AUDIOVISUAL COMMUNICATIONS

Laboratory Session on Facsimile

Fernando Pereira

The objective of this lab session about facsimile is to get the students familiar with the main aspects of a facsimile transmission. For that purpose, the application "Comunicação de Imagem"[1] will be used which includes among others a module with a facsimile communication system. This lab guide is organized in three parts corresponding to the three main modules in the facsimile communication system available (see Figure 1):

* Facsimile images encoder
* Transmission channel (with corruption of the coded bitstream)
* Facsimile bitstream decoder

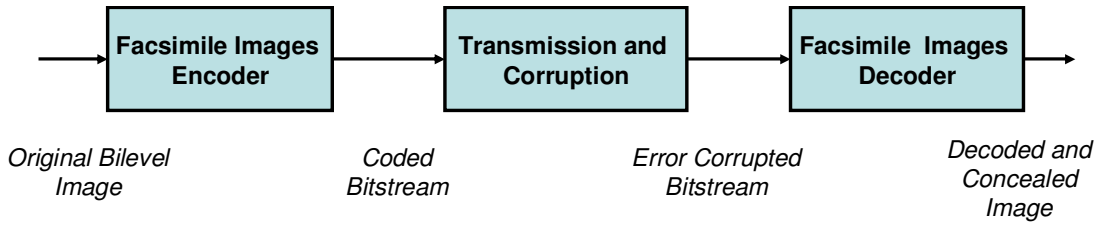

Figure 1 – Simplified architecture of the facsimile communication system.

1. FACSIMILE IMAGES ENCODER

This first module simulates the image encoding process for Group 3 facsimiles, targeting the telephone network, and Group 4 facsimiles, targeting data networks.

1.1 Select Image

This button allows selecting the image to encode from the 8 CCITT (now ITU-T) test images. These test images include letters, electronic schemes, accounting documents, book pages, etc, all with rather different statistical characteristics regarding the black and while data present and their redundancy.

[1] The application "Comunicação de Imagem" has been developed by Pedro Fernandes in the context of his 'Trabalho Final de Curso'. I would like to express here my appreciation for his work considering the possibilities this application has opened for the lab sessions of the Audiovisual Communications course.

After selecting an image in the 'FAX' directory, the image will appear in the screen. Using the mouse, the user may than control the size of the image in the screen, zoom in and zoom out parts of the image as well as define the zone of the image in visualization using the lateral slide bars.

1.2 Properties

This button allows selecting the coding method for the image just selected or already previously selected. The selected image is always mentioned in the encoder dialogue window.

The available coding methods are:

- **GROUP 3 (mandatory method)** – **Modified Huffman Method (MHM)** – This coding method exploits the horizontal redundancy (along the lines) but not the vertical redundancy (between the lines); this means one-dimensional coding is performed. Each line is ended with a *End Of Line* (EOL) codeword; the end of the page is signaled with a *End Of Page* (EOP) codeword which corresponds to 6 EOLs.
- **GROUP 3 extensions (optional)**

  2.1) Modified READ Method (MRM) – The Modified READ (relative addressing) Method uses bidimensional coding since it exploits both the horizontal and vertical redundancies in the facsimile image. To control the propagation of the errors between decoded lines, this method codes some lines one-dimensionally, periodically or not, using the MHM coding method. With this purpose, the application asks the user to introduce the value for the *k* parameter which determines the periodicity for the MHM coded lines among the MRM coded lines which will act as error propagation 'brakes'. Usually, k=2 for low resolution images (3.85 lines/mm) and k=4 for high resolution (7.7 lines/mm). Each line ends with an EOL, followed by a label bit which if a '1' indicates the next line will be MHM codec and if a '0'indicates the next line will be MRM coded.

  2.2.) Modified Modified READ Method (MMRM) – The Modified Modified READ Method is similar to the MRM method using k=∞ which means there are one bidimensionally (MRM) coded lines with the exception of the first line. Since this method is used when it can be assumed that the channel is virtually error free, there is no need to 'take care' of errors and thus neither EOLs, nor label bits are used. EOP corresponds to 6 EOLs followed by the label bit at '1'.
- **GROUP 4 - Modified Modified READ Method (MMRM)** – For Group 4 facsimiles, the coding method is always the MMRM since it is assumed that there is no need to 'take care' of errors since the channel is virtually error free. The errors are dealt with in another way, e.g. in the lower OSI model layers of the Group 4 facsimile terminal. In this case, EOP corresponds to 2 EOLs without label bits.

1.3 Code Image

This button allows coding the selected image with the selected coding method. A file name for the coded bitstream is asked to the user to store the coded bits. The application always suggests a name for this file which corresponds to the image name with an extension corresponding to the coding method ("3h" for Group 3 MHM, "3r" for Group 3 MRM, "3m" for Group 3 MMRM and "4m" for Group 4).

During the coding process, the application will show in the screen two charts with compression factors: a local compression factor corresponding to a sliding window with 8 lines which travels along

the page and a global compression factors corresponding to a cumulative window which include all lines from the start of the page to the line under coding.

1.4 Statistics

After the coding process, the *Statistics* button allows to get much statistical data about the coding process, notably:

* **For each tone, black and white:**

i) Total number of samples for each tone.

ii) Total number of bits for the codewords used to code the samples of a certain tone.

iii) Compression factors for the coding of the samples of a certain tone (ratio between the two previous values).

iv) Average length for the runs for each tone (without considering the division of the runs in make-up and terminating codewords and average length of the partial runs for each tone (considering now the division in make-up and terminating codewords).

v) Number of samples (pixels) coded with make-up and terminating codewords for the MHM. These samples correspond to MHM coding or MRM horizontal mode coding.

vi) Number of make-up and terminating codewords for the samples in v).

vii) Number of samples coded with the MRM vertical and pass modes when MRM is used.

viii) Number of coding symbols corresponding to the sample in vii).

* **For the overall image:**

i) Total number of samples in the image.

ii) Total number of code bits; this total is higher than the sum of the total bits used to code the black and white runs because it also includes the bits corresponding to the EOLs, EOP and label bits.

iii) Global compression factor.

2. TRANSMISSION CHANNEL

This module simulates the transmission of a coded bitstream through a non-ideal channel by inserting bit error corruption with two different error patterns: uniform and bursty.

2.1 Select Stream

This button allows selecting the file with the bitstream to corrupt. As mentioned above, the application suggested file name indicates the name of the image coded while its extension indicates the coding method used.

2.2 Properties

This button allows defining the type of bit errors to be introduced in the selected bitstream. The selected file is always mentioned in the dialogue window of the transmission channel module.

* **Uniform Errors –** For uniform bit errors, the user must insert the desired bit error probability which corresponds to the probability than a bit is corrupted independently of all the others.

* **Burst Errors –** For bursty bit errors, the user must insert the following parameters to characterize the desired errors:

  ◊ **Probability of Occurrence of the Start of an Error Burst** – Probability that an error burst starts at a certain bit.

  ◊ **Minimum Length of an Error Burst –** Minimum length of the bit sequence affected by a burst of errors.

  ◊ **Maximum Length of an Error Burst -** Maximum length of the bit sequence affected by a burst of errors.

  The application computed and shows the bit error probability for the bursts understood as the faction of bits corrupted (see note below).

- **Type of Errors in the Corrupted Bitstream**:

  ◊ **Uniform Errors** – The user must indicate in the corresponding check box if uniform errors must be inserted in the next bitstream 'transmission'.

  ◊ **Error Bursts** - The user must indicate in the corresponding check box if bursty errors must be inserted in the next bitstream 'transmission'

  ◊ ***Skip*** – This parameter allows skipping some initial coded lines in the corruption process; this possibility allows visualizing in the same decoded and concealed image, parts which have been corrupted and not corrupted.

Note: *The application computes the bit error probability for bursty errors (understood as the fraction of bits that have been corrupted) assuming that:*

- *After a burst starts, no other burst may start before the previous burst ends.*
- *For each bit within the burst, the bit error probability is 0.5.*

Than, using representative values, it comes:

- *If the minimum and maximum lengths of a burst is 10 and 30 bits, respectively, than the average length of a burst is 20 bits (average number of bits impacted by the burst) assuming all burst lengths are equally probable.*
- *If the probability of a bit is corrupted within a burst is 0.5, than, on average, there are 10 bits corrupted in a burst.*
- *If the probability of starting a burst in a certain bit is 0.0001 and that burst as 10 bits, on average, than there is a probability of 0.0001 of having 10 bits corrupted, this means there is a probability of* $0.0001 \times 10 = 0.001$ *that any bit is corrupted due to a burst of errors.*

2.3 Transmit Bits

This button implements the error corruption with the selected characteristics on the selected file. Before, the user has to define the name of the file where the corrupted bitstream is to be stored. The application always suggests giving the corrupted bitstream file the same name as the non-corrupted

bitstream file with the addition of a letter in the file extension signaling the type of errors introduced: “b” for burst errors, “u” for uniform errors and “a” for all, meaning both burst and uniform errors.

2.4 Statistics

After the transmission with corruption, the *Statistics* button allows to obtain much statistical data about that process, notably:

i) Number of uniform errors introduced in the bitstream.

ii) Bit error probability corresponding to the uniform errors (burst errors are excluded).

iii) Number of error bursts introduced in the bitstream.

iv) Average length of the error bursts introduced in the bitstream.

v) Number of error bits introduced with the error bursts.

vi) Bit error probability corresponding to the error bursts (in relation to the total number of bits).

3. FACSIMILE IMAGES DECODER

The last module simulates the decoding and error concealment processes for Group 3 and Group 4 facsimiles.

3.1 Select Stream

This button allows selecting the file with corrupted bitstream to be decoded and error concealed. If the application suggestions in terms of file names have been accepted by the user, the file names indicate the name of the coded image, the coding method and the type of errors inserted.

3.2 Properties

Considering that decoding bitstreams which have been corrupted leads inevitably to information losses, and thus to a reduction of the image quality, it is very important to minimize the negative subjective impact associated to the errors in the decoded and error concealed image.

The maximization of the decoded image quality, through the minimization of the negative impact of the error corruption is a decoder task (assuming it is not possible to request the sender to send again the coded data). If no channel coding is also used, the decoder is not able to correct the corrupted bits and should than exploit the so-called error concealment methods which are applied after the decoding of the image. These methods are not standard, do not require the transmission of any additional data, and may be more or less intelligent and complex depending on the manufacturers. This type of error processing where the decoder tries to ‘hide’ the effect of errors based on the received data is more important for the MRM coding method since it suffers from error propagation along the page due to the exploitation of the vertical redundancy (in addition to the horizontal redundancy).

The error concealment options available in the application at hand to minimize the negative subjective impact of the transmission errors are:

* ***Print White*** **(PW)** – Whenever a line has an error, that line is made fully white in the concealed image as well as all the next lines until a line is received in good conditions, this means a one-dimensionally coded line is received without errors.

* ***Print Previous Line*** **(PPL)** - Whenever a line has an error, that line is made the same as the previous line as well as all the next lines until a line is received in good conditions, this means a one-dimensionally coded line is received without errors.

* ***Print Previous Line/White*** **(PLW)** - Whenever a line has an error, that line is made the same as the previous line and the following lines are made white until a line is received in good conditions, this means a one-dimensionally coded line is received without errors. For k=1, PPL and PLW give very similar results.

* ***Normal Decode/Previous Line*** **(NDPL)** – Whenever a line has an error, the decoded part of the line is used and the remaining of the line is made the same as the previous line. The process proceeds with this line becoming the reference for the decoding of the next line until a line is received in good conditions, this means a one-dimensionally coded line is received without errors.

Before decoding a corrupted bitstream, the user has to select one of the error concealment methods above in order the decoder knows what to do to 'hide' the effect of the transmission errors. All the methods above, can only recover the decoding synchronism when a one-dimensionally coded line is received without errors (which may be next one-dimensionally coded line or not). For this to work, the bitstream has to include *End of Line* (EOLs) codewords for the Group 3 facsimiles. For k=∞, this means when no one-dimensionally coded lines are present beside the first one, these methods are rather useless since recovering the decoding synchronism does not happen before the end of the page. In these conditions, the facsimile system must be relying on some alternative solution to deal with the errors, notably by data retransmission using an adequate protocol or channel coding (which may still have residual errors which have to be dealt with by the source decoder). In that case, the image degradation is reduced not by using the 'hiding' methods presented above by rather by correcting the errors themselves and decoding with correct data.

3.3 Decode Stream

This button implements the decoding of the selected corrupted bitstream using for error concealment the method selected by the user. The application always suggests a file name for the decoded and concealed image in this case adding to the file extension some letters identifying the error concealment method used, e.g. PPL for the Print Previous Line method.

During the decoding process, the application shows in the screen the detected decoding problems as well as some warnings, notably:

* **Syntactic Errors –** The received codeword does not exist in the code tables which means that an 'impossible' codeword has been received and thus an error must have happened; in this case, the decoder has a location for the error, although typically the erred bit happens a bit before the decoder 'gets lost'.

* **Semantic Errors –** The decoded line has more or less than the expected number of samples, e.g. 1728 for an A4 page, which means that a semantic error has occurred. In this case, there are no syntactic errors since the initially good codewords got transformed into other 'good' codewords, syntactically valid, but semantically (in terms of meaning) the bitstream is not valid anymore. There is no possibility to know where the error is located using only the information from the line under decoding; maybe it is possible to know a bit more by comparing this line with the previously decoded line, if they are very similar.

* **Warning –** Indicates the correctly decoded lines as well as the lines which label bit has a different value from the one expected based on the parameter k; this situation does not imply

any errors, since the decoder always uses the label bit since for some reason the strict MHM periodicity may not have been followed.

After decoding and error concealment, the image is shown in the screen. Moreover, the uses is given the possibility to compare the final image with the original image which has been initially coded, getting the percentage of samples/pixels different (and thus wrong) in the two images. This objective quality metric may be rather misleading since the subjective quality differences between the several error concealment methods are typically much higher than this metric may indicate.

3.4 Statistics

After decoding and concealment, the *Statistics* button allows obtaining much statistical data about the decoding process, notably:

i) Number of lines correctly decoded.

ii) Number of label bits different from what was expected (considering the parameter *k*).

iii) Number of synchronism losses due to syntactic errors ('impossible codewords').

iv) Number of lines decoded with more or less samples than required, typically 1728 for an A4 page.

v) Number of incorrect sample in the final decoded and concealed image, this means white samples than became black and vice-versa.

vi) Percentage of the decoded and concealed image with incorrect pixels, this means white samples than became black and vice-versa.

Audiovisual Communications Course Laboratories

Laboratory Session on Facsimile

Session _____

Nº _____ **Name** _____

Nº _____ **Name** _____

Nº _____ **Name** _____

Try to address the following questions in a way speedy way but always fully understand what you are doing; if you don't understand, ask for help. The most important is to learn …

Whenever configuration details are not provided, use the default parameters available. For each case, analyze the statistics available by pressing the *Statistics* buttons.

1. **Global Compression Factors:** Code the images in the following table with the Group 3 MHM mandatory coding method, the Group 3 MRM optional coding method and the Group 4 MMRM mandatory coding method and report the global compression factor as well as the partial white and black pixels compression factors. Comment on the variation of the global compression factors depending on the coding method and, for each method, depending on the image content.

IMAGE	GROUP 3 (MHM)			GROUP 3 EXTENSION (MRM, K=4)			GROUP 4 (MMRM)		
	WHITE	BLACK	GLOBAL	WHITE	BLACK	GLOBAL	WHITE	BLACK	GLOBAL
CCITT 1									
CCITT 4									
CCITT 6									
CCITT 8									

2. **Local Compression Factors:** Regarding the CCITT 1 and CCITT 8 images, coded with the Group 3 and Group 4 mandatory coding methods (MHM and MMRM), comment on the evolution (along the page) of the local compression factor as a function of the image content.

3. **MRM Coding with Periodic MHM Coding:** Code the images in the following table with Group 3 optional coding method (MRM) for the *k* values indicated and report the obtained global compression factors. Based on the results, comment on the influence of the *k* parameter on the compression factor and the benefits or not of using higher *k* values, considering the image content.

IMAGE	GROUP 3 EXTENSION								
	K = 2			*K* = 4			*K* = 16		
	WHITE	BLACK	GLOBAL	WHITE	BLACK	GLOBAL	WHITE	BLACK	GLOBAL
CCITT 1									
CCITT 7									

4. **MHM Decoding with Uniform Transmission Errors:** Code the CCITT 5 and CCITT 8 images with the MHM coding method and transmit the resulting bitstream through a channel with **uniform error corruption** with bit error probability $P_b = 0,001$. Decode the corrupted bitstream using all the available error concealment methods this means *Print White* (PW), *Print Previous Line/White* (PLW), *Print Previous Line* (PPL) and *Normal Decode/Previous Line* (NDPL). Observe the error concealed images and comment on their subjective impact (cleanness and readability) also considering the computed percentage of incorrect pixels (white which became blacks and vice-versa), the image content characteristics and the error concealment method used. Which conclusions do you take ?

5. **MRM Decoding with Uniform Transmission Errors:** Repeat the previous question, using now MRM coding with *k=4*. Comment on the benefits and disadvantages of using each of the available error concealment techniques; if needed, also test with other images.

6. **Uniform versus Bursty Error Patterns:** Code the CCITT 3 image with the MRM coding method with k=4 and transmit the bitstream in two different channels: i) with **uniform errors** with $P_b = 0,001$; and ii) with **bursty errors** with $P_B = 0,0001$, $L_{\min} = 10$ e $L_{\max} = 50$. After, decode the two bitstreams with the *Normal Decode/Previous Line* (NDPL) error concealment method. Considering the number of corrupted bits introduced in each case (use the *Statistics* button to know the number of corrupted bits), comment on the subjective impact (cleanness and readability) on the concealed images of the two types of error distributions. If needed, change the corruption parameters in order the total number of corrupted bits for the two corruption cases is the same and only their distribution is different.

7. **Lossless Coding:** Describe a simple experiment in the application at hand which is able to demonstrate that all facsimile coding methods tested use lossless coding.