma a obter estes pesos. Os pesos relativos de complexidade a utilizar neste capítulo serão aqueles que foram obtidos no capítulo 5 onde foi cuidadosamente analisada a complexidade de descodificação dos vários tipos de macroblocos, tendo-se chegado aos pesos apresentados na Tabela 24 por ordem decrescente de complexidade.

Classe de complexidade para os macroblocos (C_j)	Tipos de macroblocos incluídos na classe de complexidade	Peso (k_j)
C_1	<i>Inter4V+InterCAE</i> <i>Inter+InterCAE</i> <i>Inter4V+IntraCAE</i>	1.00
C_2	<i>Inter+IntraCAE</i> <i>Intra+IntraCAE</i>	0.88
C_3	<i>Inter4V+NoUpdate</i> <i>Inter+NoUpdate</i> <i>Intra+NoUpdate</i>	0.77
C_4	<i>Inter4V+Opaco</i> <i>Inter+Opaco</i> <i>Intra+Opaco</i>	0.70
C_5	<i>Skipped+InterCAE</i>	0.40
C_6	<i>Skipped+IntraCAE</i>	0.32
C_7	<i>Skipped+NoUpdate</i>	0.21
C_8	<i>Skipped+Opaco</i>	0.12
C_9	Transparente	0.12

Tabela 24 – Pesos relativos de complexidade atribuídos aos vários tipos de macroblocos.

O ritmo de descodificação do modelo VCV IST mantém-se igual ao ritmo de descodificação da memória VCV especificado na norma MPEG-4 Visual para cada perfil@nível, apresentado na Tabela 23. Deste modo pode ser feita uma comparação directa entre os dois modelos, uma vez que se mantêm os recursos computacionais no descodificador, alterando-se apenas a forma de avaliar a complexidade dos macroblocos e logo a eficiência do uso desses recursos. Esta vantagem é muito importante, nomeadamente para aplicações onde os recursos computacionais são escassos e caros, como as aplicações móveis, o que exige que estes sejam bem usados.

6.4.1 Comparação entre os modelos VCV IST e VCV MPEG-4

O modo ideal de comparar os modelos VCV MPEG-4 e VCV IST seria fazendo a codificação da mesma cena, para um dado perfil@nível, utilizando ambos os modelos, ou seja permitindo que o número e tipo de macroblocos codificados fossem diferentes segundo os dois modelos envolvidos (com maior número de macroblocos codificados para o modelo VCV IST desde que os recursos de descodificação não sejam excedidos), seguida da descodificação por um descodificador MPEG-4 conforme com o perfil@nível em questão. Se ambas as cenas codificadas fossem descodificadas nos tempos adequados pelo descodificador conforme, isso comprovaria a validade do modelo VCV IST, ou seja que o modelo VCV IST permite a codificação de mais macroblocos, por fazer uma estimativa mais correcta da sua complexidade, isto sem alterar os recursos de descodificação disponíveis. Para que esta comparação e confirmação pudesse ser realizada, seria necessário dispor de um descodificador conforme para um dado conjunto de combinações perfil@nível, a funcionar em tempo real, e que permitisse comprovar efectivamente que as cenas codificadas com ambos os modelos poderiam ser correctamente descodificadas pelo mesmo descodificador para um determinado perfil@nível.

Como o descodificador utilizado nesta tese não funciona em tempo real, não pode ser utilizado como se descreve acima para comparar as complexidades efectivas de duas codificações diferentes de uma cena, nomeadamente em termos do número ou do tipo de macroblocos utilizados, neste caso devido ao uso dos dois modelos de complexidade em comparação: o modelo VCV MPEG-4 e o modelo VCV IST.

Para fazer a comparação possível entre os dois modelos de complexidade, utiliza-se em alternativa um codificador com o controlo de débito activado unicamente para o VBV. O mecanismo de retroacção que impede a violação do VCV e VMV é desactivado para que se possa observar a evolução da ocupação destas memórias sem a influência do controlo de débito em termos de VCV e VMV; neste caso, os fluxos binários codificados são iguais para os dois modelos mas a sua contabilização em termos de VCV é diferente e feita de acordo com o modelo em questão. Assim, sempre que o codificador detecta uma violação dos modelos VCV e VMV, o fluxo binário é assinalado como não conforme mas a codificação continua, o que permite que a ocupação das memórias VCV e VMV exceda os 100% (isso não acontece quando se estão a gerar fluxos binários efectivamente conformes). Este tipo de codificação permitirá verificar que, em muitas situações que se esperaria poder codificar numa dada combinação perfil@nível, enquanto o modelo VCV MPEG-4 excede os 100% de ocupação das memórias VCV, o modelo VCV IST não atinge esse limite. Este facto significa

que o modelo VCV IST permite codificar de modo conforme para um dado perfil@nível cenas de vídeo que o modelo VCV MPEG-4 não consegue codificar em condições semelhantes (por exemplo, sem “saltar” VOPs); esta conclusão é válida no pressuposto de que os pesos de complexidade determinados no capítulo 5 são válidos ou, no mínimo, conservadores para o decodificador em questão.

De seguida, apresenta-se a comparação entre os modelos VCV em estudo, mostrando-se a ocupação das várias memórias do mecanismo de verificação de vídeo durante a codificação de um conjunto representativo de cenas de vídeo. Em cada gráfico é mostrada a ocupação das memórias VMV, VBV, B-VCV, VCV para os modelos VCV MPEG-4 e VCV IST.

6.4.1.1 Comparação entre os modelos VCV: cenas com um único objecto de vídeo rectangular

Nesta secção faz-se a comparação entre o modelo VCV MPEG-4 e o modelo VCV IST em cenas constituídas por um único objecto de vídeo rectangular.

A Figura 72 mostra a ocupação das memórias para a sequência *Akiyo* rectangular (só um objecto de vídeo), com formato QCIF e a uma frequência de trama de 30 tramas/s, codificada no nível 1 do perfil *Simple*, a 64 kbit/s. Como se pode observar, o VCV MPEG-4 é largamente excedido logo a partir da primeira trama, deixando de se ver a curva correspondente (não se escolheu uma escala que permitisse a visualização desta curva porque cresce linearmente, atingindo a memória VCV MPEG-4 a ocupação de 14743% para a trama 300), o que tornaria este fluxo binário codificado não conforme com a norma MPEG-4 para o perfil@nível em questão. Para o VCV MPEG-4 não ser excedido, a sequência teria de ser codificada no nível 2 do mesmo perfil (ou seja contando com mais recursos de codificação) ou então com uma frequência de trama de 15 tramas por segundo, que é a frequência máxima a que um objecto de dimensão QCIF (99 MBs) pode ser codificado em *Simple@L1* em virtude do número máximo de macroblocos por segundo para esta combinação ser precisamente 1485 MB/s ($1485/99 = 15$ tramas/s). Recorrendo ao modelo VCV IST, a mesma sequência poderia ser codificada de um modo conforme ao mesmo perfil *Simple@L1*, já que a ocupação do VCV ronda os 25% durante o processo de codificação. O termo “conforme” significa aqui que ainda que mantendo os recursos de decodificação existentes no decodificador e especificados pela norma (decodificador conforme), o fluxo binário poderia ser decodificado dentro dos limites de tempo necessários por não ser realmente mais complexo do que outros fluxos aceites como conformes pela norma MPEG-4. A decisão de “não conformidade” em termos do modelo VCV MPEG-4 é devida à sobreavaliação da complexidade de alguns macroblocos, especialmente os do tipo *Skipped* para esta sequência, o que impede a melhor utilização dos recursos de decodificação disponíveis ao declarar-se que fluxos que poderão ser decodificados não são conformes.

Como é evidente, estas conclusões pressupõem a validade dos pesos relativos de complexidade determinados no capítulo 5 e usados no modelo de complexidade proposto neste capítulo. Ainda que esses pesos possam sofrer ligeiras alterações consoante a implementação em questão, salienta-se que a sobreavaliação associada ao modelo VCV MPEG-4 é muito expressiva para alguns casos mesmo tendo sido os pesos usados no modelo VCV IST determinados de forma conservadora, ou seja utilizando-se uma relação de máximos.

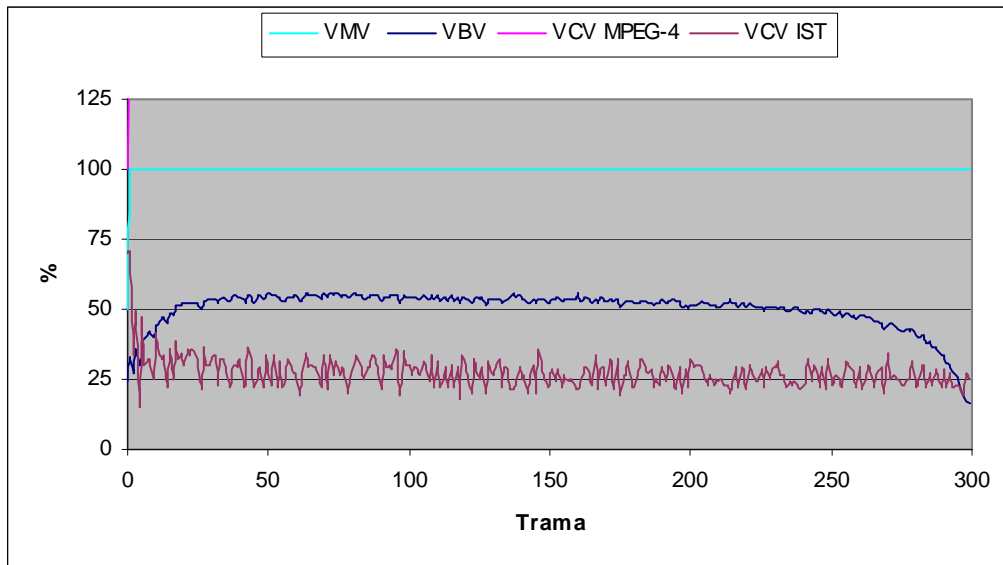


Figura 72 – Evolução da ocupação das várias memórias: sequência Akiyo, rectangular, QCIF a 30 tramas/s, para Simple@L1 a 64 kbit/s.

A Figura 73 mostra a ocupação das memórias do mecanismo de verificação de vídeo para a sequência *Stefan*, rectangular, com formato CIF e uma frequência de trama de 30 tramas/s, codificada no nível 3 do perfil *Simple*, a 384 kbit/s. Durante o processo de codificação, a ocupação do VCV MPEG-4 permanece nos 100%, tal como acontece com todas as sequências rectangulares CIF com frequência de trama 30 tramas/s codificadas neste perfil@nível. Como a ocupação não excede a dimensão de nenhuma das memórias, o fluxo binário resultante da codificação está em conformidade com a norma MPEG-4, mas a curva para o modelo VCV IST mostra que a complexidade efectiva desta sequência é consideravelmente menor do que a expressa através do modelo VCV MPEG-4, devido ao menor peso atribuído aos macroblocos opacos e principalmente aos macroblocos *Skipped*, que representam uma percentagem de 19% do número total de macroblocos.

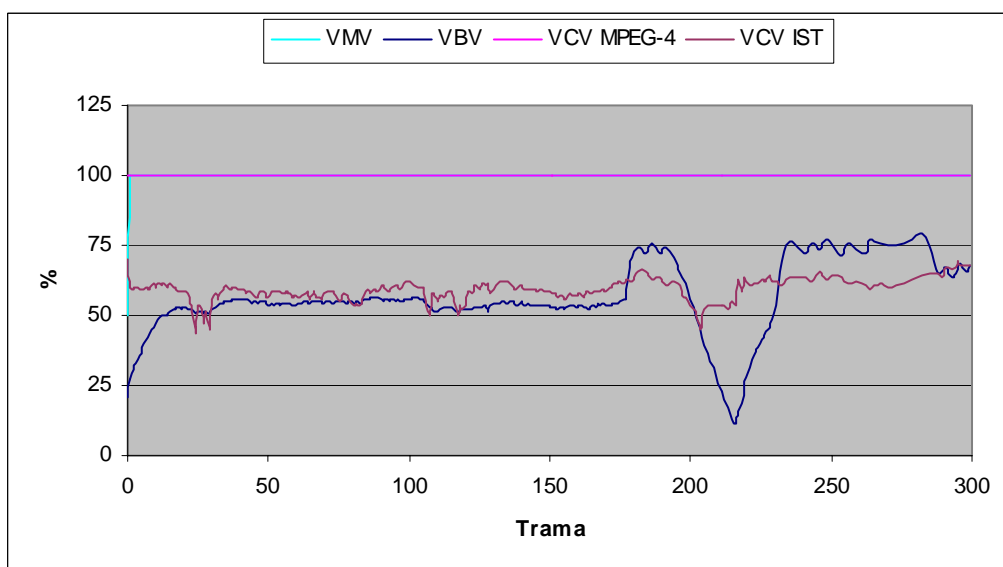


Figura 73 – Evolução da ocupação das várias memórias: sequência Stefan, rectangular, CIF a 30 tramas/s para Simple@L3 a 384 kbit/s.

6.4.1.2 Comparação entre os modelos VCV: cenas com objectos de vídeo de forma arbitrária

De seguida faz-se a comparação entre o modelo VCV MPEG-4 e o modelo VCV IST em cenas constituídas por objectos de vídeo de forma arbitrária.

A Figura 74 mostra a ocupação das memórias para a sequência *Coastguard*, composta por quatro objectos de vídeo, com formato QCIF e frequência de trama 30 tramas/s, codificada para o nível 1 do perfil *Core*, a 384 kbit/s. Como se pode observar na figura, a memória VCV MPEG-4 disponível é excedida, ou seja o fluxo binário resultante não está em conformidade com a norma. A utilização do modelo VCV IST mostra que a cena não é complexa demais para ser codificada com a combinação perfil@nível em questão, pois a ocupação da memória VCV IST ronda os 60% durante a maior parte da codificação. O pico que se observa no gráfico do VCV MPEG-4 é devido ao grande número de macroblocos transparentes existentes nos VOPs em questão para esta sequência. O modelo VCV IST atribui um peso computacional muito mais baixo aos macroblocos transparentes, levando a um pico mais atenuado e a não exceder a capacidade da memória VCV, como se pode observar na Figura 74.

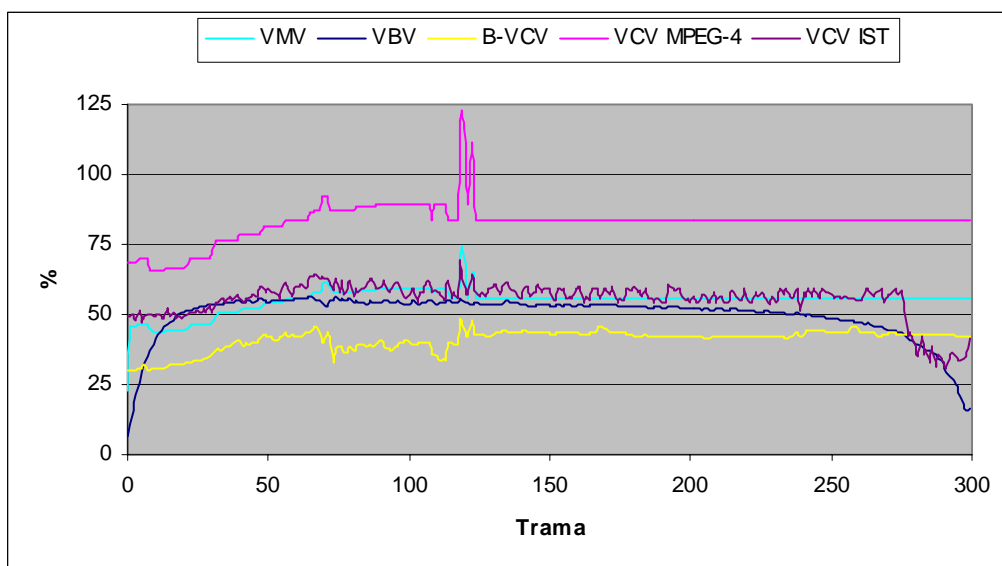


Figura 74 – Evolução da ocupação das várias memórias: sequência *Coastguard*, 4 VOs, QCIF a 30 tramas/s para *Core@L1* a 384 kbit/s.

A Figura 75 mostra a sequência *Akiyo_and_Coastguard* obtida através da aplicação desenvolvida nesta tese e apresentada no capítulo 3. A sequência é composta por quatro objectos de vídeo: “apresentadora” (retirado da sequência *Weather*), “logótipo MPEG-4” (retirado da sequência *Children*), “margem” (retirado da sequência *Coastguard*) e “rio” com dois barcos (retirado da sequência *Coastguard*). A sequência está no formato QCIF, com frequência de trama de 30 tramas/s e foi codificada no nível 1 do perfil *Core* a 384 kbit/s.



Figura 75 – Sequência Akiyo_and_Coastguard, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.

Apesar de também possuir quatro objectos de vídeo, esta cena é mais complexa do que a anterior (*Coastguard*) pois existe uma grande sobreposição entre os vários objectos (o que não acontecia na sequência *Coastguard*) e as suas *bounding boxes*, o que se vai reflectir na ocupação da memória VCV. A Figura 76 mostra a ocupação das memórias para a sequência *Akiyo_and_Coastguard*, onde se pode ver que a capacidade da memória VCV MPEG-4 é claramente excedida a partir da trama 30, o que tornaria o conjunto dos fluxos binários não conforme. Na mesma figura pode-se observar que a ocupação da memória VCV IST se mantém sempre abaixo dos 75%, o que mostra que a complexidade efectiva desta cena é suficientemente baixa para que possa ser codificada a 30 tramas/s no nível 1 do perfil *Core*.

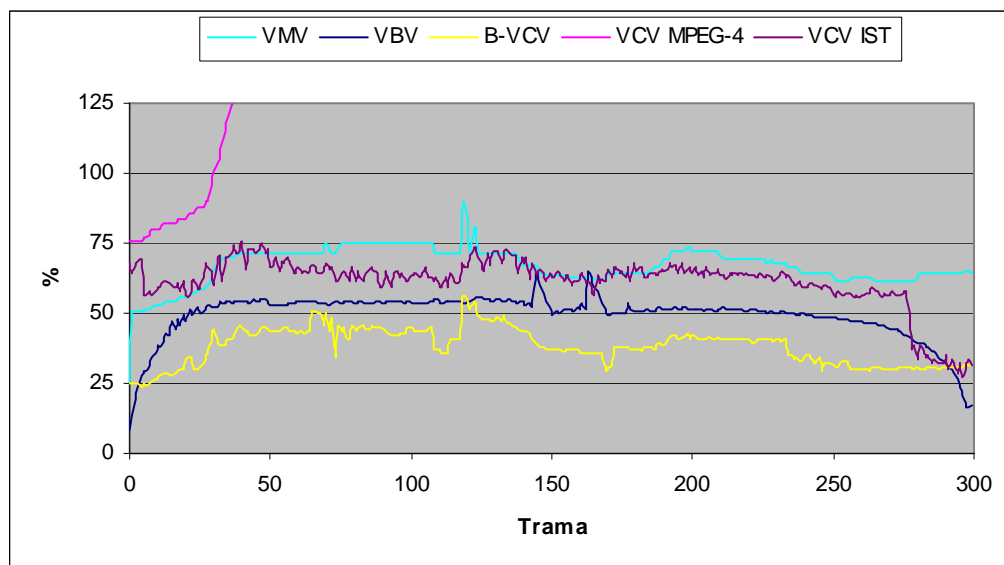


Figura 76 – Evolução da ocupação das várias memórias: sequência Akiyo_and_Coastguard, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.

O número de macroblocos transparentes presentes numa cena influencia determinantemente as consequências do uso do modelo VCV MPEG-4. A Figura 77 mostra uma imagem da sequência *Children_and_Flag*, criada usando a aplicação desenvolvida nesta tese, e que é composta por três objectos de vídeo: “crianças e bola” (retirado da sequência *Children*), “logótipo MPEG-4” (retirado da sequência *Children*) e “mastro” (retirado da sequência *Container*). A sequência está no formato QCIF, a 30 tramas/s e foi codificada no nível 1 do perfil *Core* a 384 kbit/s. As *bounding boxes* dos objectos “crianças e bola” e “mastro” são muito grandes e possuem muitos macroblocos transparentes, o que é crítico para o modelo VCV MPEG-4; para além disso, estas *bounding boxes* variam pouco ao longo de

toda a cena e estão sempre sobrepostas, como se pode observar na Figura 77, o que contribui para uma grande ocupação do VCV.



Figura 77 – Sequência *Children_and_Flag*, 3 VOs, QCIF a 30 tramas/s para *Core@L1* a 384 kbit/s, com as *bounding boxes* representadas para cada objecto.

Apesar de esta sequência ter apenas 3 objectos de vídeo e de grande parte dos macroblocos serem transparentes (58% de macroblocos transparentes, 41% de macroblocos de fronteira e 1% de macroblocos opacos para QCIF), não é possível codificá-la no nível 1 do perfil *Core* utilizando o modelo VCV MPEG-4, como se pode observar na Figura 78. A ocupação da memória VCV MPEG-4 é sempre bastante alta, devido à dimensão e sobreposição das *bounding boxes* dos objectos “crianças e bola” e “mastro”. Quando o “logótipo” surge na cena, a ocupação do VCV MPEG-4 aumenta bastante, tornando o conjunto dos fluxos binários não conforme pois a ocupação do VCV ultrapassa os 100%. A ocupação do VCV MPEG-4 só diminui quando o logótipo fica mais pequeno (com menos macroblocos) e estabiliza finalmente quando o logótipo fica estático no canto superior esquerdo da cena, pois as dimensões das *bounding boxes* dos vários objectos não variam muito. Na mesma figura pode-se verificar que a ocupação da memória VCV IST está sempre abaixo dos 50%, pois é atribuído um peso computacional mais baixo aos macroblocos transparentes, o que mostra que esta cena tem efectivamente uma complexidade pequena e pode ser codificada em *Core@L1*.

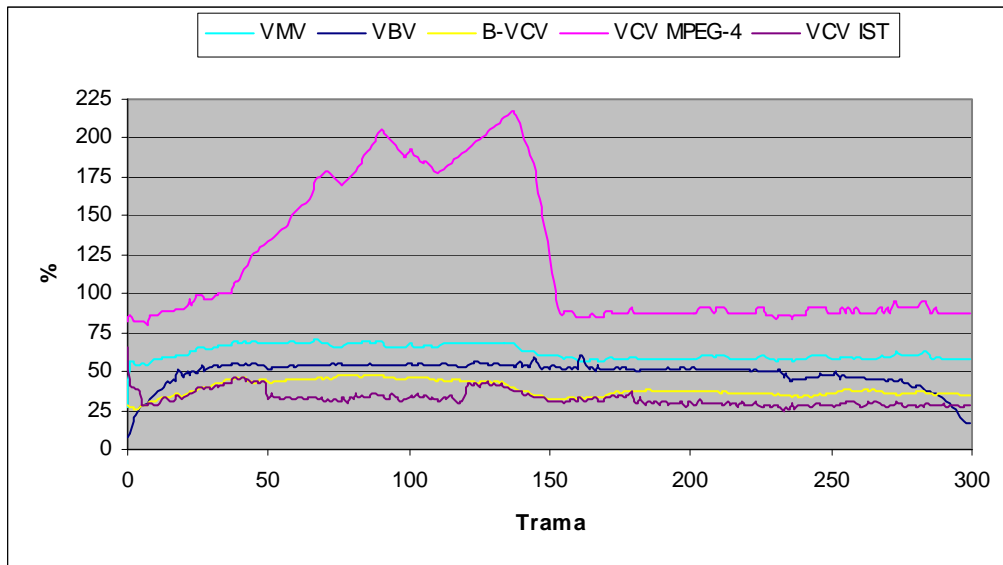


Figura 78 – Evolução da ocupação das várias memórias: sequência *Children_and_Flag*, 3 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.

A Figura 79 e a Figura 80 mostram a ocupação das memórias do mecanismo de verificação de vídeo para a sequência *News*, composta por quatro objectos de vídeo, em formato QCIF, codificada no nível 1 do perfil *Core*, usando frequências de trama de 15 tramas/s (384 kbit/s) e 30 tramas/s (384 kbit/s), respectivamente.

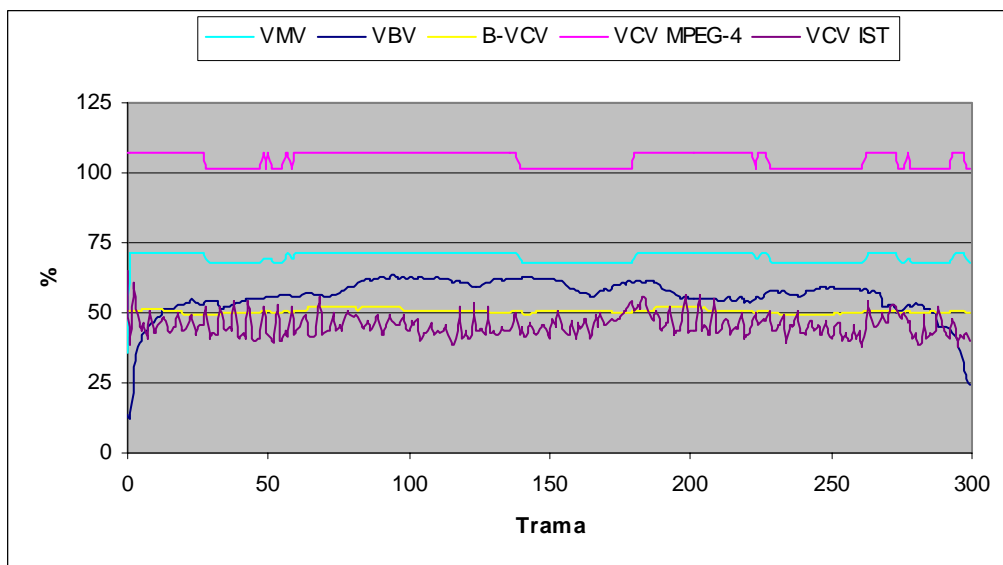


Figura 79 – Evolução da ocupação das várias memórias: sequência *News*, 4 VOs, QCIF a 15 tramas/s para Core@L1 a 384 kbit/s.

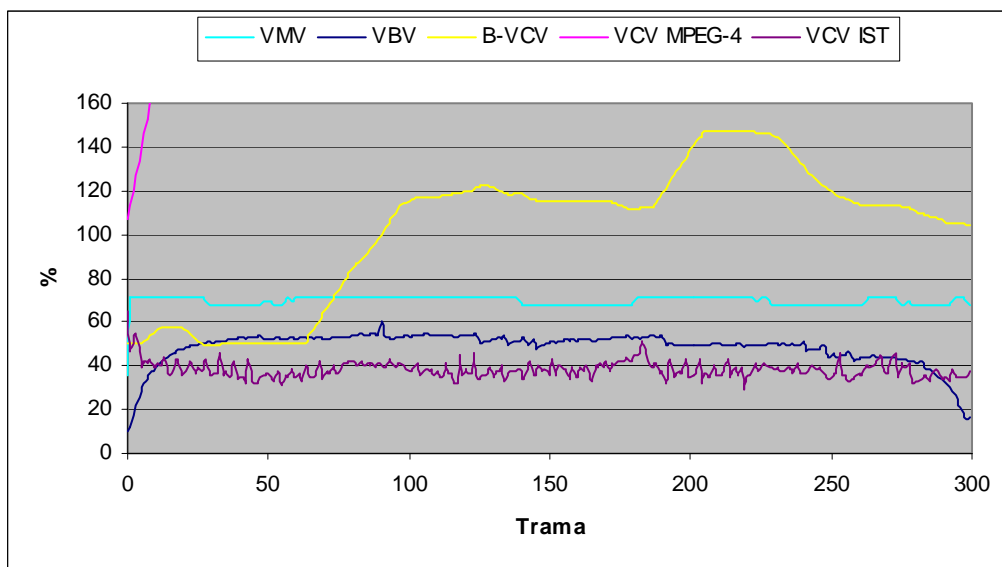


Figura 80 – Evolução da ocupação várias memórias: sequência *News*, 4 VOs, QCIF a 30 tramas/s para Core@L1 a 384 kbit/s.

Como se pode observar nas figuras acima, a memória VCV MPEG-4 é claramente excedida para ambos os casos, sendo que a 30 tramas/s a curva do VCV MPEG-4 aumenta sempre (deixando de ser visível no gráfico) e também a memória B-VCV é excedida, dado o grande número de macroblocos de fronteira que esta sequência contém (30% de macroblocos transparentes, 48% de macroblocos de fronteira e 22% de macroblocos opacos para QCIF). Utilizando o modelo VCV MPEG-4, estas cenas nunca poderiam ser codificadas no nível 1 do perfil *Core*, a não ser que se aceitassem “saltos de tramas” (frequência temporal variável). Note-se que o nível 1 do perfil *Core* suporta um máximo de quatro objectos de vídeo numa cena com dimensão típica QCIF, mas para isso ser possível os objectos não podem ter muitos macroblocos e as suas *bounding boxes* devem sobrepor-se o mínimo possível (para evitar mais macroblocos transparentes), senão existe o risco de o VCV ser excedido, como acontece nos casos anteriores. O modelo VCV IST mostra mais uma vez que é perfeitamente possível codificar estas cenas no nível 1 do perfil *Core*, ou seja a sua complexidade é sobre-estimada com o modelo VCV MPEG-4.

A mesma situação ocorre quando se codifica a sequência *News*, em formato CIF, no nível 2 do perfil *Core* a 2000 kbit/s (Figura 81). A capacidade da memória VCV MPEG-4 é claramente excedida durante o processo de codificação. Por outro lado, a ocupação da memória VCV recorrendo ao modelo alternativo VCV IST ronda os 40% durante a codificação, mostrando que se poderia codificar esta cena no perfil *Core@L2*. A influência dos macroblocos transparentes no modelo VCV MPEG-4 é novamente facilmente perceptível neste exemplo. A Figura 82 mostra a evolução do número de macroblocos transparentes, opacos e de fronteira. O número de macroblocos opacos e de fronteira mantém-se, aproximadamente, estável ao longo da sequência, existindo uma oscilação entre dois valores para o número de macroblocos transparentes. Como se pode observar nas figuras, sempre que o número de macroblocos transparentes aumenta existe um correspondente aumento da ocupação da memória VCV MPEG-4 e quando o número de macroblocos transparentes diminui, a ocupação do VCV MPEG-4 diminui. Por outro lado, a ocupação da memória VCV

IST mantém-se aproximadamente constante, já que os macroblocos transparentes têm pouco peso computacional neste modelo.

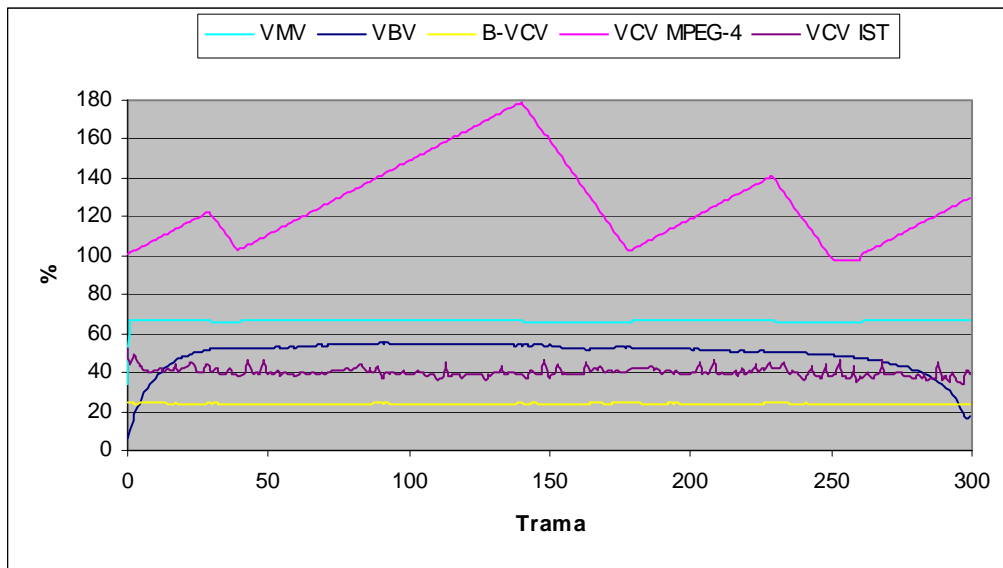


Figura 81 – Evolução da ocupação das várias memórias: sequência News, 4 VOs, CIF a 30 tramas/s para Core@L2 a 2000 kbit/s.

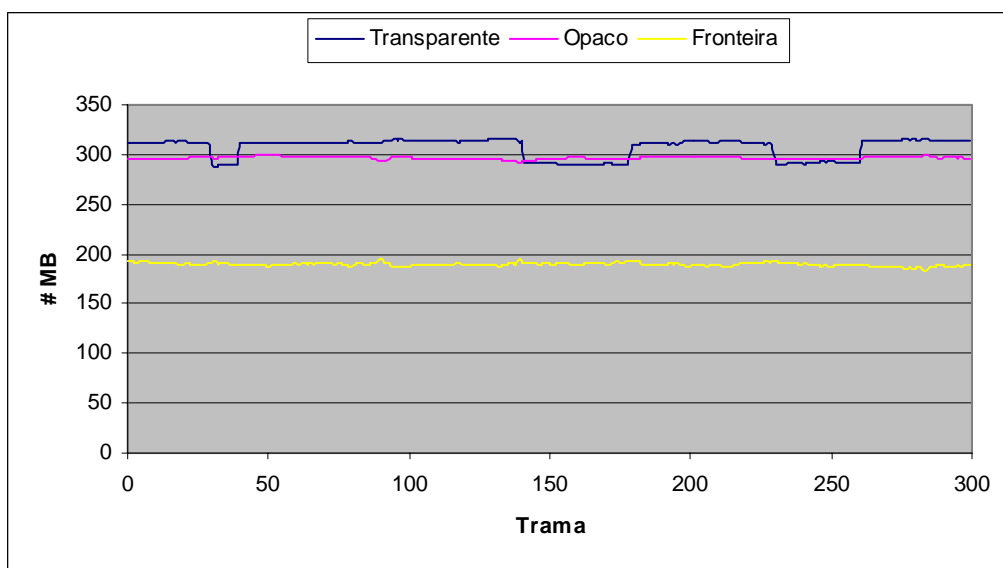


Figura 82 – Evolução do número de macroblocos: sequência News, 4 VOs, CIF a 30 tramas/s para Core@L2 a 2000 kbit/s.

Assumindo como suficientemente válidos os pesos de complexidade determinados no capítulo 5, os exemplos anteriores mostram que a adoção do modelo VCV IST permitiria que fluxos binários considerados não conformes para um dado perfil@nível usando o modelo MPEG-4, passassem a ser conformes, isto mantendo os mesmos recursos de decodificação. Por outras palavras, o modelo VCV IST é mais eficiente no sentido de que permite codificar mais cenas de vídeo em conformidade com a norma para os mesmos recursos de codificação e decodificação. O modelo VCV MPEG-4 sobre-estima a complexidade das cenas, especialmente quando os objectos de vídeo incluem muitos macroblocos transparentes nas suas *bounding boxes* ou quando as *bounding boxes* dos vários VOPs se sobrepõem muito.

6.5 Conclusões

Neste capítulo foi descrito o modelo alternativo proposto para o VCV, baseado na atribuição de pesos diferentes às várias classes de macroblocos usados na codificação de vídeo, relacionados com a sua complexidade efectiva de decodificação. Foram apresentados resultados que mostram claramente as vantagens do modelo desenvolvido relativamente ao modelo especificado na norma MPEG-4 Visual que sobre-dimensiona a complexidade de vários tipos de macroblocos, sobretudo os transparentes, tornando-se ineficiente e desperdiçando recursos. Para além das cenas especialmente criadas para ilustrar situações particularmente críticas em termos do modelo VCV MPEG-4, a aplicação de criação, codificação e decodificação de vídeo MPEG-4, desenvolvida nesta tese e apresentada no capítulo 3, permitiu gerar todos os resultados apresentados neste capítulo.

O modelo VCV IST permite codificar cenas consideradas muito complexas pelo modelo VCV MPEG-4 para um determinado perfil@nível, que podem ser decodificadas sem alterar os recursos de decodificação disponíveis e logo fazendo uma melhor utilização desses recursos. A utilização eficiente dos recursos de decodificação é muito importante, nomeadamente em cenários onde estes recursos são sempre escassos devido a vários motivos, como sejam os terminais móveis e portáteis. Estes tipos de terminais deverão estar na base de muitas das aplicações onde a norma MPEG-4 deverá começar por implantar-se, como o demonstra o facto do fórum 3GPP (*3rd Generation Partnership Project*), responsável pela especificação dos ambientes UMTS, ter já adoptado a norma MPEG-4 Visual para a codificação de vídeo.

Capítulo 7

Conclusões e Trabalho Futuro

O mecanismo de verificação de vídeo especificado na norma MPEG-4 é fundamental para garantir a conformidade dos fluxos binários gerados por um codificador para uma determinada combinação perfil@nível. Nesta tese foi proposto um novo modelo VCV, que faz parte do mecanismo de verificação de vídeo, e que se destina a verificar se a capacidade computacional necessária a um decodificador para decodificar os fluxos binários associados à codificação de uma cena, medida em macroblocos por segundo, não excede os valores especificados por uma dada combinação perfil@nível. O modelo VCV actualmente especificado na norma MPEG-4 apenas faz a distinção entre macroblocos de fronteira e os restantes, o que leva a que não exista qualquer distinção entre a complexidade de decodificação inerente aos vários tipos de macroblocos dentro de cada um destes dois tipos, penalizando especialmente as cenas que contêm muitos macroblocos transparentes ou uma grande sobreposição entre *bounding boxes*.

O modelo VCV proposto nesta tese permite obter uma estimativa mais fiável da complexidade dos objectos de vídeo codificados, baseando-se na atribuição de pesos relativos de complexidade às várias classes de macroblocos identificadas como relevantes. Este modelo apresenta grandes vantagens relativamente ao modelo especificado pela norma MPEG-4, uma vez que permite codificar (e decodificar) cenas que eram consideradas demasiado complexas pelo modelo VCV MPEG-4, sem alterar os recursos de decodificação necessários. Torna-se assim possível fazer um melhor uso dos recursos de decodificação disponíveis, permitindo codificar mais conteúdo de vídeo que pode efectivamente ser decodificado por um decodificador conforme para um determinado perfil@nível.

Na presente tese foi também desenvolvida uma aplicação de edição, codificação e decodificação de vídeo MPEG-4, que oferece grande flexibilidade em termos de criação e codificação de conteúdo de vídeo baseado em objectos. Os fluxos binários podem ser gerados para um determinado perfil e nível, e logo em conformidade com a norma MPEG-4 Visual. Esta aplicação revelou-se muito útil para criar cenas de vídeo MPEG-4 em que uma grande

variedade de objectos de vídeo podem ser combinados, para qualquer escolha de perfil e nível. Na verdade, esta aplicação permitiu criar alguns casos críticos em termos de cenas visuais com o objectivo de mostrar as fragilidades do modelo VCV MPEG-4.

A aplicação de edição, codificação e descodificação de vídeo MPEG-4, desenvolvida no contexto desta tese, mostrou-se adequada aos objectivos para os quais foi concebida, oferecendo uma interface intuitiva e fácil de utilizar, aliada à capacidade de codificação e descodificação de cenas compostas por vários objectos, segundo uma combinação perfil@nível escolhida pelo utilizador. Tanto quanto se sabe, até à data da escrita desta tese, esta é a primeira aplicação a nível mundial a oferecer este tipo de funcionalidade. No entanto, esta aplicação pode com certeza ser melhorada, acrescentando-se novas capacidades e funcionalidades, nomeadamente:

- **Fluxos de descrição da cena em BIFS** – A aplicação poderia ser complementada com um codificador e descodificador de BIFS para os fluxos de descrição da cena. Este formato (*Binary Format For Scenes*), especificado pela norma MPEG-4 Sistemas [8], permite descrever a composição de uma cena através das relações espaciais e temporais dos vários objectos que a compõem e também a sua evolução dinâmica. Para além disso, permite também criar novos objectos gráficos (2D e 3D) ou de texto e inseri-los na cena. A aplicação disponível representa as relações espaciais e temporais dos objectos na cena, mas não utiliza o formato BIFS especificado pela norma MPEG-4, o que é uma limitação em termos de interoperabilidade. Em vez do fluxo BIFS usa-se um ficheiro de texto para armazenar a informação de descrição de cena. A utilização de fluxos em BIFS permitiria também acrescentar aos objectos de vídeo na cena texto e gráficos codificados justamente como informação de BIFS e não como vídeo normal, o que é um dos grandes benefícios da norma MPEG-4. A não utilização do formato BIFS foi, no entanto, uma opção consciente para limitar o volume de trabalho a realizar no âmbito desta tese.
- **Implementação das ferramentas de codificação actualmente não disponíveis** – Em termos de interface, a aplicação está preparada para utilizar algumas ferramentas de codificação incluídas na norma MPEG-4 Visual mas ainda não disponíveis no codec utilizado. Por exemplo, a interface da aplicação permite a utilização de escalabilidade, isto é, a codificação usando várias camadas de melhoramento que são adicionadas a uma camada base para a obtenção de escalabilidade espacial e temporal. No entanto, a versão do codificador de vídeo MPEG-4 utilizado não suporta esta funcionalidade. Se for incluída a ferramenta de escalabilidade no codificador, será possível codificar objectos de vídeo de um modo escalável, recorrendo sempre à mesma interface fácil e intuitiva que é disponibilizada pela aplicação. A escalabilidade permite que o mesmo conteúdo codificado correspondente a um objecto de vídeo possa ser descodificado por um descodificador simples com uma qualidade básica ou por um descodificador mais evoluído com uma qualidade melhor. Além disso, a aplicação está preparada para a utilização de B-VOPs, mas esta ferramenta de codificação também não está disponível no codificador de vídeo usado.

Em termos de ferramentas não-normativas, seria interessante incluir na aplicação ferramentas de resiliência a erros, nomeadamente no descodificador para permitir a eficiente detecção, localização e ocultação de erros.

- **Descodificação e visualização em tempo real** – Caso fosse implementado um descodificador com a capacidade de funcionar em tempo real, seria interessante mostrar o resultado da descodificação em tempo real e não utilizando ficheiros como agora. Seria também possível descodificar fluxos binários recebidos em modo *streaming*, por exemplo através da Internet, e visualizá-los em tempo real.
- **Implementação dos novos *tipos de objecto* e perfis de vídeo** – A aplicação desenvolvida suporta os *tipos de objecto* e perfis de vídeo naturais definidos na versão 1 da norma MPEG-4 Visual. Depois de esta versão da norma ter sido terminada, foram acrescentados novos *tipos de objecto* e perfis à norma, que apesar de estarem presentes na interface da aplicação, não podem ser seleccionados por não estarem implementados no codec de vídeo usado. Assim, a aplicação suportará estes novos perfis e *tipos de objecto* desde que sejam implementados no codec de vídeo.
- **Integração dos *tipos de objecto* e perfis visuais sintéticos** – Uma funcionalidade bastante interessante que poderia ser acrescentada à aplicação seria a integração dos *tipos de objecto* e perfis visuais sintéticos, por exemplo malhas 2D animadas, mapeadas com imagens ou vídeo, ou faces 3D animadas. Com esta funcionalidade seria possível criar cenas onde objectos naturais e sintéticos se encontram harmoniosamente integrados, sendo cada um deles codificado de acordo com as suas características. Esta funcionalidade é, sem dúvida, uma das funcionalidades mais importantes da norma MPEG-4.
- **Inclusão da edição, codificação e descodificação de voz e áudio** – Para ter cenas audiovisuais seria necessário que a aplicação permitisse a edição, codificação e descodificação de voz e áudio, eventualmente associados a objectos de vídeo. Esta funcionalidade requer a integração de codecs de voz e áudio e também a adaptação da aplicação para poder editar, codificar e descodificar voz e áudio, acrescentando à aplicação duas novas dimensões à dimensão vídeo que já possui. Relembre-se que a codificação de áudio e voz se baseia em tecnologias muito diferentes, atendendo às diferentes características dos sinais em questão.

Como a aplicação foi desenvolvida de forma modular, a integração de novas ferramentas, como as acima referidas, pode se feita de um modo relativamente simples. Do mesmo modo, novas versões do codificador ou do descodificador podem ser facilmente integradas na aplicação, bastando para isso realizar uma mera substituição de ficheiros.

Os estudos realizados com vista à avaliação da complexidade de descodificação dos vários tipos de macroblocos de vídeo no contexto da norma MPEG-4, embora feitos com o maior rigor e cuidado possíveis, não esgotam o trabalho que pode ser efectuado nesta área. Assim, algumas possibilidades em termos da continuação do trabalho relacionadas com o modelo de complexidade VCV seriam:

- **Confirmação da validade do modelo VCV proposto usando um descodificador em tempo real** – Apesar de o descodificador utilizado nesta tese estar em conformidade com a norma MPEG-4, não descodifica os objectos de vídeo em tempo real. Para poder confirmar a validade do modelo VCV proposto, seria conveniente utilizar um descodificador em tempo real que permitisse descodificar fluxos binários codificados segundo esse modelo. Deste modo seria possível comprovar experimentalmente que o

modelo VCV proposto permite codificar mais conteúdo relativamente ao modelo VCV especificado na norma MPEG-4 e que ainda assim pode ser decodificado por um decodificador conforme para uma dada combinação perfil@nível. Isto significa que as cenas de vídeo que são consideradas “legais” segundo o novo modelo (mas “ilegais” segundo o modelo VCV MPEG-4) podem ser efectivamente decodificadas, respeitando os tempos relevantes, sem qualquer alteração dos recursos de decodificação disponíveis.

- **Confirmação da avaliação dos pesos relativos de complexidade usando outros decodificadores otimizados** – Os pesos relativos para a complexidade de decodificação dos vários tipos de macroblocos foram obtidos utilizando o decodificador de vídeo MoMuSys/IST. Seria interessante utilizar outros decodificadores otimizados para confirmar os valores destes pesos ou talvez fazer algumas correcções que fossem entretanto julgadas necessárias.
- **Inclusão no modelo dos pesos relativos a B-VOPs** – O modelo VCV proposto nesta tese foi desenvolvido para todos os tipos de macroblocos que podem surgir em VOPs I e P, usando as ferramentas de codificação incluídas na versão 1 da norma MPEG-4, mas é natural que a complexidade de decodificação associada aos novos tipos de macroblocos existentes nos B-VOPs seja maior do que aquela associada aos macroblocos que fazem parte de I-VOPs ou P-VOPs. Assim, caso se dispusesse de um codec que permitisse a codificação/decodificação de B-VOPs, poder-se-ia acrescentar ao modelo os pesos relativos de complexidade associados aos tipos de macroblocos associados aos B-VOPs.
- **Inclusão no modelo dos novos tipos de macroblocos de vídeo** – Alguns dos novos *tipos de objecto* e perfis especificados pela norma MPEG-4 depois da versão 1 usam novos tipos de macroblocos de vídeo por incluírem novas ferramentas de codificação. O modelo VCV proposto nesta tese poderia ser completado incluindo estes novos tipos de macroblocos depois de medidas de complexidade relativas a estes novos tipos de macroblocos terem sido obtidas.
- **Refinamento do modelo utilizando estatísticas da ocorrência dos macroblocos** – Como trabalho futuro, poderia também tentar-se refinar o modelo de complexidade através de estatísticas que reflectam a probabilidade de ocorrência de determinados tipos de macroblocos e dentro de cada tipo a probabilidade de ocorrência de macroblocos mais complexos. Deste modo poder-se-ia obter um modelo que oferecesse uma estimativa da complexidade de uma cena codificada de um modo mais eficaz, em alternativa ao modelo proposto onde a estimativa é baseada nos valores máximos da complexidade dos vários tipos de macroblocos. Para esta análise ser o mais precisa possível, seria necessário utilizar um decodificador optimizado e que efectuasse a decodificação em tempo real.

Os resultados alcançados nesta tese demonstram que o modelo de verificação de vídeo actualmente especificado na norma MPEG-4 faz um uso pouco eficiente dos recursos de decodificação disponíveis, para uma dada combinação perfil@nível, em virtude da sobre-estimação da complexidade efectiva dos objectos de vídeo feita pelo modelo VCV MPEG-4. Este facto pode constituir uma forte limitação, por exemplo para terminais móveis, onde a capacidade computacional se paga cara em termos de volume e de consumo de energia. Pelo contrário, o modelo VCV proposto permitirá a codificação de conteúdo de vídeo para um dado perfil@nível que não pode ser codificado com o actual modelo definido na norma

MPEG-4, isto sem alterar os recursos de descodificação necessários que serão sempre os associados ao perfil@nível em questão.

O autor espera que o trabalho desenvolvido na presente tese contribua para a divulgação das enormes potencialidades da norma MPEG-4 e para a discussão em torno do actual modelo de complexidade especificado na norma, que já mostrou ter fortes limitações. Nesse sentido, a aplicação desenvolvida (versão executável) foi já disponibilizada, para teste e demonstração, a seu pedido, a várias dezenas de empresas, universidades e institutos de investigação, desde a China ao Brasil, incluindo a Europa e os EUA.

Bibliografia

- [1] ISO/IEC 11172: “Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5 Mbit/s”, 1993.
- [2] ISO/IEC 13818: “*Information Technology – Generic Coding of Moving Pictures and Associated Audio Information*”, 1996.
- [3] ISO/IEC 14496-2: 1999, “*Information Technology – Coding of Audio-visual Objects – Part 2: Visual*”, Dezembro de 1999.
- [4] F. Pereira, “*MPEG-4: Why, What, How and When?*”, Image Communication Journal, Número Tutorial sobre a Norma MPEG-4, Vol. 15, nº 4-5, Janeiro de 2000.
- [5] <http://www.bloomberg.com/tv/index.html>
- [6] J. Valentim, P. Nunes. L. Ducla Soares, F. Pereira, “*IST MPEG-4 Video Compliant Framework*”, Documento ISO/IEC M5844, Reunião MPEG de Noordwijkerhout, Março de 2000.
- [7] AOE Group, “*Proposal Package Description (PPD) – Revision 3*”, Documento ISO/IEC N998, Reunião MPEG de Tóquio, Julho de 1995.
- [8] ISO/IEC 14496-1: 1999, “*Information Technology – Coding of Audio-visual Objects – Part 1: Systems*”, Dezembro de 1999.
- [9] ISO/IEC 14496-3: 1999, “*Information Technology – Coding of Audio-visual Objects – Part 3: Audio*”, Dezembro de 1999.
- [10] MPEG, “*Text for FDIS 14496-4: Conformance Testing*”, Documento ISO/MPEG N3067, Reunião MPEG de Maui, Dezembro de 1999.
- [11] MPEG, “*Text for FDIS 14496-5: Reference Software*”, Documento ISO/MPEG N2805, Reunião MPEG de Vancouver, Julho de 1999.

- [12] ISO/IEC 14496-6: 1999, "*Information Technology – Coding of Audio-visual Objects – Part 6: Delivery Multimedia Integration Framework (DMIF)*", Dezembro de 1999.
- [13] ISO/IEC 14772-1, "*The Virtual Reality Modeling Language*", 1997, <http://www.vrml.org/Specifications/VRML97>.
- [14] T. Ebrahimi, C. Horne, "*MPEG-4 Natural Video Coding – An Overview*", Signal Processing: Image Communication, Número Tutorial sobre a Norma MPEG-4, Vol. 15, nº 4-5, Janeiro de 2000, pp. 365-385.
- [15] N. Brady, F. Bossen, N. Murphy, "*Context-based Arithmetic Encoding of 2D Shape Sequences*", ICIP'97, Sessão Especial sobre Codificação de Forma, Santa Barbara, CA, 1997, pp. I-29-32.
- [16] 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, Dezembro de 1999, pp. 1170-1189.
- [17] C. K. Chui, "*An Introduction to Wavelets*", Academic Press, 1992.
- [18] ISO/IEC FCD15444-1:2000, "*JPEG 2000 Image Coding System*", Março de 2000.
- [19] S. Martucci, I. Sodagar, T. Chiang, Y. Zhang, "*A Zerotree Wavelet Video Coder*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 7, No. 1, Fevereiro de 1997, pp. 109-118.
- [20] R. Talluri, "*Error Resilient Video Coding in ISO MPEG-4 Standard*", IEEE Communication Magazine, Vol. 6, No. 6, Junho de 1998, pp. 112-119.
- [21] R. Koenen, "*Profiles and Levels in MPEG-4: Approach and Overview*", Signal Processing: Image Communication, Número Tutorial sobre a Norma MPEG-4, Vol. 15, nº 4-5, Janeiro de 2000, pp. 463-478.
- [22] P. Correia, F. Pereira, "*Proposal for an Integrated Video Analysis Framework*", Proceedings of the International Conference on Image Processing 1998 (ICIP'98), Chicago, Outubro de 1998.
- [23] Independent JPEG Group: código fonte, <ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6a.tar.gz>
- [24] C. Loeffler, A. Ligtenberg and G. Moschytz, "*Practical Fast 1-D DCT Algorithms with 11 Multiplications*", Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1989 (ICASSP '89), pp. 988-991.
- [25] P. Nunes, F. Pereira, "*MPEG-4 Compliant Video Encoding: Analysis and Rate Control Strategies*", Proceedings of the ASILOMAR 2000 Conference, Pacific Grove – CA, EUA, Outubro de 2000.