Distortion Model for H.264/AVC Sequences

Master Thesis

Author: **Sara Martínez García**

Supervisors: *Luca Superiori and Markus Rupp*

Technische Universität Wien (TU)
Institut für Nachrischtentechnik und Hochfrequencztechnik (INTHFT)
Universitat Politècnica de Catalunya (UPC)
Escola Politècnica Superior de Castelldefels (EPSC)

19 April 2010

To Francesc Iu Rillo and Elisa Isabel Martínez García,
for giving me strength to fulfill my targets and make
me believe that everything is possible

To my father, for teaching me to work hard on
everything that I do

Abstract

Nowadays, wireless communications are becoming the center of attention. Among the different applications, much advancement has been made in video streaming over this type of networks.

Wireless networks are error-prone networks. Therefore, as a result of the occurrence of an error, video streams can be incorrectly received and, thus, lost by being discarded by the lower layer entities. Consequently, the decoded video sequence becomes damaged and visual artifacts appear.

The scope of this project consists in the implementation and testing of a method in order to increase the quality of H.264/AVC Baseline profile encoded video sequences with QCIF (144x176 pixels) resolution transmitted over error-prone IP-based channels, concretely UMTS networks.

In order to increase the quality of this type of video sequences, the followed strategy lies in the protection of those packets which are estimated to cause higher distortion if they are lost.

CONTENTS

Chapter: Contents

LIST OF FIGURES

Chapter: List of figures

LIST OF ABBREVIATIONS

[ASO]	Arbitrary Slice Ordering
[AVC]	Advanced Video Coding
[CABAC]	Context-Adaptive Binary Arithmetic Coding
[CAVLC]	Context-Adaptive Variable Length Coding
[DCT]	Discrete Cosine Transform
[FMO]	Flexible Macroblock Ordering
[GHD]	Group of packets causing high distortion
[GLD]	Group of packets causing low distortion
[GOP]	Group of Pictures
[I]	Intra
[IEC]	International Electrotechnical Commission
[IP]	Internet Protocol
[ISO]	International Organization for Standardization
[ITU - T]	International Telecommunication Union. The "-T" refers to the "Standardization" group
[JTC1]	Joint Technical Committee 1
[MB]	Macroblock
[MBAFF]	Macroblock-Adaptive Frame-Field
[MC]	Motion Compensation
[MPEG]	Moving Picture Experts Group
[MV]	Motion vectors
[NAL]	Network Adaptation Layer
[P]	Predictive
[PC]	Principal Component
[PCA]	Principal Component Analysis
[P_Skip]	Skipped Predictive macroblock
[PSNR]	Peak Signal-to-Noise Ratio
[QCIF]	Quarter Common Intermediate Format
[QoS]	Quality of Service

Chapter: List of abbreviations

[QP]	Quantization Parameter
[RTP]	Real Time Protocol
[sMB]	Sub-macroblock
[SNR]	Signal-to-Noise Ratio
[TCP]	Transport Control Protocol
[T1's]	Trailing 1's
[UDP]	User Datagram Protocol
[UMTS]	Universal Mobile Telecommunications System
[VCEG]	Video Coding Experts Group
[VCL]	Video Coding Layer
[xDSL]	Family of DSL (Digital Subscriber Line) technologies
[YPSNR]	Luminance Peak Signal-to-Noise Ratio

1. INTRODUCTION

Nowadays we live in the so-called “Information society” where the exchange of information is a significant economic. With the increasing quantity of wireless devices being used by the society (thousands of millions of people worldwide use their cell phones, PDAs or Internet daily), wireless communications are becoming the center point of attention.

Concretely, this project deals with the transmission of real time video streaming over wireless networks, specifically over “Universal Mobile Telecommunications System” (UMTS) networks. In this case the video files have QCIF (Quarter Common Intermediate Format) resolution (144x176 pixels) and are encoded by means of the H.264/AVC (Advance Video Coding) standard.

As a consequence of the wireless channel, which is an error-prone channel, video streams can be incorrectly received or lost as a result of the occurrence of an error which, eventually, causes a packet loss taking into account that the lower layer entities discard any damaged packet. Therefore, the video stream is incorrectly decoded and visual artifacts appear as a result.

The scope of this project consists in the implementation and testing of a method in order to increase the quality of this type of video sequences transmitted over error-prone IP-based (Internet Protocol based) channels. The implementation has been done by means of the H.264/AVC JM 15.0 (FRExt) decoder (1) and MATLAB (R2008b).

The strategy followed in order to increase the quality of the transmitted video sequences consists in the protection of those packets which are estimated to cause higher distortion (decrement of YPSNR) if they are lost. Therefore, the implemented method estimates the total distortion that the different packets can cause if they are lost. In order to estimate this distortion, the method makes use of the features of the transmitted packets.

The following chapters explain a brief introduction to the field of study as well as the different steps followed in order to implement the method and the results obtained.

2. MOTIVATION AND SCOPE

The motivation of this project lies in the standard transmission of H.264/AVC video sequences over wireless channels. In order to give a better understanding of how this transmission affects this type of encoded video sequences, the following subchapters explain the working of the H.264/AVC standard and afterwards the effects of the errors on the decoded video sequences. Finally, taking the later information into account, in subchapter 2.3 it is explained the scope of the project.

2.1. H.264/AVC OVERVIEW

H.264/AVC, also known as MPEG-4, is the latest international video coding standard. It was jointly developed by the ITU-T (International Telecommunication Union) "Video Coding Experts Group" (VCEG) and the ISO/IEC (International Organization for Standardization/ International Electrotechnical Commission) "Moving Picture Experts Group" (MPEG) (2) (3).

The importance of new network access technologies like cable modem, xDSL (Digital Subscriber Line) and UMTS created demand for the H.264/AVC standard. For this reason, the goals of this standardization effort were enhanced compression efficiency and network friendly video representation for interactive (video telephony) and non-interactive applications (broadcast, streaming, storage and video on demand).

Therefore, H.264/AVC was designed to support a wide variety of applications and to operate over several types of networks and systems. As a result, this standard provides gains in compression efficiency of up to 50 % over a wide range of bit rates and video resolutions compared to previous standards (2).

2.1.1. PROFILES AND LEVELS

The H.264/AVC standard defines profiles and levels in order to maximize the interoperability while limiting the complexity. A profile is defined as a subset of coding tools that can be used to generate a bitstream, whereas a level places constraints on certain key parameters of the bitstream.

Therefore, by means of profiles and levels, minimum bounds on the decoding capabilities can be set in order to target different application domains. Hence, H.264/AVC defines different profiles. Next, taking into account the scenario of this project, three of them are explained and their tools can be seen in Figure 2.1 (2):

- **Baseline:** Designed to minimize the complexity and provide high robustness and flexibility, this profile is widely used in video conferencing and mobile applications. Concretely, it corresponds to the profile used in this project.
- **Main:** This profile typically allows the best quality at the cost of higher complexity and delay.
- **Extended:** This profile was designed to combine the robustness of the Baseline profile with a higher degree of coding efficiency and greater network robustness for such applications as flexible video streaming (3).

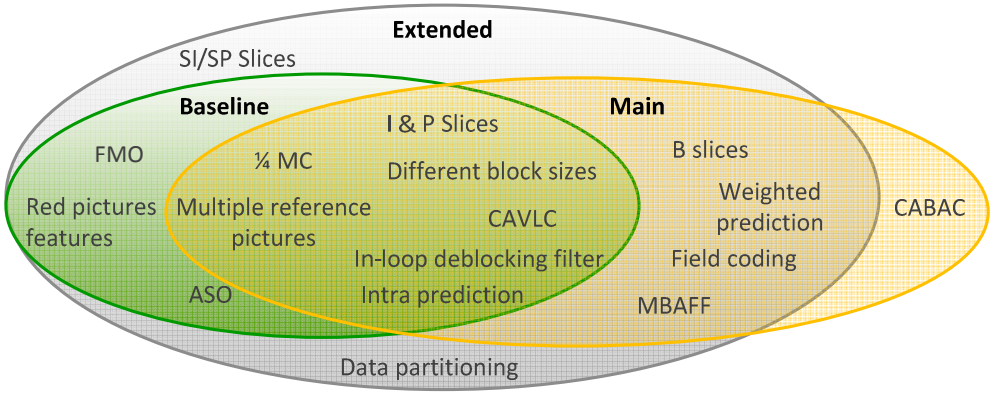


Figure 2.1 H.264/AVC profiles and corresponding tools (2)

2.1.2. STRUCTURE

The H.264/AVC standard defines two conceptual layers: the "Video Coding Layer" (VCL) and the "Network Adaptation Layer" (NAL). The first one defines the efficient encoding representation of the video whereas the second one is designed to provide "network friendliness" by facilitating the ability to map H.264/AVC VCL data to different transport layers.

As in all prior ITU-T and ISO/IEC JTC1 (Joint Technical Committee 1) video standards since H.261, the VCL design follows the so-called "block-based hybrid video coding approach" in which each coded picture is divided into block-shaped units called "macroblocks". Each one of them consists of three components: Y, C_r and C_b, where Y is the luminance component which represents brightness information and C_r and C_b are the chrominances and represent the color information. Due to the fact that the human visual system is more sensitive to luminance than to chrominance, the chrominance signals are both subsampled by a factor of 2 in horizontal and vertical direction. Therefore, a macroblock consists of one block of 16x16 samples for the luminance component and two blocks of 8x8 samples for the chrominance. This is called 4:2:0 sampling with 8 bits of precision per sample.

These macroblocks are coded in Intra or Inter mode. Then, the prediction error, which is the difference between the original and the predicted block, is transformed using a 4x4 integer transformation with similar properties as the 4x4 DCT (Discrete Cosine Transform). The task of the transformation is to reduce the spatial redundancy of the prediction error signal.

As a result of the transformation, a block of transformed coefficients is obtained which is next quantized using a "quantization parameter" (QP) that can take up to 52 different values when video format supports 8 bits per decoded sample. Typically, the result is a block in which most or all the coefficients are zero, with few non-zero coefficients. The aim of the quantization process is to reduce the precision of the coefficients according to the QP. If QP is set to a high value (the step size doubles with each 6 increments of QP), then more coefficients are set to 0 whereas if QP is set to a low value, then more non-zero coefficients will remain after the process. Thus, this process keeps the information from the low frequencies whereas the high frequencies are set to 0.

Next, the quantized transform coefficients of a block are generally scanned in a zig-zag fashion and then transmitted using entropy coding methods. In H.264/AVC, two methods of

entropy coding are supported: a low complexity technique based on the usage of context-adaptively switched sets of variable length codes (Context-Adaptive Variable Length Coding (CAVLC)) and the algorithm of context-based adaptive binary arithmetic coding (Context-Adaptive Binary Arithmetic Coding (CABAC)). The first one corresponds to the method used in the Baseline profile, and is therefore the one used in this project.

In order to reconstruct the same image on the decoder side, the dequantized coefficients are inversed transformed and added to the prediction signal. The result is the reconstructed macroblock.

A sequence of macroblocks is grouped together forming a "slice". Therefore, a picture is a collection of one or more slices in H.264/AVC (Figure 2.2). Moreover, slices are self-contained in the sense that the data within them can be correctly decoded without use of data from other slices belonging to the same frame. As a consequence, in an IP network environment and in the case of this project implementation, each slice corresponds to a packet sent through the network (2) (3) (4).

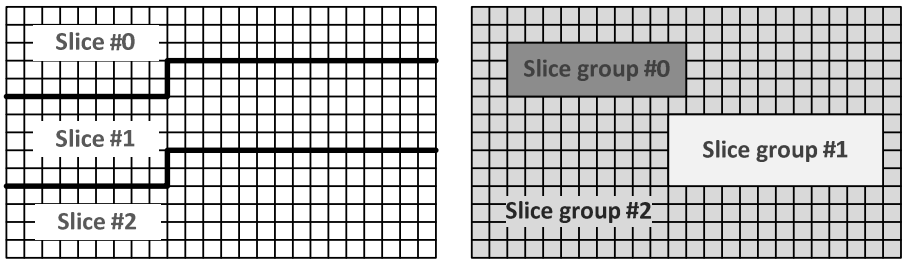


Figure 2.2 Subdivision of a picture into slices when not using FMO[1] (left) and when using it (right)

H.264/AVC supports five different slice-coding types (4):

- **I (intra) slice:** A slice in which all the macroblocks are coded using Intra mode. In this case, Intra mode means that the samples of a macroblock are spatially predicted by using only information of already encoded macroblocks of the same slice.
- **P (predictive) slice:** In this case the macroblocks are coded using Inter mode, i.e. they are predicted by means of "motion compensation". Therefore, the macroblocks are predicted by estimating motion of macroblocks belonging to already encoded reference pictures. For this purpose, each macroblock can be divided into smaller partitions. Figure 2.3 shows the partitions supported by the standard**.**

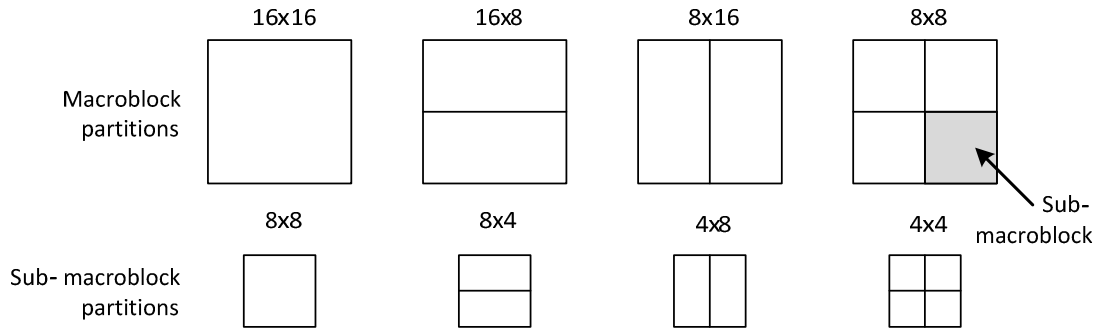


Figure 2.3 Partitioning of a macroblock and a sub-macroblock for motion compensation prediction (2)

[1] FMO stands for "Flexible Macroblock Ordering" and allows modifying the way pictures are partitioned into slices and macroblocks by using the concept of slice groups.

Then, a motion vector is estimated and transmitted for each block that refers to its position in an already transmitted reference image stored in memory. Apart from this motion vector, the picture reference index is also sent.

In addition, a P macroblock can also be coded in the so-called "P_Skip type". For this coding type, neither a quantized prediction error signal, nor a motion vector or reference index parameter is transmitted. The useful definition of the P_Skip coding type is that large areas with no change or constant motion can be represented with very few bits.

H.264 allows multiple reference picture motion compensation, so the motion can be estimated from multiple pictures that lie either in the past or in the future (in the case of B frames) in display order (3) (Figure 2.4). Nevertheless, in this project a buffer size of 1 is used, which means that the previous frame is used as reference in order to do the Inter or temporal prediction (5).

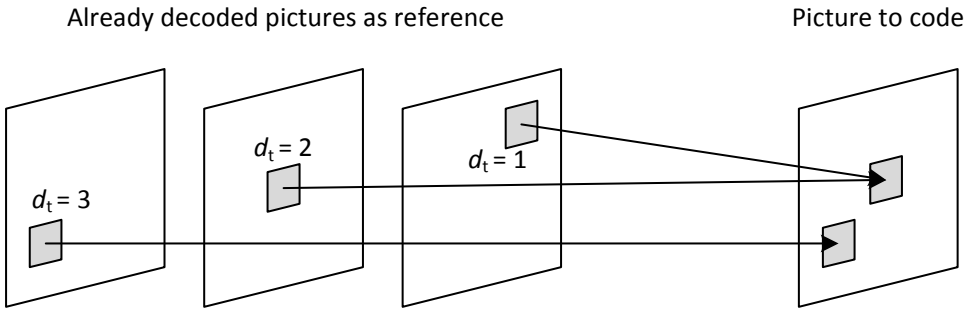


Figure 2.4 Motion-compensated prediction with multiple reference images. In addition to the motion vector, also a picture reference parameter d_t is transmitted (2)

- **B (bi-predictive) slice:** In B slices two motion vectors, representing two estimates of the motion, per sub-macroblock partitions are allowed for temporal prediction. They can be from any reference picture contained in the buffer in future or past in display order (3).
- **SP (switching P) and SI (switching I) slice:** These slices are specified for efficient switching between bitstreams coded at various bit-rates**.**

As in this project the profile used corresponds to the Baseline, only I and P slices are available. However, although the standard defines I and P slices, it is forbidden to have different types of slices in the same frame, therefore the terms of I and P frames are more common used. Since temporal prediction is more effective (in terms of compression) than spatial prediction, once encoded, the size of I frames is in average 3-4 times bigger than the size of P frames. In the case of the project scenario, usually a P frame will fit in a single slice, while I frames will need therefore 3-4 slices (5).

The set of frames from an I frame up to the P frame preceding the next I frame is defined as "Group of Pictures" (GOP) as shown in Figure 2.5. As it was commented before, a buffer size of 1 has been selected in this project. Therefore, since the decoded frames are put in the "decoder picture buffer", if its size is set to 1, that means that the temporal referencing in one frame (inter-prediction) can only be made from the previous frame.

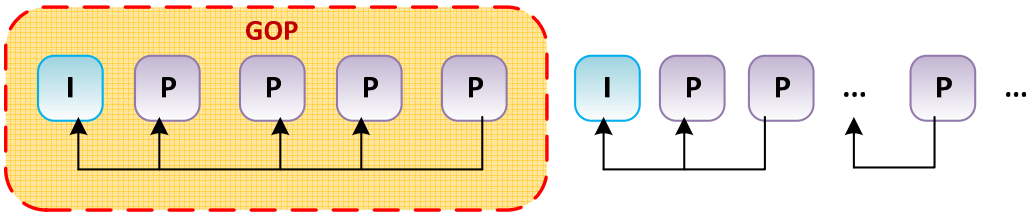


Figure 2.5 Structure of a GOP

Taking into account the temporal dependency between the frames within a GOP, if some error is produced during the transmission of data through the network, then as a consequence, a visual error in the video sequence will propagate within the frames of the GOP. Therefore, in subchapter 2.2, the errors in which this project will focus will be explained as well as the visual result of these errors in the video sequence.

2.2. Effects of errors

Video transmission over UMTS and wireless networks in general is a challenging task taking into account the requirements in delay and data rates of services over such networks and also, acknowledging the fact that the mobile environment is characterized by harsh transmission conditions in terms of attenuation, shadowing, fading and multi-user interference.

2.2.1. Transport of H.264/AVC video

Before explaining the effect of the errors in the video stream, it is first needed to know how the data is structured in order to be transported. In this case, the video stream is encoded and the resulting coded video data (the bitstream obtained after the entropy coding process and the other additional information) is organized into NAL units each of which is effectively a packet that contains an integer number of bytes. The first byte of each NAL unit is a header byte that contains an indication of the type of data in the NAL unit, and the remaining bytes contain payload data of the type indicated by the header.

NAL units are classified into VCL and non-VCL units. The first ones contain the data associated to the video slice whereas the second ones contain any associated additional information such as parameter sets and supplemental enhancement information. The parameter sets correspond to important header information that can apply to a large number of VCL NAL units, therefore each VCL NAL unit refers to a non-VCL NAL unit (4). In the case of this project, the research focuses on the loss of VCL NAL units.

Afterwards, independently of their type, NAL units are encapsulated into the “Real Time Protocol” (RTP) as shown in Figure 2.6. The resulting packets are then transported end-to-end by means of “User Datagram Protocol” (UDP). Although UDP offers a simple and unreliable datagram transport service, it is used instead of “Transport Control Protocol” (TCP) due to TCP’s unpredictable delay characteristics which are not suitable for real-time applications such as video streaming. As a consequence, transmission errors cannot be avoided (5) (6). Nevertheless, H.264 has been designed in order to provide several error resilience features for wired and wireless transmission (7).

Figure 2.6 Packetization of NAL units in RTP/UDP/IP

2.2.2. VISUAL ARTIFACTS PROVOKED BY ERRORS

Usually two kinds of errors are present in today's IP transmission systems: bit inversion errors or packet losses. In the standardization of the H.264/AVC standard, it was assumed that bit errors are detected by the lower layer entities (in this case the IP or UDP checksum) and any remaining transmission error result in a packet loss, therefore a slice loss, as was commented before in subchapter 2.1 (7). Consequently, this project will focus on the later type of errors.

Packet or slice losses produce visual artifacts in the image. Packet losses in I frames tend to produce much more visible artifacts than for P frames. This is because of the different type of information that the NAL unit transports for each type of prediction.

Furthermore, since H.264/AVC uses temporal prediction, the error will propagate in the P frames following the error since they directly use a corrupted frame as reference (since this project uses a buffer of size 1). The error then will propagate until the end of the GOP at which point the picture buffer will be cleared. In the case of packet losses in I frames, the error will propagate spatially in the same I slice and also temporally in the following set of P frames (5).

Since the probability that a single I frame is corrupted becomes higher because encoded I frames are bigger, a packet loss would be more probable in these frames. However, as in this project it is assumed that a bigger quantity of P frames are sent instead of I frames, then it is much more probable that a P frame becomes corrupted. Therefore, this project will focus on the estimation of the distortion produced by packet losses (VCL NAL units) in P frames. Figure 2.7 shows some of the visual artifacts caused by packet losses in P frames.

Figure 2.7 Examples of visual artifacts

2.3. SCOPE OF THE PROJECT

As it was said in subchapter 2.2, the mobile environment is characterized by harsh transmission conditions, therefore provoking that wireless channels (as the one used in UMTS) are prone to errors. Thus, the transmission working of H.264/AVC is as shown in Figure 2.8.

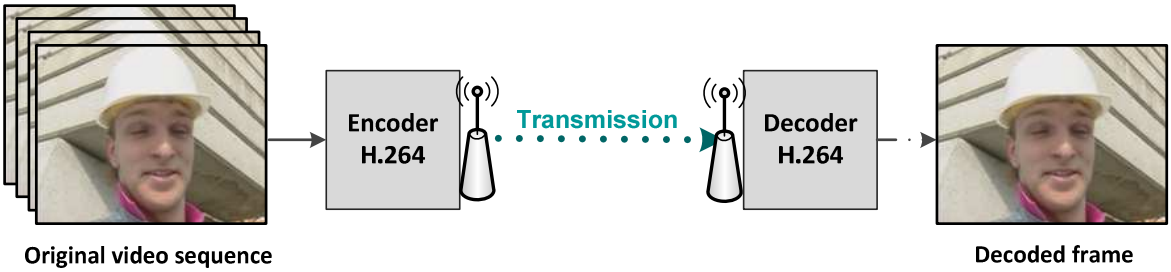


Figure 2.8 Transmission of video sequences in H.264/AVC (wireless network)

The sequence of frames (or video sequences) that is wanted to be transmitted (in the case of this project, the format of the video sequences used is QCIF which means a resolution of 144x176 pixels) is encoded by means of the H.264/AVC encoder and then transmitted through the wireless channel. Then, the decoder receives it and decodes it, giving as a result the reconstructed video sequence.

If the transmission is successful and no error is produced, then the reconstructed video sequence at the decoder must be an approximation of the original one taking into account the losses related to the reconstruction process (e.g. dequantization and inverse transformation). However, if some error occurs and, as a consequence, a packet loss is produced; then the decoded video sequence will be corrupted.

Since in this scenario the packets are being sent by means of UDP, thus the quality control in the transmission process is deficient. Moreover, the decoder side has mechanisms in order to conceal visual artifacts produced by errors, but they cannot reconstruct the lost information. Hence, if the final quality of the decoded video sequence is wanted to increase, then the solution has to be built in the encoder side.

This project proposes to take advantage of the possibility of having different logical channels with different “Quality of Service” (QoS) parameters in order to send the most important packets through the highest QoS channel. Thus, in order to distinguish which packets are more important than the others, the project proposes to define the importance of a packet depending on the total distortion than this packet can cause in the decoded sequence if it is lost.

Hence, those packets which are estimated to cause higher distortion in the decoded video sequence if they are lost would be considered more important and, therefore, sent through higher QoS channels and, those packets which are estimated to cause lower distortion; they would be considered less important than the others and consequently, sent through lower QoS channels.

Therefore, the aim of this project is to build a ranking model (Figure 2.9) which will work by GOP basis. In this case the I frames will be directly sent through the highest QoS channel and then, the ranking model will take all the packets of the P frames of the GOP and, per

each one of them, it will estimate the total distortion than this packet can cause within the GOP if it is lost.

Afterwards, depending on this value, the ranking model will decide through which channel the packet should be sent.

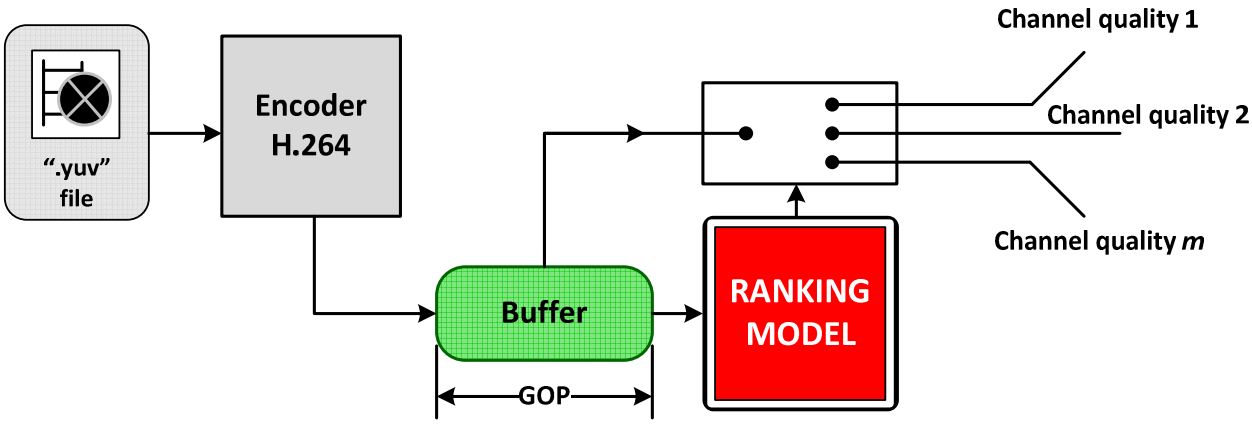


Figure 2.9 Working scheme of the ranking model

In order to estimate the distortion that each packet of the P frames can cause, the ranking model will analyze the features of the packets. In this case, the estimation of the total distortion caused by the loss of a packet is split in two calculations. First, it is needed to estimate the distortion caused in the frame where the packet is lost and then, to estimate the evolution of the distortion in the following P frames until the end of the GOP caused because of the temporal error propagation (section 2.2.2).

Next, the following chapters explain the procedures carried out in order to estimate each of the mentioned distortions and also, the steps followed in order to create the ranking model as well as the results obtained.

3. DISTORTION CHARACTERIZATION (I)

As it has been mentioned in the previous chapter (subchapter 2.3), two different estimations of the distortion have to be calculated in order to estimate the total distortion that a packet can cause within the GOP. Since the distortion has to be estimated by means of the features of the packets; this chapter, therefore, is centered on studying how the features of the lost packets affect the distortion measured in the decoded sequence.

The following chapters explain the steps followed in order to carry out this study as well as the results obtained for different types of scenarios.

3.1. INITIAL WORKING ENVIRONMENT

Before starting the explanation of all the stages carried out in order to fulfill the scope of the project, it is needed to know the initial environment of the project (Figure 3.1).

First, the H.264/AVC JM 15.0 (FREXt) reference software was downloaded (1). A workspace containing the encoder and decoder projects was obtained. In this case, in order to work with this workspace, Microsoft Visual Studio 2008 has been used. When building each of these projects, two applications are obtained: "lencod.exe" and "ldecod.exe". These correspond to the JM encoder and the JM decoder respectively.

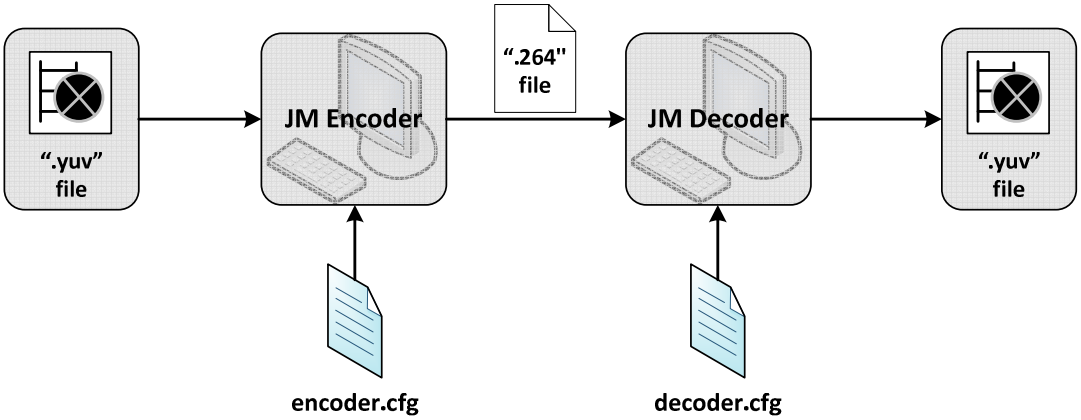


Figure 3.1 Initial working environment

Then, in order to encode a video sequence, the following is typed in the "Command Prompt" where all the compiled files are situated:

```
lencod – d encoder.cfg
```

Therefore, the "-d" implies that the JM encoder uses the configuration described in the file "encoder.cfg". In this last file, the different parameters for the encoding process are established: video sequence to encode (in this project the video sequences correspond to ".yuv" files), frame rate, QP, GOP size, among others.

Then, as a result, the encoder outputs a ".264 file" which contains the different encoded packets. Next, in order to start the decoding, in the "Command Prompt" it is typed the following:

```
ldecod decoder.cfg
```

Therefore, the decoder takes all the decoding parameters from the "decoder.cfg" file such as ".264" file to decode, name of the outputted file and reference file among others. Finally, the reconstructed video sequence, a ".yuv" file, is obtained.

Next, the procedures carried out in order to study the relationship between the distortion and the features of the lost packets are explained.

3.2. DATA ANALYSIS

In order to detect which characteristics of the video sequence have an effect on the final quality of the decoded video sequence, this first stage is centered in extracting the features of the different packets of the encoded video sequence. Concretely, the search has been focused in the following set of parameters:

- Quantized coefficients
- Motion vectors
- P_Skip mode
- Macroblock type

These features have been selected since they are susceptible to affect the final quality of the decoded video sequence. Subsequently, the steps followed in order to extract these features are explained.

It has to be pointed out that for simplicity purposes in order to explain the work carried out, from now on "macroblock" and "sub-macroblock" will be referred as "MB" and "sMB", respectively.

3.2.1. MODIFICATIONS IN THE JM H.264/AVC DECODER

Although it was intended to create an algorithm that works at the end of the encoder, it was found that, in order to extract the data to be analyzed, it was easier to modify the decoder. Therefore, the JM reference H.264/AVC decoder was modified by means of Microsoft Visual Studio in order to output the previous features meanwhile the encoded video sequence is being decoded.

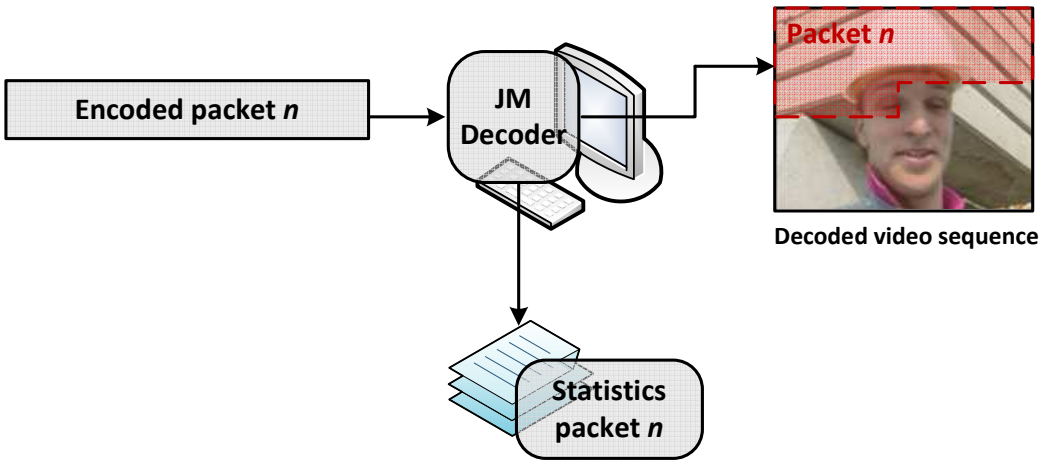


Figure 3.2 Working of the modified JM decoder

As Figure 3.2 shows, concretely, the modified decoder outputs the previous features of each encoded packet of the video sequence at the same time that is decoding each of them. In this case, the outputted features are stored in text files in order to be, afterwards, processed by means of MATLAB.

3.2.2. RESULTS

Once the variables that store the parameters of interest were detected, the code of the JM H.264/AVC decoder was modified (see Appendix A) in order to output these variables and, as a result, a set of 7 different text files are obtained. These text files store the information of all the packets of the encoded video sequence and concretely each of them store the following data:

- Nonzero coefficients per each sMB
- Coefficients equal to zero per each sMB
- Absolute vertical motion vectors
- Absolute horizontal motion vectors
- Relative vertical motion vectors
- Relative horizontal motion vectors
- Type of MB and P_Skip mode

Since in this project the Baseline profile is being used, the entropy coding method corresponds to CAVLC. Therefore, in the CAVLC entropy coding method, the number of nonzero quantized coefficients and the actual size and position of the coefficients are coded separately and that is the reason behind the need to store them in two different text files. For extra information, see Appendix B.

Also, apart from the absolute motion vectors, which from now on will be named “MV”, it was decided to store also the relative ones. When encoding, the encoder calculates the predicted MV per each MB but what is sent is the error which corresponds to the difference between the predicted and the absolute MV since the decoder predicts the MV and adjusts the value with the error in order to obtain the absolute MV. Furthermore, the vertical and horizontal components are stored apart (in the text files) in order to analyze whether one of them has a higher effect on the final quality or not.

3.3. DATA PROCESSING

Once the parameters of interest of the encoded video sequence were outputted in text files, they were processed by means of MATLAB. Actually, the MATLAB version used in this project is the R2008b.

Concretely, this stage is divided in two steps. The first one corresponds to create a structure that stores all the information of the text files so it is possible to have all the information well organized in a data element. Then, the second step is focused on obtaining quality parameters of the decoded video sequence in the case when some packets are removed.

3.3.1. CREATE A STRUCTURE

As it was said before, the text files obtained in the previous stage are processed by means of MATLAB in order to create a structure (“struct” in MATLAB terms) that stores all the information. In order to carry out this task, different scripts were created which extract the

information within each of the text files and do some operations in order to fill the different fields of the structure. For a more detailed account on the working of the scripts in order to create the structure, go to Appendix A subchapter C.1.

As a result, the following structure, from now on called "PIC", is obtained (Figure 3.3):

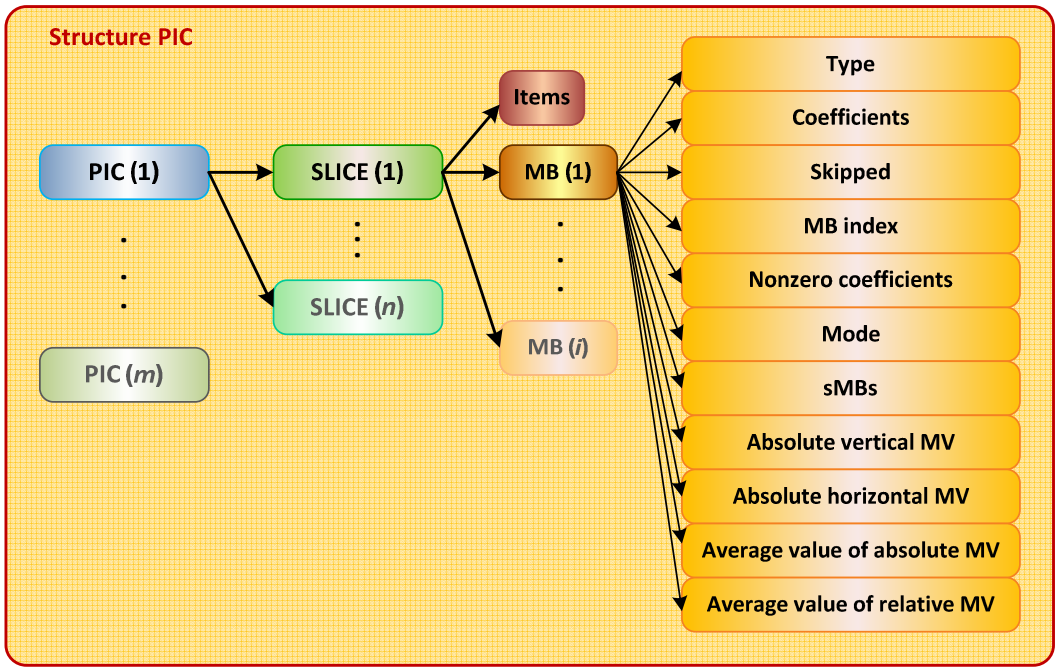


Figure 3.3 PIC structure

It can be seen that the structure consists of different elements. The first elements of the structure correspond to the frames. Therefore, "PIC (1)" corresponds to the first frame of the video sequence and "PIC (*m*)" corresponds to the last one. Then, each of them is composed of a number of slices and, again, each of those slices is composed by two fields. The first field called "Items" stores the number of MBs in that slice. The second field corresponds to a vector of MBs and, for each MB; a number of features is stored:

- **Type:** Indicates the type of MB which depends among other factors on the type of slice to which the MB belongs and how the MB has been portioned (8).
- **Coefficients:** As a result of the transformation of the predicted error of a block of 4x4 samples, a block of 16 coefficients is obtained. These blocks (the blocks of 4x4 samples) will be treated as sMB. Therefore, as each MB is composed of 16x16 samples, that means a total of 16 sMB per MB. Thus, the field "Coefficients" is an array that stores per each position (sMB) a vector of 16 coefficients.
- **Skipped:** Refers to the parameter P_Skip (section 2.1.2). Its value can be "0" if the MB has been skipped and "1" if it has not been skipped.
- **MB index:** Stores the index of the MB within the frame.
- **Nonzero coefficients:** Stores the total number of nonzero coefficients of the whole MB.
- **Mode:** This field can get three different values depending on if the MB has been split (Figure 2.3). If it takes the value "1", that means that the MB has not been subdivided (16x16). Then, if it takes the value "2", then that means that the MB has been

subdivided in 2 (16x8 or 8x16) and finally, if it takes the value "4" that means that the MB has been subdivided in 4 (8x8).

- **sMBs:** As what is transformed is the error between prediction and original, if the prediction of a sMB is equal to the original, then for that sMB there will be no coefficients. Therefore, this field indicates the number of sMBs within the MB that have a vector of coefficients.
- **Absolute vertical MV:** For each sMB there is a MV with vertical and horizontal component. This field corresponds to a vector of 16 positions where, in each position, it is stored the vertical component of the MV of each sMB within the MB.
- **Absolute horizontal MV:** In this case it stores the horizontal component of the MV of each sMB.
- **Average value of absolute MV:** Taking the last two fields, it stores the average absolute movement of the MB in vertical and horizontal directions.
- **Average value of relative MV:** In this case it is stored the average relative movement (difference between the predicted MV and the absolute ones) of the MB in vertical and horizontal directions.

3.3.2. CALCULATE QUALITY PARAMETERS

In the previous step, the information of the statistics of the encoded video sequence has been obtained or, in other words, the information that would be obtained at the output of the encoder. Then, this step is based on acquiring information about the distortion that a loss of packets can produce in the final quality of the decoded video sequence. Therefore, this step is based on the following scheme (Figure 3.4):

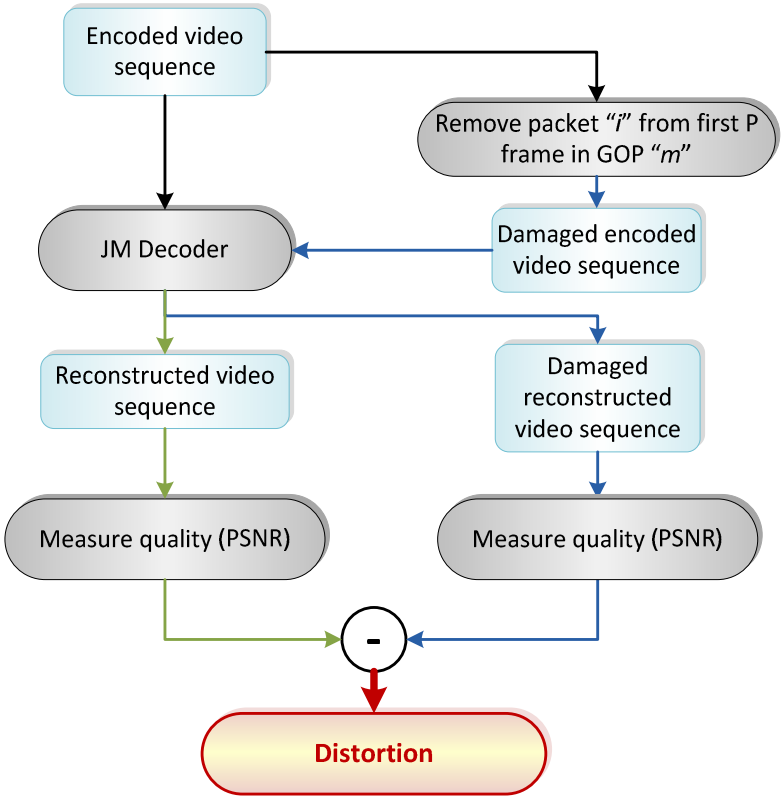


Figure 3.4 Distortion measurement scheme

3.3.2.1. REMOVAL OF PACKETS

When the JM encoder encodes a video sequence, it outputs a file with ".264" extension which stores the information of the encoded packets. This file is taken by the JM decoder which decodes it and, as a result, the reconstructed video sequence is obtained. In order to encode and decode the different video sequences, two files with ".cfg" extension are used, one for the encoder and the other for the decoder. These files indicate the configuration of the encoder and the decoder in order to encode and decode (Figure 3.1).

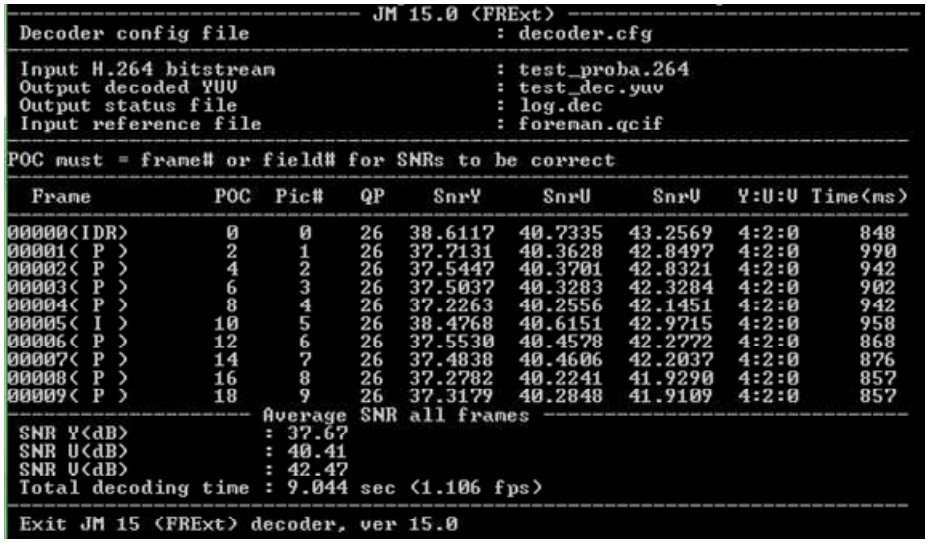
In this case, what is wanted to know is which effect a removal of a specific packet of the first P frame of each GOP can produce in the final quality.

Therefore, a MATLAB script called "skip_n_packet_P.m" was created. This script takes as input the ".264" obtained after the encoding process and reads it. Next, it removes the indicated packet (first, second…) from the first P frame of each GOP.

So, at the end, a list of "damaged .264 files" is obtained in which each ".264" file corresponds to the encoded sequence but removing a determined packet from the first P frame of each GOP. The amount of "damaged .264 files" created must be equal to the amount of slices (remembering that a slice corresponds to a packet) per frame.

3.3.2.2. DECODING THE CORRECT AND DAMAGED ".264" FILES

Once the correct and damaged ".264" files have been created, they are decoded. When decoding, the JM decoder prints valuable information in the screen of the "Command Prompt" as Figure 3.5 shows. Some of this information corresponds to the quality (SNR (Signal-to-Noise Ratio)) of each decoded picture taking into account a reference video sequence which corresponds to the original video sequence encoded in the beginning.



```
------------------------- JM 15.0 (FREXt) -------------------------
Decoder config file                    : decoder.cfg
-------------------------------------------------------------------
Input H.264 bitstream                  : test_proba.264
Output decoded YUV                     : test_dec.yuv
Output status file                     : log.dec
Input reference file                   : foreman.qcif
-------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
-------------------------------------------------------------------
  Frame          POC  Pic#   QP     SnrY     SnrU     SnrV   Y:U:V Time(ms)
-------------------------------------------------------------------
00000(IDR)         0     0   26  38.6117  40.7335  43.2569  4:2:0      848
00001( P )         2     1   26  37.7131  40.3628  42.8497  4:2:0      990
00002( P )         4     2   26  37.5447  40.3701  42.8321  4:2:0      942
00003( P )         6     3   26  37.5037  40.3283  42.3284  4:2:0      902
00004( P )         8     4   26  37.2263  40.2556  42.1451  4:2:0      942
00005( I )        10     5   26  38.4768  40.6151  42.9715  4:2:0      958
00006( P )        12     6   26  37.5530  40.4578  42.2772  4:2:0      868
00007( P )        14     7   26  37.4838  40.4606  42.2037  4:2:0      876
00008( P )        16     8   26  37.2782  40.2241  41.9290  4:2:0      857
00009( P )        18     9   26  37.3179  40.2848  41.9109  4:2:0      857
------------------- Average SNR all frames ------------------------
SNR Y(dB)              : 37.67
SNR U(dB)              : 40.41
SNR V(dB)              : 42.47
Total decoding time    : 9.044 sec (1.106 fps)
-------------------------------------------------------------------
Exit JM 15 (FREXt) decoder, ver 15.0
```

Figure 3.5 Information printed in the screen while decoding

Therefore, the ".264" file corresponding to the correct encoded video sequence is decoded and the outputted information in the screen is stored in a text file. Then, a set of MATLAB scripts were created in order to automatically decode the "damaged .264 files" and store the result of the decoding of each of them in a different text file. For an extended account of the created scripts and their working, go to Appendix A subchapter C.2.

3.3.2.3. CALCULATING THE DISTORTION

Once all the "damaged .264 files" have been decoded and the result of each decoding has been saved in a text file, the distortion produced by each sequence of removed packets is calculated.

Therefore, a script called "measure_distortion.m" was created. This script takes the text file corresponding to the correctly decoded video sequence and the text file corresponding to one of the damaged video sequences and extracts the information of the PSNR (Peak SNR) of each sequence, concretely the one related to the luminance (therefore the YPSNR), by means of another script called "evaluate_psnr.m".

Then, it stores the YPSNR of the correctly decoded sequence in a vector and the YPSNR of the damaged decoded sequence in another vector. Moreover, it also calculates the difference between them, what will be treated in this project as "distortion". As a result, three vectors are obtained the length of which is equal to the amount of frames of the video sequence. Thus, each position of the vectors corresponds to a determined frame.

Then, as there are different "damaged .264 files", other script was created: "batch_measure_distortion.m". This script calls the text file of the correctly decoded video sequence and the text files of the damaged decoded sequences. For each of the last files, it calls the "measure_distortion.m" script and calculates the three previously mentioned vectors. The results are stored in a structure called "Quality" which has a length equal to the amount of "damaged .264 files".

3.4. DATA CORRELATION (I)

As in the last stage the information about the distortion produced by the removal of packets has been obtained, in this stage this information is matched with the features of the removed packets.

Therefore, first it is explain the steps followed in order to do the later, and then the results obtained for different types of scenarios.

3.4.1. MATCH FEATURES WITH DISTORTION

First, in order to obtain the characteristics of the removed packets, the script "observing_features.m" was created. This script takes as an input the structure PIC and the index of the removed packet and calculates per each first P frame (where the packet has been removed) the average characteristics of the removed packet. Therefore, it outputs a set of 6 different vectors:

- Average number of nonzero coefficients of the removed packet
- Average horizontal motion of the removed packet
- Average vertical motion of the removed packet
- Average "Skipped" value of the MBs of the removed packet
- Average "Mode" value of the MBs of the removed packet
- Average number of sMB with coefficients of the removed packet

So, if it is called “observing_features(PIC, 1)”, the outputted vectors will store per each position the average features of the first packet (the one removed) of each first P frame of the different GOPs of the encoded video sequence.

Then, in order to match the features of the packets removed with the distortion provoked by them, the script “analyse_max_distortion.m” was created. This script takes the created structures Quality and PIC and calls the previous script. Next, it matches the features of the lost packets with the distortion generated by them and outputs two graphics per each of the last vectors. The first one (which is represented in green) shows the amount of packets with a determined value of the feature and the second one (represented in pink) shows the average distortion caused by a determined value.

For instance, in the case of coefficients, the first graphic will show the amount of removed packets that have one specific average amount of nonzero coefficients and the second one will show the average distortion generated by the removal of packets with a determined average value of nonzero coefficients. For a more detailed account on the working of these scripts, go to Appendix A subchapter C.3.

3.4.2. ERROR AT PICTURE LEVEL

The first scenario to be analyzed corresponds to the “error at picture level” one. In this case, the aim is to analyze the distortion generated at picture level or in other words, the distortion generated in the P frame where a packet has been removed.

In order to do so, the video sequence is encoded with a GOP size equal to 2, therefore creating a pattern of “IPIPIPIPIP” frames. By doing this, the quality of the P frame in a GOP depends exclusively on the previous I frame and, hence, does not suffer from temporal error propagation.

Moreover, the analysis is made in the case of having small packets which means that the slices are composed of only 2 MBs. The reason behind this lies in the fact that by taking small packets composed of few consecutive MBs, which have very similar features; it is possible to guarantee that each packet has a distribution of features more or less constant. Therefore, the match between the features of the removed packets and the distortion generated by them is more accurate.

In the case of this scenario, two video sequences were analyzed and next the results for each of them are explained.

3.4.2.1. RESULTS FOR “FOREMAN.QCIF”

The video sequence “foreman.qcif” composed of 399 frames was encoded with a frame rate of 30 frames/s, slice size of 2 MBs, GOP size equal to 2 and QP for both I and P frames of 26. This parameter was set to a low value, which means that more nonzero coefficients remain after the quantization process (section 2.1.2), in order to establish a better relationship between the distortion and the number of nonzero coefficients of the lost packets.

Since the frame resolution being used is 144x176 pixels, which in turn corresponds to 9x11 MB, having a slice size of 2 MBs implies that each frame is subdivided into 50 slices or packets. Therefore, when calculating the distortion, 50 sequences of removal of packets were created; the first one corresponding to the removal of the first packet of each P frame (since there is only one P frame in each GOP) and the last one corresponding to the removal of the 50^{th} packet of each P frame.

Then, the features of the removed packets were matched with the distortion that they generated by means of the scripts mentioned at the beginning of this chapter. Next, some of the obtained graphics are shown:

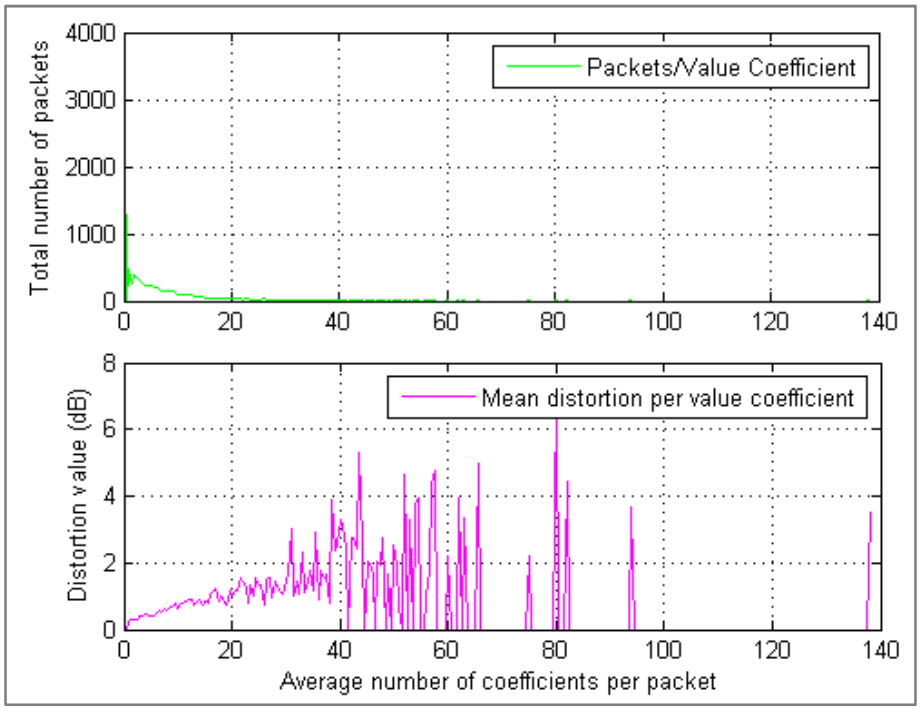


Figure 3.6 Distortion at picture level as a function of the average number of nonzero coefficients of the lost packets for "foreman.qcif"

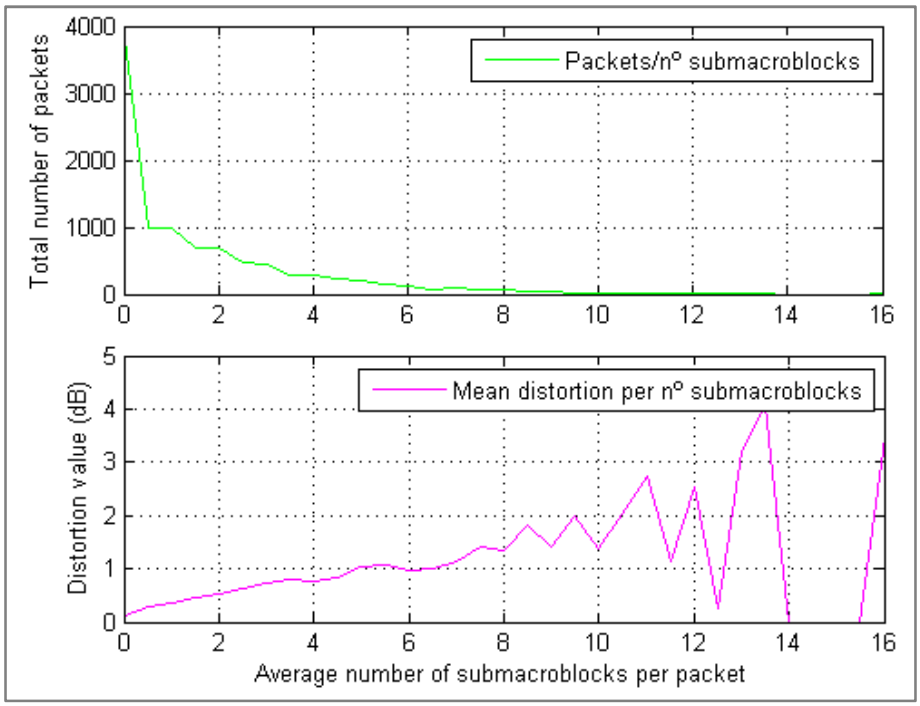


Figure 3.7 Distortion at picture level as a function of the average number of sMBs of the lost packets for "foreman.qcif"

Both, Figure 3.6 and Figure 3.7, show that most of the lost packets have a low number of nonzero coefficients and sMBs with nonzero coefficients which is shown by the green graphs. Then, the pink ones show that the relationship between the distortion and the

average number of nonzero coefficients and sMBs with coefficients has a growing tendency, which means that those packets with a high number of sMBs with coefficients, and therefore with a high number of nonzero tend to provoke higher distortion if they are lost.

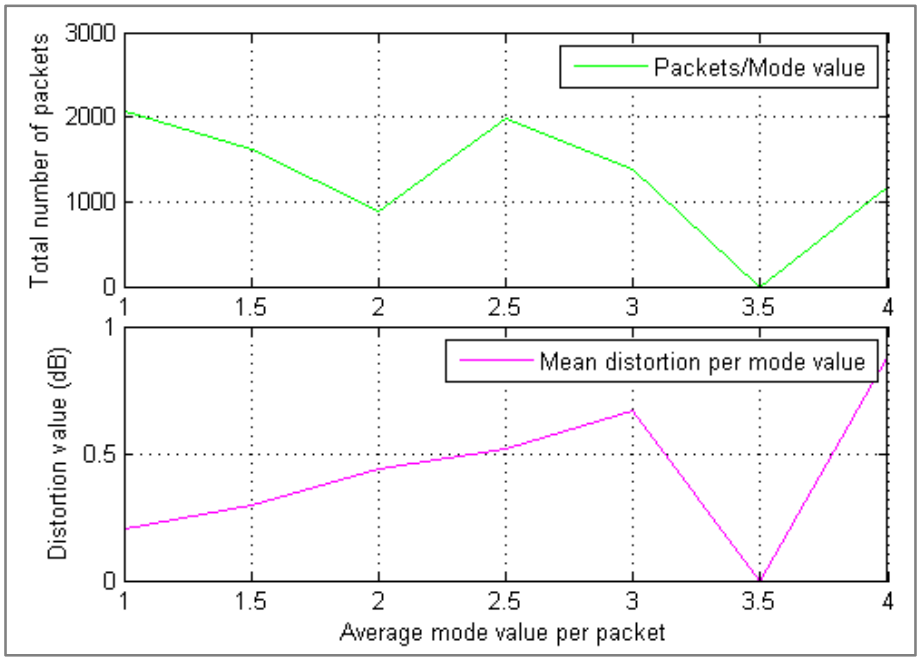


Figure 3.8 Distortion at picture level as a function of the average mode value of the lost packets for "foreman.qcif"

Then, taking a look at Figure 3.8 and considering the definition of mode (section 3.3.1), it can be appreciated that the distortion seems to increase in the case of removing those packets which are composed of MBs that have been subdivided. The more subdivided (higher mode value), the higher the distortion although in this case there is a sudden drop since there is no packet with an average mode value of 3.5 taking into account the possible values of the mode (subchapter 3.3.1) and that the packet size is of 2 MBs.

Furthermore, in the case of the average skipped value, Figure 3.9, it can be seen that removing those packets which are composed of MBs which are not skipped provokes higher distortion and that removing packets which are composed of skipped MBs seems to provoke no distortion which makes sense with what was expected. Nevertheless, it can be appreciated that the maximum distortion corresponds to 0.5 dB which is explained by the fact that the distortion represented is the average of the distortion of all the packets (see Appendix A subchapter C.3). Therefore, since most of the packets are not skipped, the whole distortion they provoke is averaged by a huge amount of packets having as a result that the maximum distortion per packet seems to be low.

This also explains the different distortion ranges in the previous figures.

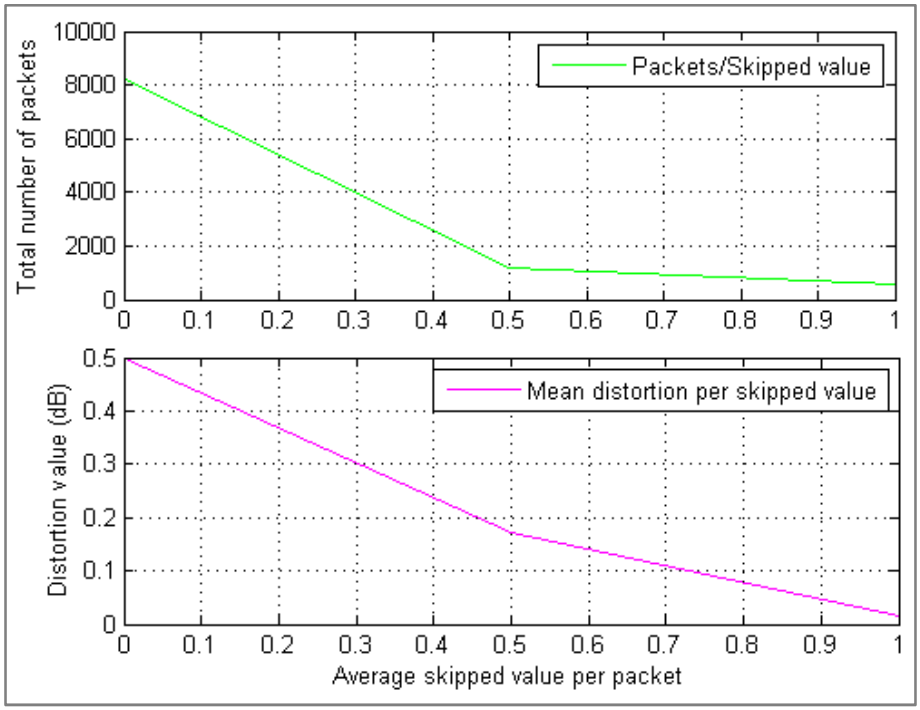


Figure 3.9 Distortion at picture level as a function of the average skipped value of the lost packets for "foreman.qcif"

Nevertheless, the information that these graphics provide is not enough since the video sequence analyzed does not have the amount of statistics needed. This can be easily observed by taking a look at Figure 3.6 and Figure 3.7 where there are some blank regions because, in these cases, there are no packets with a characteristic amount of nonzero coefficients or sMBs.

Consequently, a larger video sequence was analyzed in order to have packets with enough and characteristic statistics.

3.4.2.2. RESULTS FOR "COMBINED.YUV"

As it has been said in the previous section, the video sequence "foreman.qcif" did not have the amount of statistics needed for the analysis, therefore a new video sequence called "combined.yuv" was used. For the purpose of having a huge amount of statistics, this video sequence is a composition of three video sequences: "foreman.yuv", "carphone.yuv" and "mobile.yuv".

The video sequence, which is composed of 720 frames, was encoded with the same frame rate, slice and GOP size of the previous case, but this time the QP values for I and P frames were set to 24 which is a lower value than the one used in the previous case. Therefore, each frame is still divided into 50 packets.

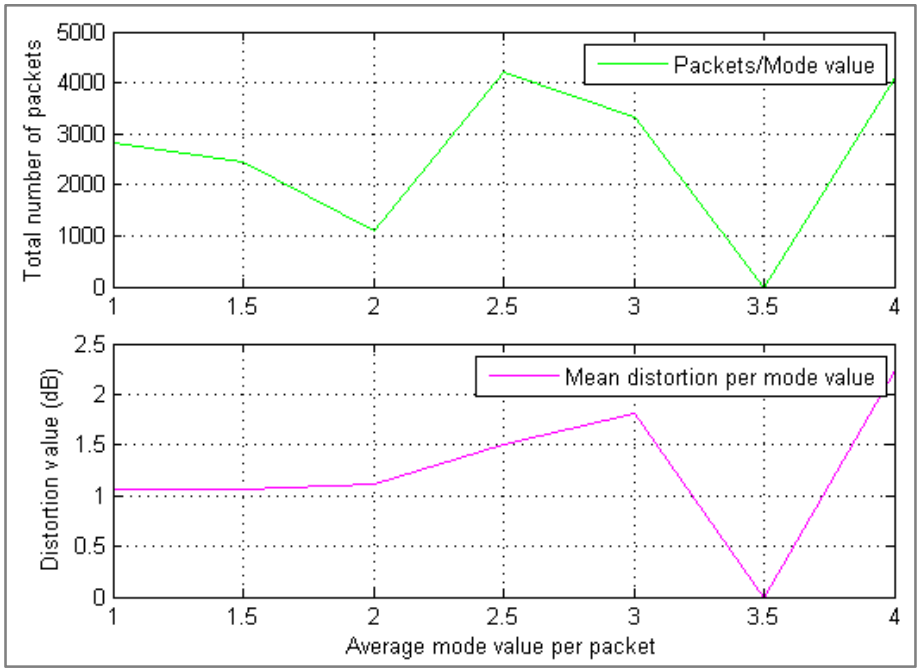


Figure 3.10 Distortion at picture level as a function of the average mode value of the lost packets for "combined.yuv"

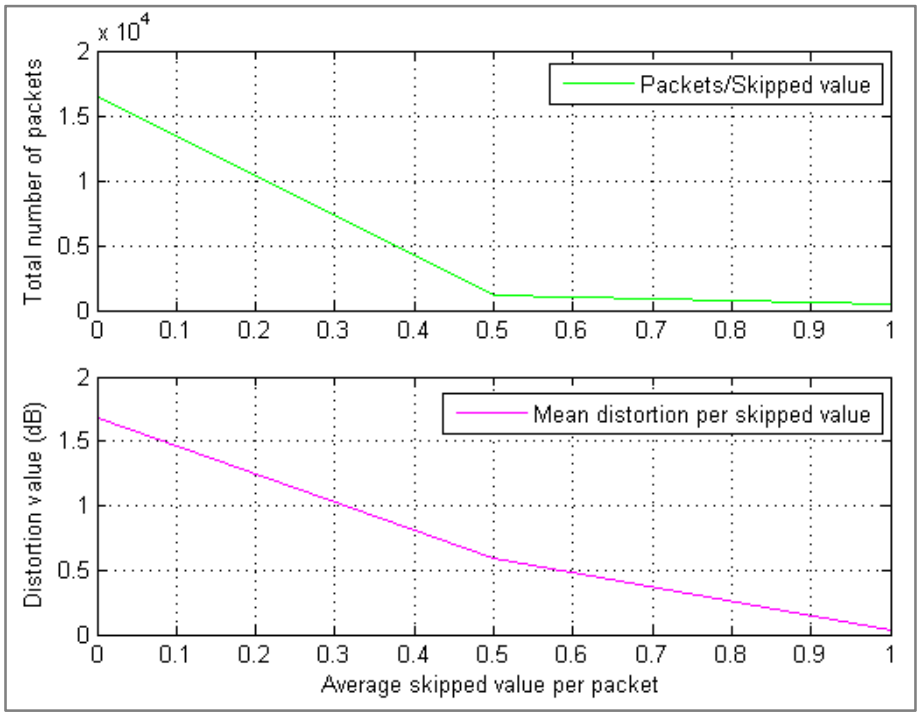


Figure 3.11 Distortion at picture level as a function of the average skipped value of the lost packets for "combined.yuv"

Taking a look at both Figure 3.10 and Figure 3.11, it can be seen approximately the same relationship between distortion and average skipped and mode value as in the case of the video sequence "foreman.qcif" (Figure 3.8 and Figure 3.9).

But, when analyzing the relationship with the number of nonzero coefficients and sMBs (Figure 3.12 and Figure 3.13), it can be appreciated that now there are no blank regions and the relationship seems more continuous. The hint obtained for "foreman.qcif" is confirmed

by the results of these two graphics since the distortion seems to increase when removing packets with high number of sMBs with coefficients and therefore, high number of nonzero coefficients.

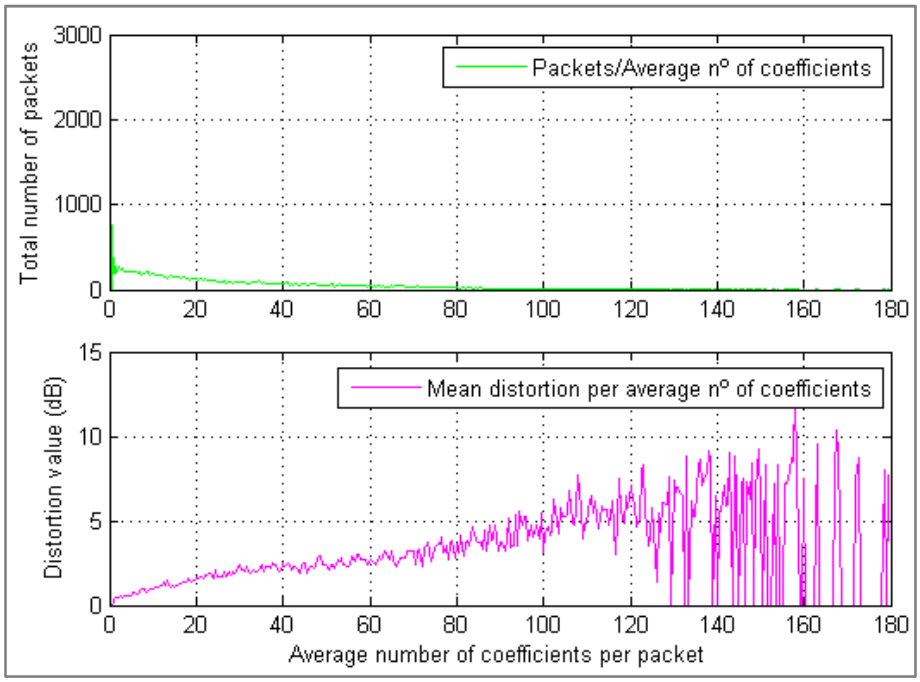


Figure 3.12 Distortion at picture level as a function of the average number of nonzero coefficients of the lost packets for "combined.yuv"

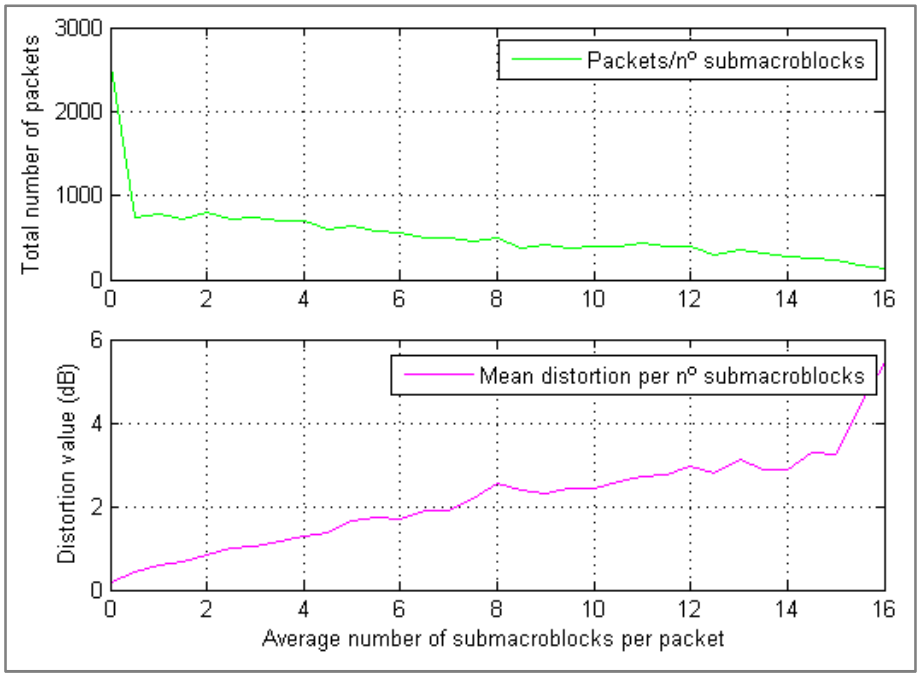


Figure 3.13 Distortion at picture level as a function of the average number of sMBs of the lost packets for "combined.yuv"

Now that some results were obtained in the case of analyzing the distortion at the frame or picture level, it was proceed to continue and analyze how the features of the lost packets could characterize the distortion caused because of the temporal error propagation.

3.4.3. TEMPORAL ERROR PROPAGATION RESULTS

In this case, the focus is set on studying the relationship between the features of the lost packets and the distortion in the case of the temporal error propagation in the video sequence. That corresponds to how the error propagates in time (frames) when there is more than one P frame per each I frame or, in other words, when the GOP size is bigger than 2.

In order to do that, a packet is removed from the first P frame of each GOP and then the error will propagate to the following P frames within the GOP due to bad reference. Therefore, the first P frame of the GOP is damaged by the removal of the packet, but the following ones become damaged because of bad reference (from the first P frame) that propagates to the rest of the P frames within the GOP because of the temporal error propagation.

Consequently, in this case, it is interesting to study the effect of the motion vectors in the measured distortion since they indicate to which sMBs of the previous frame the sMBs of the current frame are pointing to.

The analysis in this case is split in two cases. The first one corresponding to still having small packets, and the second one corresponding to simulating a loss of “real” packets.

3.4.3.1. SMALL PACKETS ANALYSIS

In this case, the video sequence “foreman.qcif” is encoded again with frame rate of 30 frames/s, slice size of 2 MBs, QP for both I and P frames equal to 24 but this time, the GOP size is set to 10. Therefore, each GOP will be composed of 1 I frame and 9 P frames.

Then, the processes explained in subchapter 3.3 were followed in order to create the structures PIC and Quality. Since the slice size is of 2 MBs, then each frame is still divided into 50 slices and therefore Quality is composed of 50 positions where the first one stores the results of having removed the first packet of the first P frame of each GOP and the final position corresponds to the results of having removed the last packet (the 50^{th}) of the first P frame of each GOP.

Figure 3.14 shows the distortion (in the figure depicted as “Difference”) in case of removing the third packet. It can be seen that the distortion at the beginning of each GOP is zero. This corresponds to the I frame that is error free. Then, in the first P frame, a packet is removed and the error is propagated to the rest of them. As a result, the distortion has a characteristic shape since when the packet is removed, the distortion within the GOP (in the other P frames after the first one) starts decreasing, increasing or remaining constant. Here in the example, it has been selected a GOP where the distortion decreases.

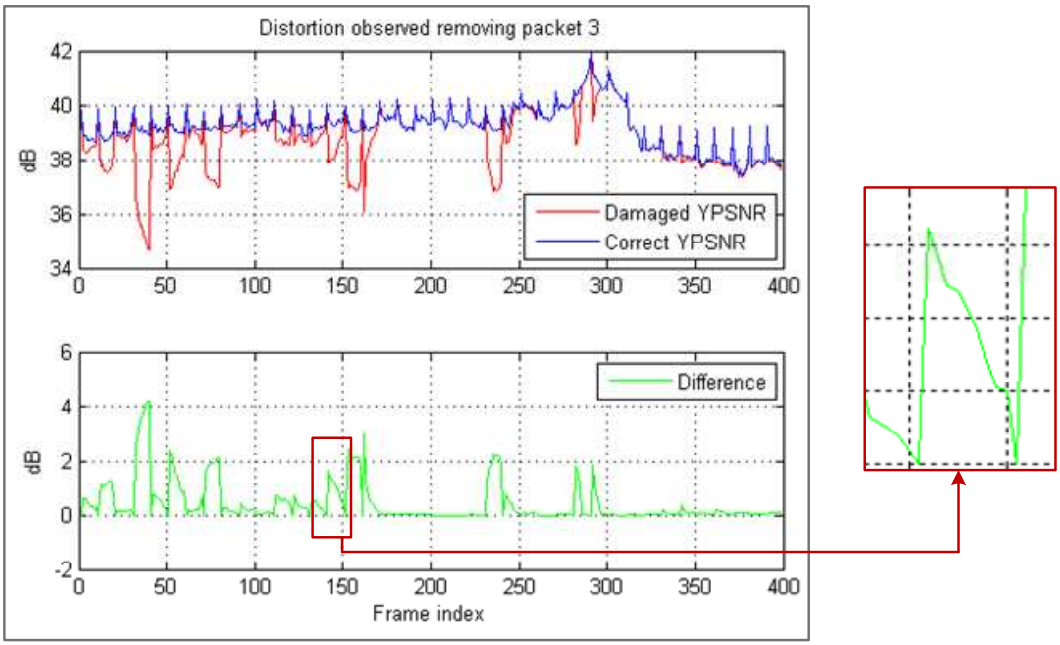


Figure 3.14 Observed distortion when removing packet 3 from "foreman.qcif"

Next, the relationship between the features of the removed packets and the distortion they provoke was calculated. But this time, instead of matching the features with the distortion in the first P frame, they were matched with the average distortion generated in the P frames of the different GOPs of the video sequence, obtaining as a result the following set of graphics.

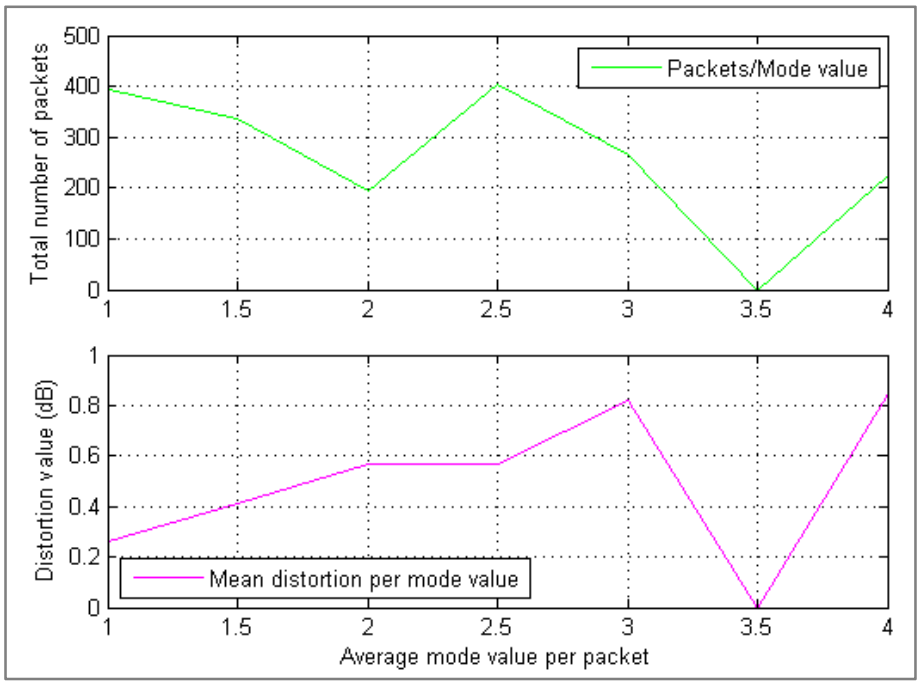


Figure 3.15 Average GOP distortion as a function of the average mode value of the lost packets for "foreman.qcif" (packets of 2 MBs)

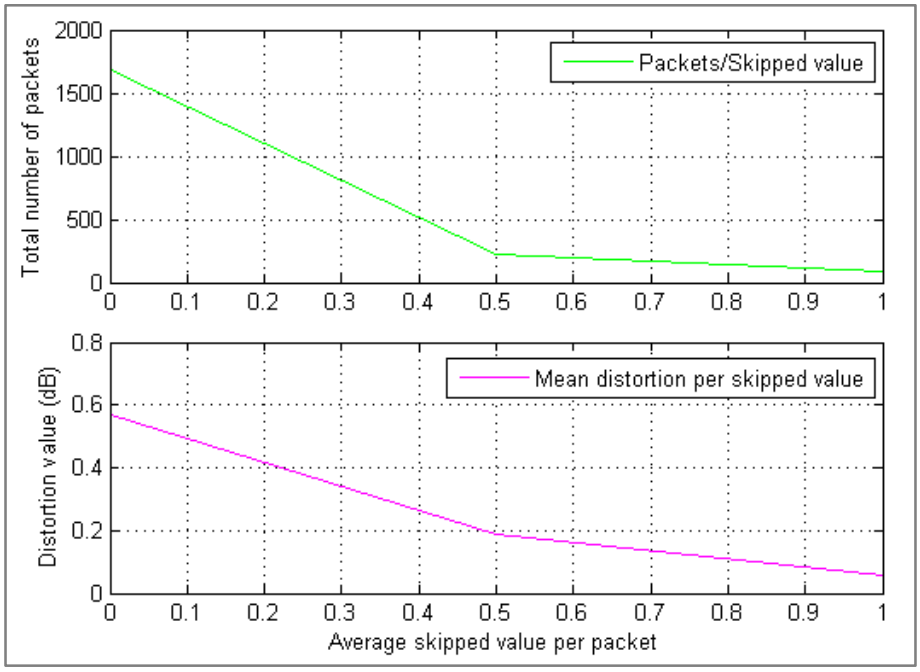


Figure 3.16 Average GOP distortion as a function of the average skipped value of the lost packets for "foreman.qcif" (packets of 2 MBs)

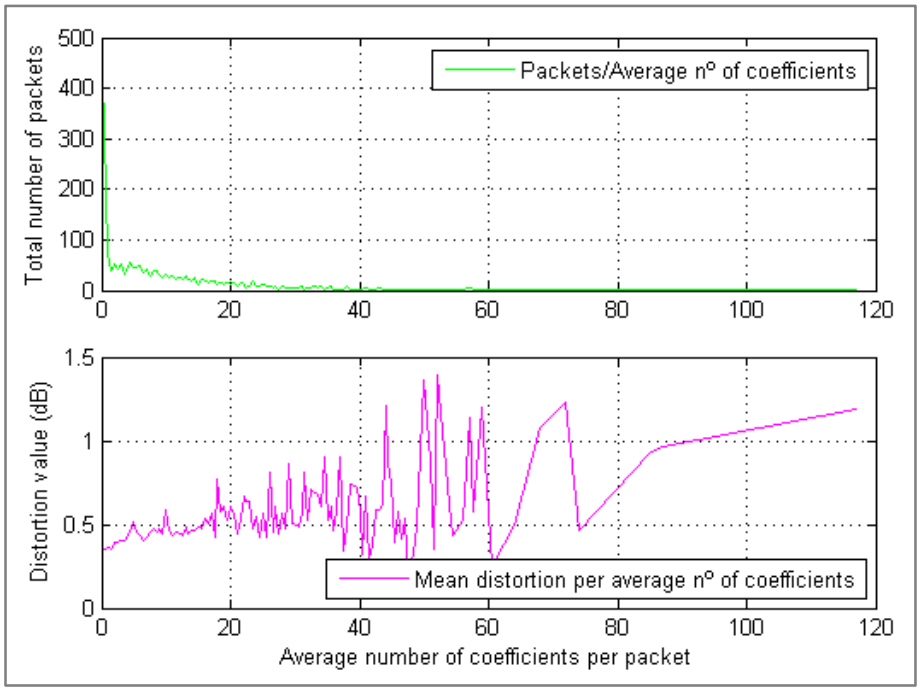


Figure 3.17 Average GOP distortion as a function of the average number of nonzero coefficients of the lost packets for "foreman.qcif" (packets of 2 MBs)

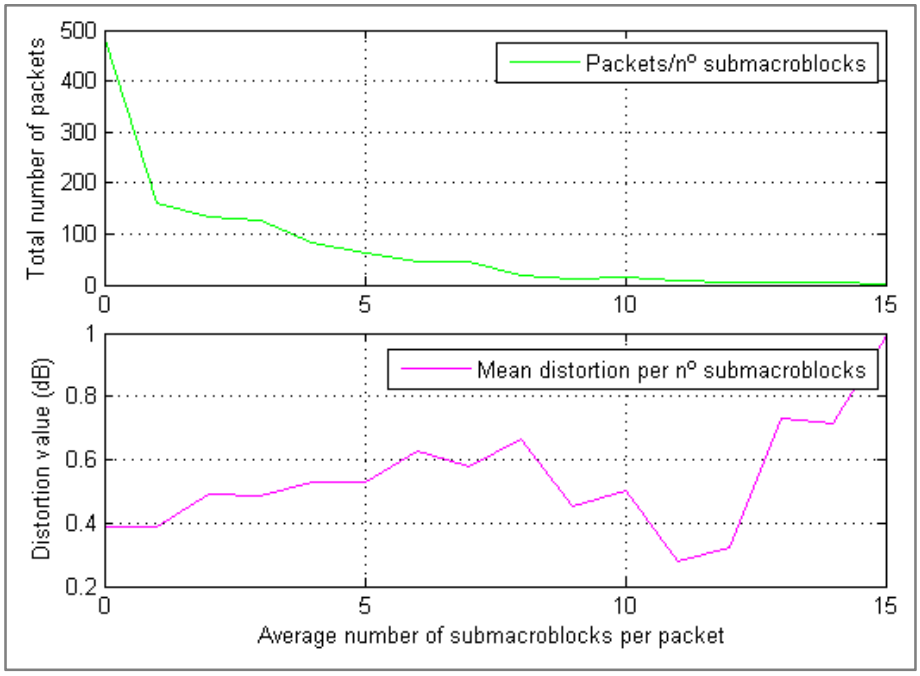


Figure 3.18 Average GOP distortion as a function of the average number of sMBs of the lost packets for "foreman.qcif" (packets of 2 MBs)

It can be seen that the results show that the distortion in each case follows more or less the tendencies observed in the case of GOP equal to 2 (section 3.4.2). However, the y axes show ranges of distortion lower than in that case since the distortion caused in each whole GOP is being averaged. In addition, still it can be appreciated that the statistics are not enough.

At the beginning of this subchapter, it is commented that it is possible the existence of a relationship between the observed distortion and the MV or, in other words, that the camera movement affects the obtained quality. Therefore, in order to check this, a MATLAB script called "ave_features_frame.m" was created. This script calculates the average motion of each frame in vertical and horizontal directions and compares it to the observed distortion for a determined packet. Figure 3.19 shows one of the observed results.

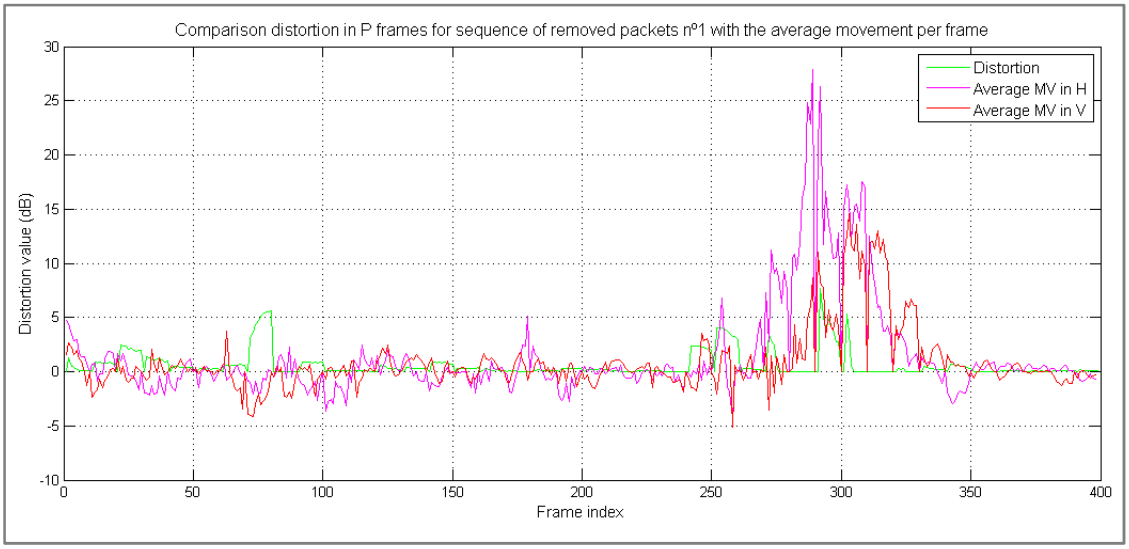


Figure 3.19 Comparison between distortion and movement when removing the first packet

By analyzing the relationship between distortion in the GOP and the movement of the frame (that is averaging the motion of all the MBs of the frame), it was possible to check that in the case of the packets corresponding to the borders of the frame (remembering that the packets are composed of 2 MBs), there was some relationship between the distortion measured in the reconstructed video sequence and the vertical camera movement calculated by means of the vertical motion of the MBs of the frame.

This can be explained by the fact that when there is a loss of packets in the borders of the frame (taking into account the small size of the packets) and the camera moves, then it is possible that the distortion:

- **Decreases:** When for instance the lost packets are in the upper border of the frame and the camera moves down. As a result, the sMBs of the lost packets are not used as a reference in the following frames and the distortion decreases.
- **Increases:** Following the same example, if the camera instead of moving down, moves up, as a consequence that new elements appear in the frame and the information of the border has been lost, the distortion starts increasing because the new part that appears cannot be correctly generated.

However, for those packets being in the middle of the frame, it was checked that the relationship between the vertical camera movement and the distortion was not as clear as in the previous case.

3.4.3.2. "REAL" PACKETS ANALYSIS

Now that some results have been obtained in the case of how the camera movement affects the obtained quality when studying temporal error propagation, it is proceed to study the relationship between distortion and the video features when having "real" packets.

Previously, the relationship between the video features and the distortion has been studied but in the case of having small packets which were composed of only 2 consecutive MBs. Therefore, there were packets with for instance an average number of nonzero coefficients equals to zero until packets with a high number of nonzero coefficients. Consequently, as each packet had specific features, it was easier to build the relationship.

However, in the environment in which this project is centered, packets usually are composed of a different amount of MBs larger than 2. Therefore, it will be difficult to have packets with a characteristic amount of features.

Generally, as the features of each packet are being averaged in order to do the analysis, there will be a huge amount of packets with more or less the same average features although this does not mean that the MBs within the packets have the same amount of features. For instance, there can be a packet the MBs of which have more or less the same amount of nonzero coefficients and another packet which have MBs with high number of nonzero coefficients and MBs with low number and it results that the average number of nonzero coefficients for both of them is approximately the same.

Therefore, in order to simulate a "real" loss of packets, now the packets of the video sequence will be composed of 11 MBs, which means that each frame will be subdivided in 9 slices and each slice (packet) will consist in a whole row of the frame.

So, for the same reasons as in the case of GOP size equal to 2 (section 3.4.2.2), the video sequence selected to do the analysis is "combined.yuv" instead of "foreman.qcif". It is encoded with the same parameters as "foreman.qcif" in the previous section but this time the slice size is changed to 11.

Then the procedures in order to create the structures PIC and Quality are followed. As a result, this time Quality will have only 9 positions since each frame is composed of 9 slices and therefore 9 sequences of packet removal were created. The distortion follows the shape showed in Figure 3.14.

Next, the relationship between the average distortion within the GOPs and the features of the removed packets was calculated.

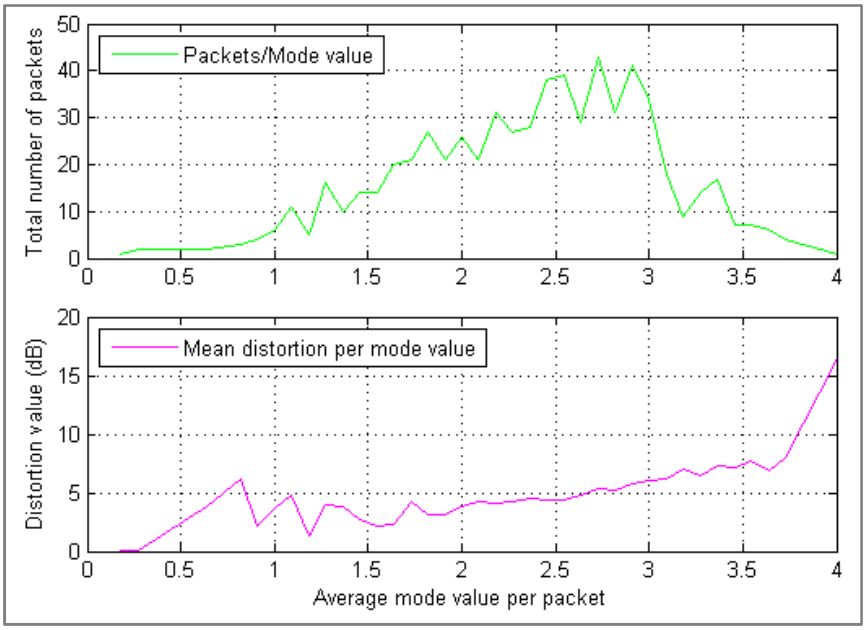


Figure 3.20 Average GOP distortion as a function of the average mode value of the lost packets for "combined.yuv" (packets of 11 MBs)

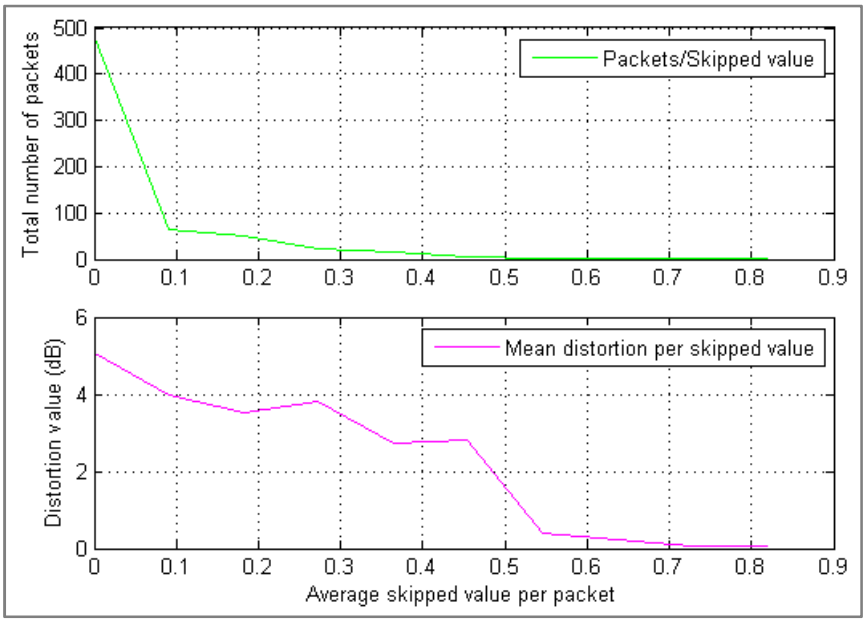


Figure 3.21 Average GOP distortion as a function of the average skipped value of the lost packets for "combined.yuv" (packets of 11 MBs)

Figure 3.20 and Figure 3.21 are a clear example that the possible average values of the features, x axes, have increased from a simple separation of 0.5 (the value of the features of a MB is an integer number, therefore when averaging packets of 2 MBs that is the corresponding obtained separation) to different values separation since now the packets are composed of 11 MBs.

In the green plots, it can be appreciated that the packets tend to concentrate in a determined region of the possible values of the feature. In Figure 3.20 it can be seen that most of the packets are concentrated within the region of [1.5 , 3] of the mode value. In the case of the skipped value, since it has been seen that few MBs are skipped taking into account the parameters used when encoding, therefore most of the packets will be composed of non-skipped MBs and few of them will have a skipped MB. This can be seen in Figure 3.21 where the packets concentrate on "0" and then the amount of packets that have a skipped MB decreases.

The distortion in these two cases (pink plots) shows what has been expected. Those packets that are composed of high subdivided MBs (high mode value) tend to create higher distortion than those which are composed of MBs not so subdivided (low mode value). Furthermore, those packets which have some skipped MBs provoke less distortion than those which are entirely composed of non-skipped MBs.

However, in the case of the average number of nonzero coefficients and sMBs, the amount of possible average values is much bigger than in the previous two cases. This can be seen as the high frequency that appears in the graphics of Figure 3.22 and Figure 3.23.

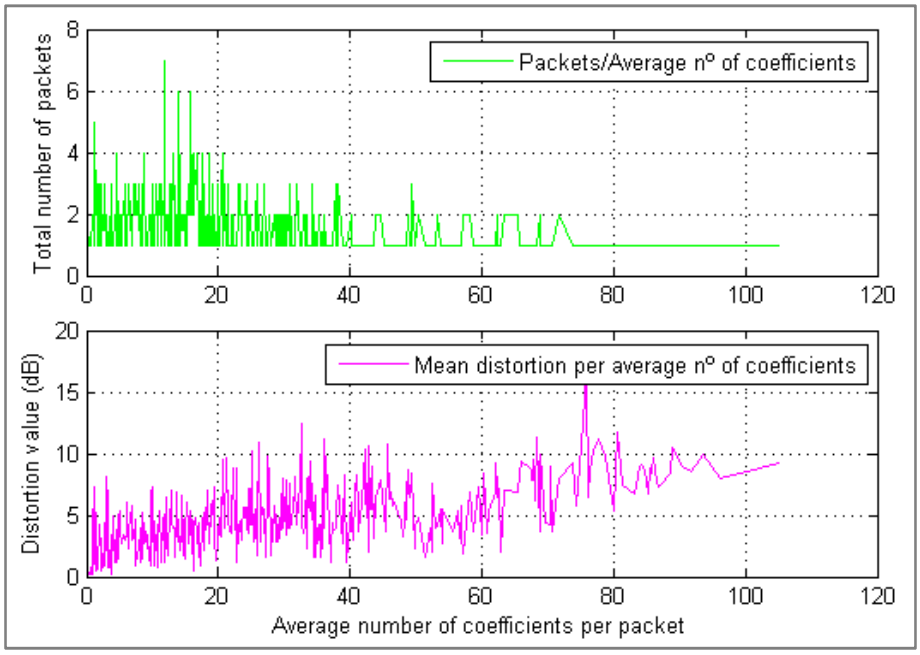


Figure 3.22 Average GOP distortion as a function of the average number of nonzero coefficients of the lost packets for "combined.yuv" (packets of 11 MBs)

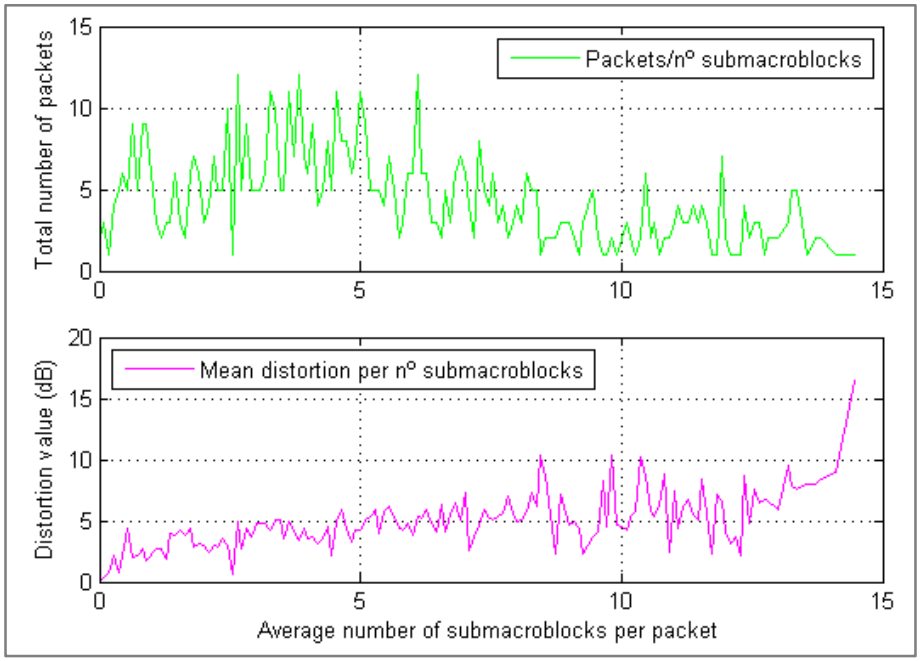


Figure 3.23 Average GOP distortion as a function of the average number of sMBs of the lost packets for "combined.yuv" (packets of 11 MBs)

Therefore, in order discard this high frequency, regions of average value of nonzero coefficients and sMBs with coefficients were created and the packets were grouped depending on the amount of nonzero coefficients and sMBs that they had. Then, the distortion of each region was calculated as the average distortion of all the packets of the region. Figure 3.24 and Figure 3.25 show the obtained results.

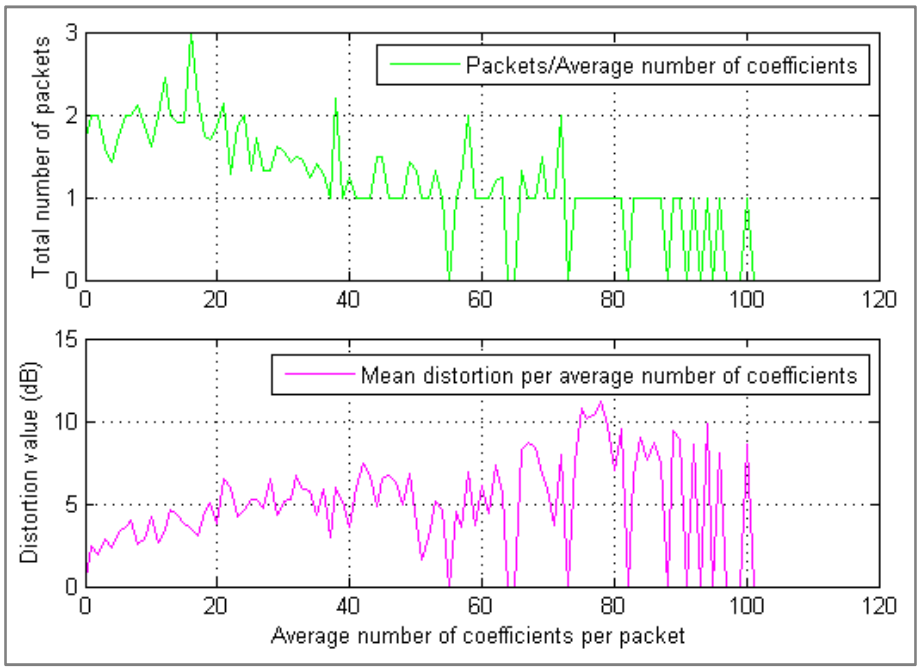


Figure 3.24 Average GOP distortion as a function of groups of number of nonzero coefficients of the lost packets for "combined.yuv" (packets of 11 MBs)

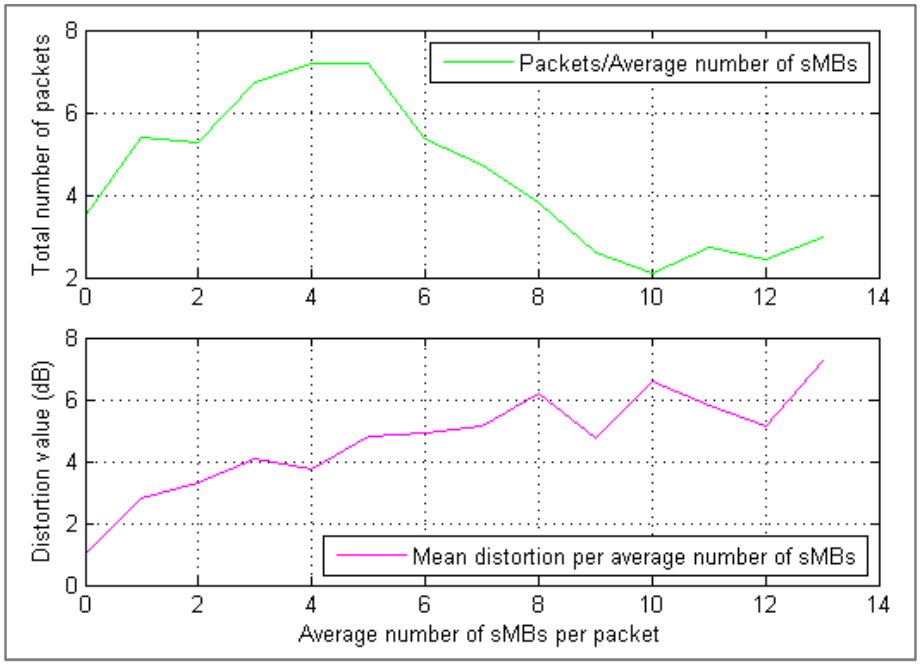


Figure 3.25 Average GOP distortion as a function of groups of number of sMBs of the lost packets for "combined.yuv" (packets of 11 MBs)

In Figure 3.24 it is appreciated that a huge quantity of the lost packets (green plot) has almost the same average amount of nonzero coefficients and they concentrate on a range of low – medium values of number of nonzero coefficients. The distortion (pink plot) in this case seems to be consistent with the previous cases and those packets with high number of nonzero coefficients tend to provoke higher distortion if they are lost.

The previous explanations are also suitable in the case of the number of sMBs with coefficients since they follow the same tendencies (Figure 3.25).

However, as it was expected, given the size of the packets, the distortion increasing tendency in these 2 cases is not as clear as in the case of GOP size equal to 2 (section 3.4.2.2).

3.5. Data correlation (II)

In the last subchapter it has been obtained an approximation of how the distortion is related to each of the different features of the packets of the video sequences for different types of scenarios (different GOP and packet sizes). Some graphics have been obtained which show how each of the features affect the distortion and in which measure, which has been useful in order to distinguish which features have a bigger effect on the distortion.

However, the aim of this project is to express the distortion as a function of all of them together and not to express the distortion as a function of each of them alone. Consequently, this chapter explains the steps followed in order to correlate the features of the packets with the distortion in the frame where the packet was lost or, in other words, with the distortion at picture level.

3.5.1. CORRELATION BETWEEN FEATURES

Before starting the correlation between all the features and the distortion, the video features have been correlated between them in order to understand better their relationship with the observed distortion.

Among all the features, the ones that are more likely to have a bigger effect over the quality of the decoded video sequence are:

- Amount of nonzero coefficients
- Amount of sMBs with nonzero coefficients
- Type of MB
- MB mode

It is expected that the type of MB would indicate the MB mode and how the MB has been subdivided. Then, depending on in how many sMBs the MB has been divided, it is possible to have an idea of the number of nonzero coefficients related to that MB since those MBs subdivided in many sMBs are likely to have a higher number of nonzero coefficients. Therefore, packets composed of those MBs are expected to have a higher effect on the quality if they are lost.

The interest of the correlation between those features lies in checking whether the latter is true. It is thought that once having an idea, it would be easier to understand how they affect the distortion.

In order to do that, a MATLAB script called “correlation_features_PIC.m” was created. This script takes the information of the structure PIC, concretely the features of the MBs of the P frames, since in this project encoded I frames:

- Are supposed to be scheduled more rarely than P frames and, thus, their number is much lower than the number of P frames
- Their statistics are much higher than the ones for P frames and therefore they would mask the features of the MBs of the P frames

Therefore, only the statistics of the P frames were considered for this analysis. Then, the script stores the previous features of each MB in 4 different vectors and plots them against each other, correlating therefore the data at MB level. Next, some of the obtained results are shown.

3.5.1.1. RESULTS

In this case, it is not important the GOP or the packet size since the correlation is being calculated at MB level. However, the following results correspond to the video sequence “foreman.qcif” in the case of a GOP size equal to 10 and packet size of 11 MBs and encoded with the same parameters used in section 3.4.3.2.

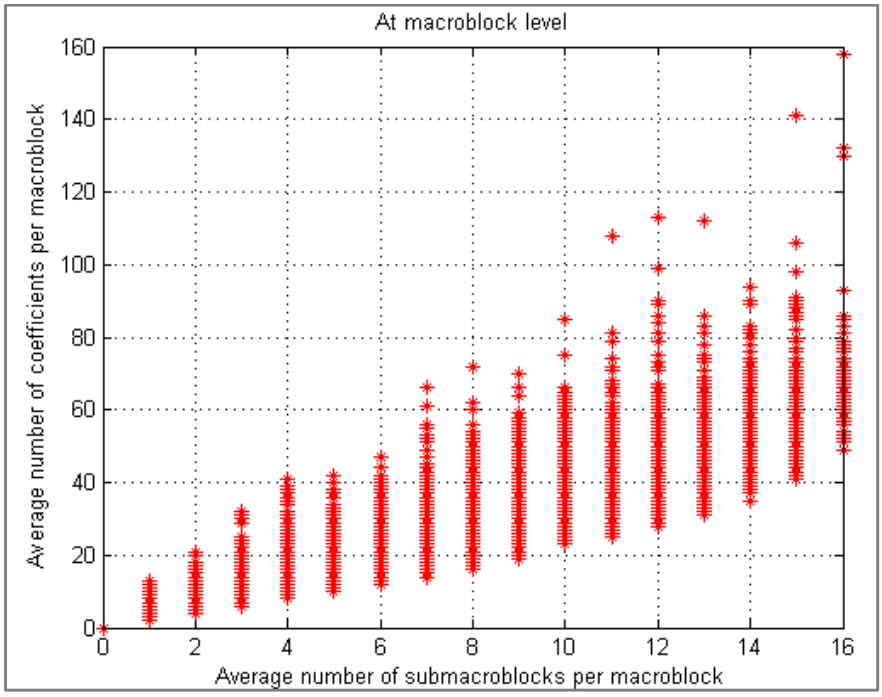


Figure 3.26 Correlation between the average number of coefficients and sMBs ("foreman.qcif")

It can be seen that there is some relationship between the number of sMBs with coefficients per MB and the average number of nonzero coefficients per MB. Figure 3.26 shows and increasing trend that follows more or less what was expected. However, the points do not seem very compacted. If they were highly compacted that would mean a high correlation between these two features.

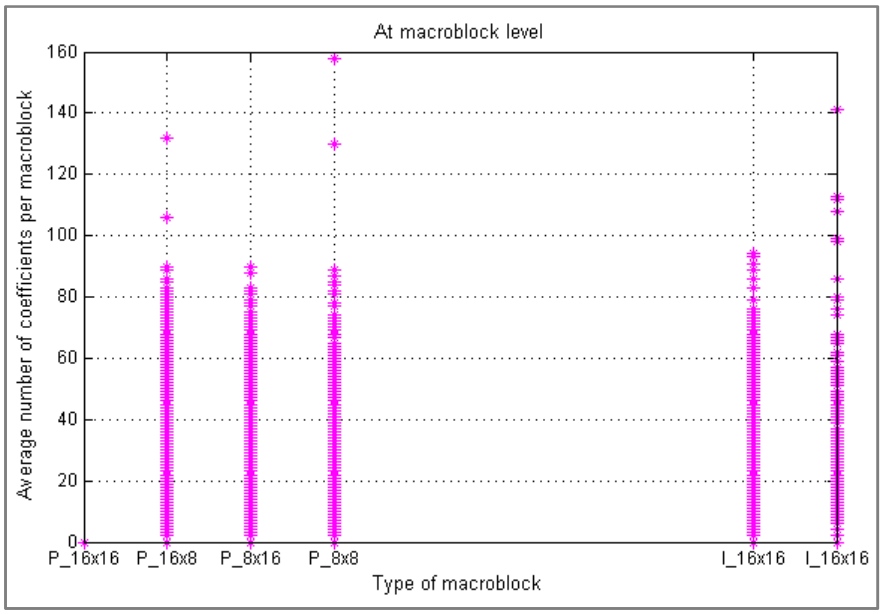


Figure 3.27 Correlation between the average number of coefficients and MB type ("foreman.qcif")

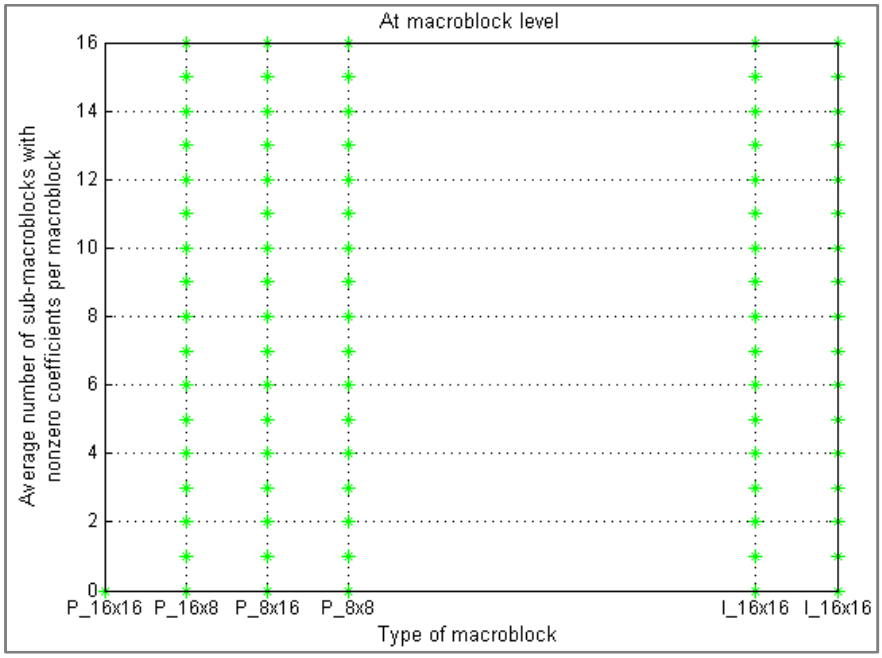


Figure 3.28 Correlation between the average number of sMBs and MB type ("foreman.qcif")

In Figure 3.27 and Figure 3.28, it can be seen that the points are within lines where each line corresponds to a category of the MB type. This indicates that, at least, each average number of sMBs or nonzero coefficients per MB is once within a MB type. It can be appreciated that the MBs of the P frames of the video sequence are of 6 different types. The first 4, which correspond to types [0-3], indicate how the MBs are subdivided and that they are "P" (Predictive) MBs. The last 2 types, which correspond to types [8-9], indicate that there are some MBs of the P frames which are "I" (Intra) MBs that in this case are not subdivided (16x16).

The reason behind the fact that there are I MBs within P frames is because the standard H.264/AVC allows that some MBs within P frames can be coded as I MBs instead of P. However, the process cannot be reversed or, in other words, it is not possible to have P MBs within I frames.

In the cases shown in Figure 3.27 and Figure 3.28, although types [8-9] correspond to Intra 16x16 MBs, they are coded with a different type of Intra mode prediction and luma and chroma block pattern (8).

In Figure 3.29, which shows the correlation between MB mode and the average number of nonzero coefficients, it can be also appreciated that each average number of nonzero coefficients is once within a MB mode. However, since the range of possible sMB values is much lower than the range of possible values of nonzero coefficients, therefore the result visible in the graph is that there are much more points and they are overlapped. This is also applicable to Figure 3.27.

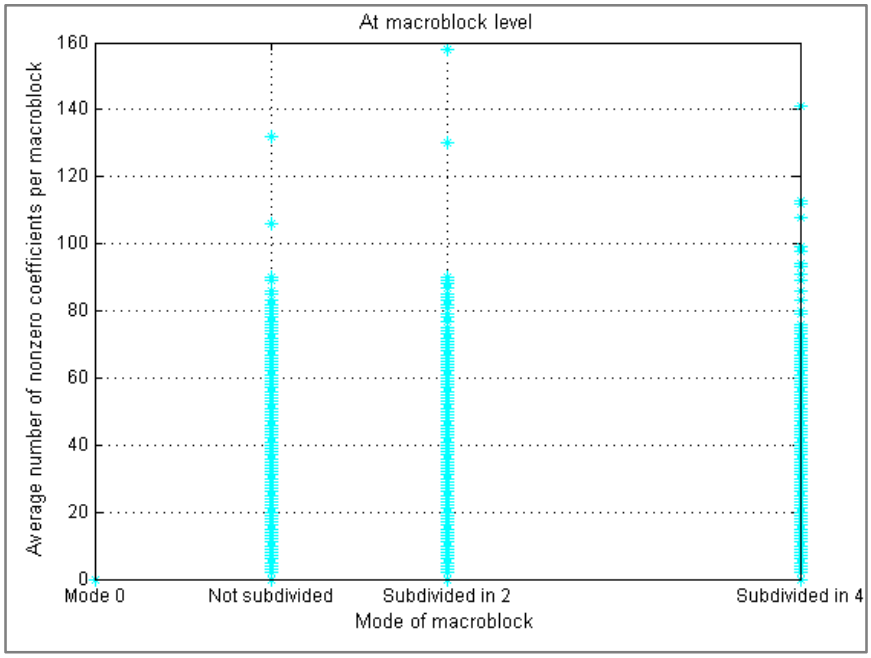


Figure 3.29 Correlation between the average number of coefficients and the MB mode ("foreman.qcif")

However, these graphs do not give any information of the density of the points. What is seen as a single point can be in reality different points overlapped. Consequently, the correlation of the features is represented by taking only the average point of each column which will help to show where the points concentrate.

Considering the correlation between the average number of nonzero coefficients and the average number of sMBs per MB (Figure 3.30), when taking the average value of each column, it can be appreciated that the cloud of points visible in Figure 3.26 has been reduced to a line which shows an increasing tendency. This result verifies what was expected since those MBs with more sMBs with nonzero coefficients were expected to have a higher number of nonzero coefficients.

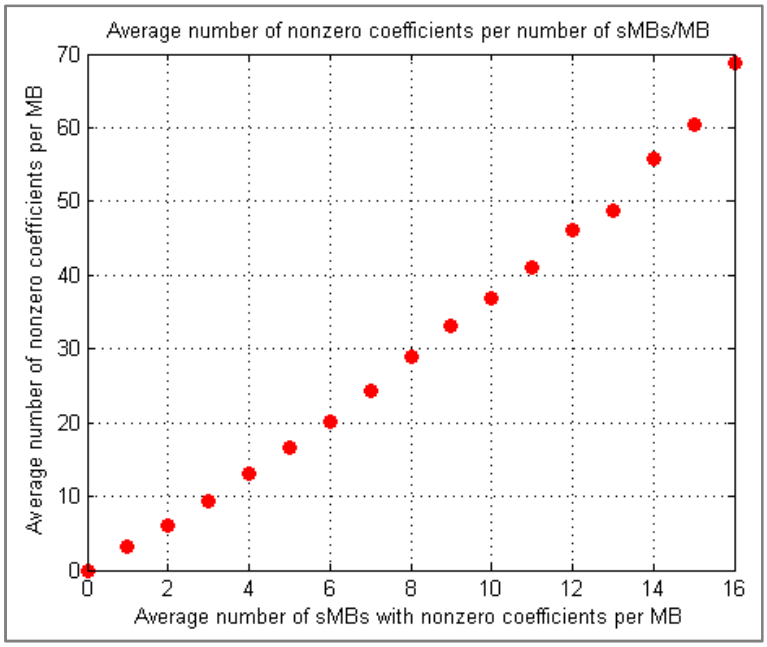


Figure 3.30 Correlation between the average number of coefficients and sMBs (Average)

Then, in the case of the correlation between the MB type and the average number of nonzero coefficients (Figure 3.31), it can be appreciated that the number of nonzero coefficients is lower in the case of having P MBs than when having I MB. This fits what was expected since the Intra mode implies the use of more nonzero coefficients than the Inter mode (the one used for P frames).

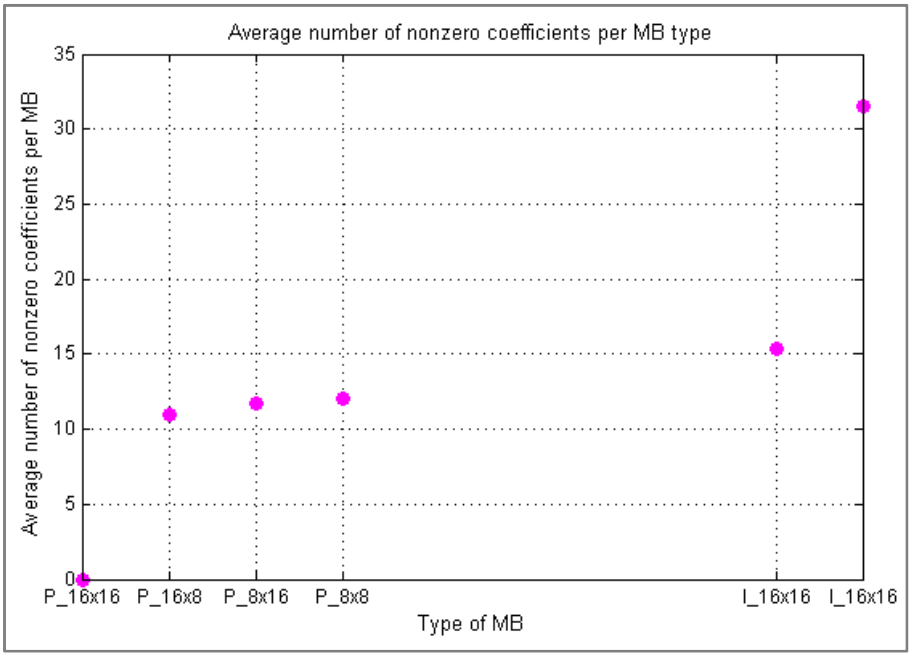


Figure 3.31 Correlation between the average number of coefficients and MB type (Average)

Since the relationship between the number of nonzero coefficients and the sMBs with nonzero coefficients per MB is very close as showed in Figure 3.30, therefore the results in the case of the correlation between the MB type and the number of sMBs per MB (Figure 3.32) are very similar to the ones showed in Figure 3.31. Here it can be also appreciated that P MBs have lower number of sMBs with nonzero coefficients that I MBs for the same reason as in the previous case.

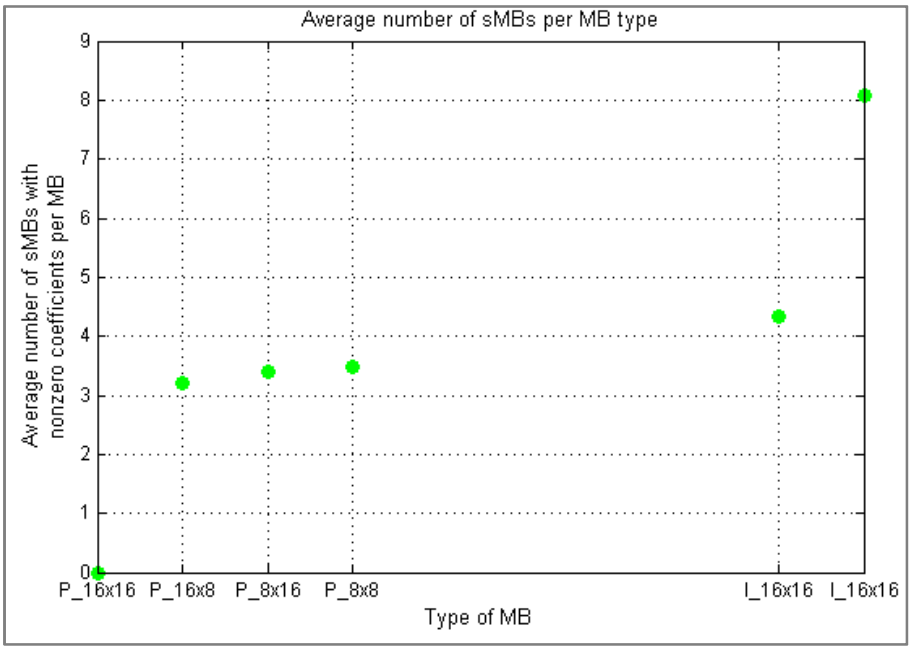


Figure 3.32 Correlation between the average number of sMBs and MB type (Average)

When analyzing the correlation between the number of nonzero coefficients and the MB mode, Figure 3.33 shows that those MBs being subdivided in more partitions tend to have a higher amount of nonzero coefficients. However, it can be seen that the slope of this increasing tendency is small compared to the increasing tendency showed in Figure 3.30.

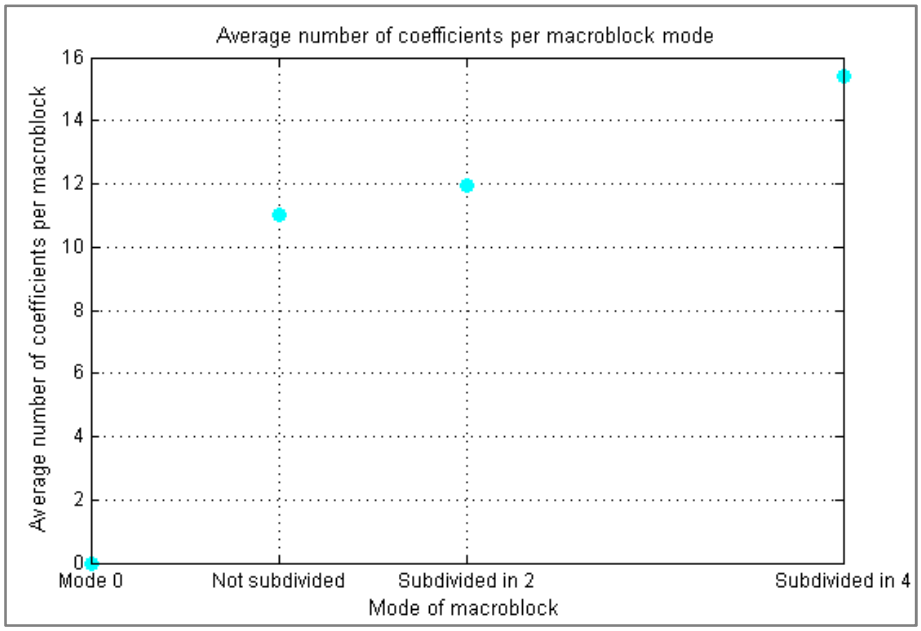


Figure 3.33 Correlation between the average number of coefficients and the MB mode (Average)

Finally, in the cases of Figure 3.31 and Figure 3.32 it can be appreciated that the number of nonzero coefficients and sMBs is zero for the type "P_16x16" which would correspond to type equal to zero. Furthermore, in Figure 3.33, it can be also seen a point that shows that the number of nonzero coefficients is zero for "Mode 0". These points correspond to MBs being skipped (mode and type equal to zero) and, therefore, their number of nonzero coefficients and sMBs containing nonzero coefficients is zero.

Now that some results have been showed in the case of the correlation between the features of the video sequence, next it is studied the correlation between the features of the lost packets and the distortion generated by them.

3.5.2. Features and distortion correlation

In the previous chapters it has been analyzed the relationship of the features of the video sequence between themselves and also the relationship between the features of the lost packets and the distortion they generate on the decoded video sequence. The results obtained have allowed having an idea of how each of the different features affects the distortion and which features of the lost packets seem to affect the distortion in a higher degree.

Therefore, the next step consists in estimating the distortion at picture level as a function of all the features of the lost packets. Although in the last section only 4 features of the video sequence were analyzed, it was decided to use 7 features in order to correlate the distortion with the features of the lost packets.

- Absolute vertical MV ("mvv")
- Absolute horizontal MV ("mvh")

- MB type (“type”)
- MB mode (“mode”)
- Number of nonzero coefficients (“coef”)
- Number of sMBs containing nonzero coefficients (“smb”)
- Skipped MB (“mb”)

Since, with the information available, it was not possible to determine a mathematical function that calculates the estimated distortion with the previous features as inputs; therefore, it was decided to estimate the distortion by means of interpolation.

However, doing an interpolation with 7 different variables is a complex process. Therefore, it was decided to reduce the dimensions of the original input data set of 7 variables to a data set of 3 variables which is an adequate number in order to carry out the interpolation process.

In order to reduce the dimensions of the original data set, a tool called “Principal Component Analysis” or PCA was used. This tool helps to extract information from complex data sets. The first descriptions of PCA were given at the beginning of the century 1900 and in the 1930s. Now, it is mostly used as a tool in exploratory data analysis and in the creation of predictive models. For more information about PCA, see Appendix D.

In this case, PCA was used in order to re-express the distortion as a function of a lower number of variables, 3 in this case, which would maintain the highest amount of information of the original data set of 7 variables.

Therefore, the PCA consists in re-expressing the original data set as a data set of new variables, which are called components. Components are uncorrelated between them and, furthermore, each component is a linear combination of the inputs of the original data set. Thus, the goal of this step was to re-express the original 7 input data set as a data set of 3 components which would maintain the highest variance of the original one.

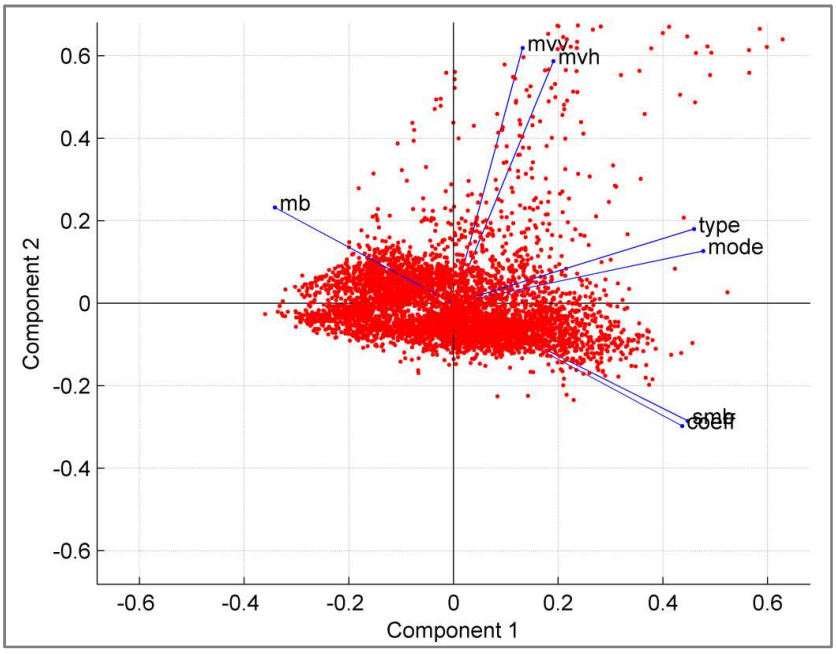


Figure 3.34 Example of PCA

Figure 3.34 shows an example of the PCA. In this example, PCA is being used in order to reduce the original input data set to just 2 components. The blue vectors represent the variance direction of each of the 7 variables and the red points represent the values of each of the different vectors. It can be appreciated that some of the original variables are highly correlated between them. For instance, the number of sMBs with nonzero coefficients per MB and the number of nonzero coefficients per MB are correlated.

Components 1 and 2 in this example are a linear combination of the 7 inputs and maintain the highest variance of the cloud of points which represents all the values within the 7 vectors.

Once the components were calculated, the coordinates of the input data set of 7 variables would be transformed to component coordinates, and the distortion would be estimated by means of interpolation.

The process of reducing the 7 input data set to 3 variables or components by calculating the PCA was simple. However, demonstrating that the obtained component set was correct was complex and could not be carried out completely taking into account the scheduled time for this project. Considering the last reasons and the fact that the PCA process was not the main scope of the project, it was decided to leave this PCA process for further investigation.

Therefore, in the following discussion, it was assumed that by means of the PCA it was possible to estimate the distortion at picture level, and it was estimated the evolution of the distortion in the following frames of the GOP because of the temporal error propagation. In order to estimate this evolution, it was taken the real distortion value in the frame where the packet was lost.

4. DISTORTION CHARACTERIZATION (II)

In chapter 3 it has been studied how the features of the lost packets affected the distortion measured in the decoded sequence. At the end of the chapter it has been explained the procedure followed in order to estimate the distortion at picture level by means of the features of the lost packets. Therefore, having the distortion in the frame where the packet is lost, this chapter is centered in analyzing and estimating how the distortion will evolve in the following frames until the end of the GOP.

4.1. INITIAL OBSERVATIONS

In the previous chapter, the analyses were done for specific packet sizes which were small compared to real packets. In this chapter, the analysis is carried out for real packets (with a size bigger than 11 MB) with a fixed size in Bytes which means a variable size in MBs.

From practical observations it was observed the existence of a relationship between the camera movement and the evolution of the distortion in the frames within a GOP. This relationship was first commented in section 3.4.3.1, although the effect of the movement was not as visible since the packets in that case were composed of only 2 MBs.

In the case that the packet on the top of the frame is lost, showed in Figure 4.1 by the red shape; when the camera moves down, it was observed that, as a consequence of the camera movement, the sMBs of the lost packet were not being used as reference in the following frames. As a result, the number of sMBs pointing to the lost packet decreased and, in addition, it was observed that the distortion within the GOP decreased too.

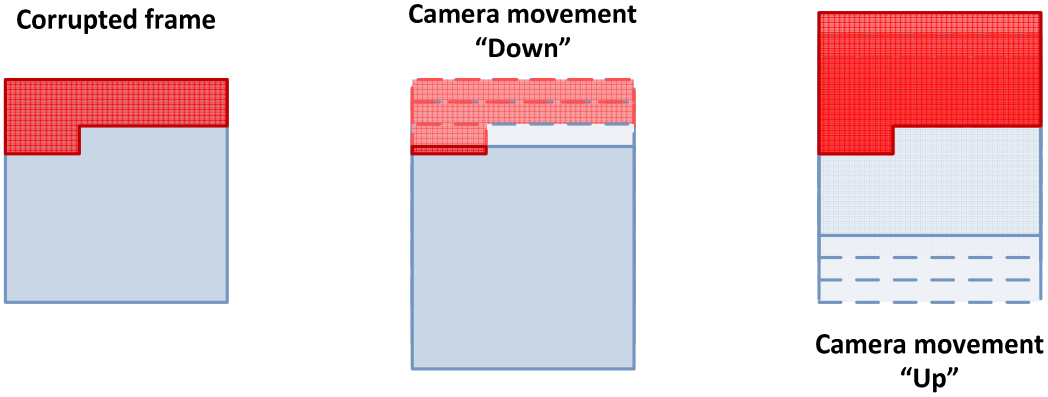


Figure 4.1 Distortion behavior within the GOP as a consequence of the camera movement (case 1)

However, if the camera moved up instead of moving down, it was observed that new information was appearing on the top of each new frame. This new information was not directly referencing the sMBs of the lost packet. However, the mechanism of the H.264/AVC standard implies that if a MB in a P frame appears in an area which was corrupted in the previous frame, the MB also becomes corrupted. Therefore, the new information appearing in each P frame, appeared in an area which was corrupted in the previous frame and, consequently, it was observed that in this case the distortion within the GOP increased.

In the case that the packet lost is in the bottom of the frame as showed in Figure 4.2, the camera movement affects the distortion within the GOP in the opposite way. When the camera moves down, as new elements appear in the end of the new P frames, these elements appear in the area where the packet was lost and therefore, they become corrupted and cannot be correctly reconstructed. Consequently, in this case, the distortion within the GOP was observed to increase.

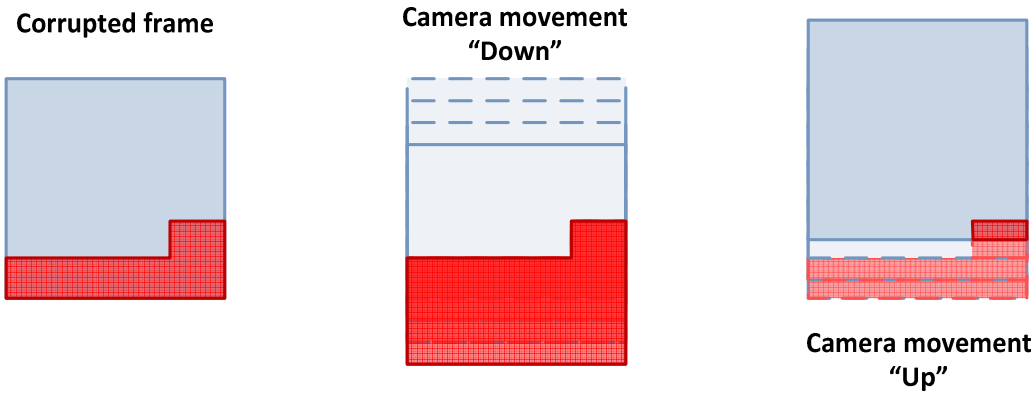


Figure 4.2 Distortion behavior within the GOP as a consequence of the camera movement (case 2)

However, when the camera moves up, the sMBs of the lost packet disappear gradually and, therefore, are not used as reference for the following P frames. It was observed that, in this case, the distortion within the GOP decreased.

As a conclusion of these observations, it can be seen that when the lost packet is on the top of the frame and the camera moves down or when the packet is in the end of the frame and the camera moves up; as a consequence of the camera movement, the number of sMBs of the P frames following the corrupted frame which are pointing to the lost packet decreases. In these cases, it was observed that the distortion within the GOP decreased.

On the contrary, when the lost packet is in the bottom of the frame and the camera moves down or when it is on the top of the frame and the camera moves up, it can be seen than in these cases the number of sMBs of the following P frames pointing to the lost packet increases. Furthermore, in these cases the distortion was observed to increase within the GOP.

Therefore, taking into account these observations, the relationship between the number of sMBs pointing to the lost packet and the distortion generated within the GOP is going to be analyzed in order to estimate the evolution of the distortion within the GOP.

4.2. Distortion and number of sMBs analysis

Taking into account the observations mentioned before, it was analyzed the evolution of the number of sMBs belonging to the P frames following the corrupted frame and which were pointing to the lost packet in order to match the results with the observed distortion within the GOP.

Figure 4.3 shows the general scheme of this analysis. In this case, once a packet is lost for instance in the P frame "f", it is needed to calculate how many sMBs in the P frame "f+1" are pointing to the sMBs of the lost packet in P frame "f". Then, for the P frame "f+2", it is calculated how many sMBs were pointing to the sMBs in P frame "f+1" which were pointing

to the lost packet. This operation is followed until the last P frame of the GOP which in Figure 4.3 corresponds to "h-1".

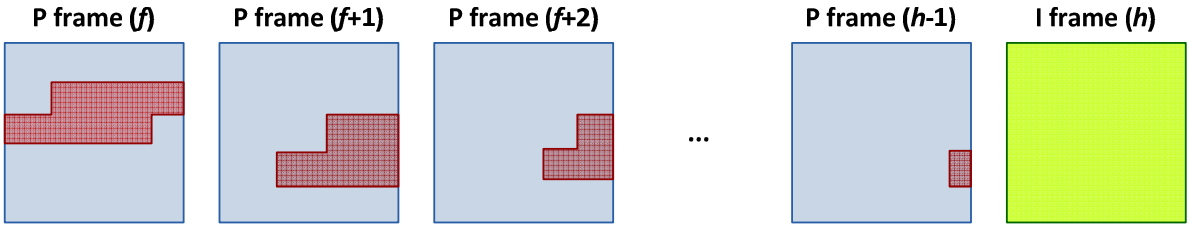


Figure 4.3 Scheme of the calculation of the evolution of the number of sMBs pointing to the lost packet

In order to calculate the amount of sMBs pointing to the corrupted sMBs in each P frame, the MV are used. By means of the MV it is possible to calculate the position to which a sMB is pointing in the previous frame since:

$$\text{Position sMB frame}(f) = \text{Position sMB frame}(f+1) + \text{MV sMB frame}(f+1) \quad (1)$$

Therefore, in order to calculate this evolution of the number of corrupted sMBs, a script called "generate_dist_movement_map.m" was created. But before explaining the working of this script, in order to provide an easier way to work with the MV, a script called "movement_pict.m" was created. This script takes the structure PIC and, per each frame, it calculates 2 arrays, one that stores the movement of the sMBs in vertical direction and the other one in horizontal direction. The dimensions of these 2 arrays correspond to the frame size in pixels, therefore 144x176.

Next, the script takes the position of each sMB and calculates its position in pixels. Then, the script calculates the movement of each sMB by dividing the absolute MV by 4 since the MV store the movement with a resolution of a quarter of a pixel and, taking into account the resolution of the 2 created arrays, it is needed the movement at pixel level. Finally, it stores the previous values in each corresponding array and in the pixels corresponding to the sMB.

As a result, this script outputs a structure called "movement", the length of which is the same as the amount of frames of the video sequence. Per each position (frame), the structure stores the 2 mentioned arrays that correspond to the movement of the sMBs of the frame. Furthermore, for each position, it stores a structure called "MB" with a size equal to the amount of MBs within the frame, which taking into account the frame resolution corresponds to 99. Then, for each position in "MB", it stores the position of each sMB within the frame and also its movement in pixels.

Once created the structure "movement" for the video sequence, it is called the "generate_dist_movement_map.m" script. This script takes the previous structure together with PIC, Quality, the GOP size and the index of the frame where the packet is lost as well as the index of the removed packet. Then, by taking the last two indexes, it reconstructs the position of the lost packet within the frame. In order to represent it, the script creates an array with a size 36x44 (which corresponds to the frame size but in sMBs instead of pixels) and represents the non-corrupted sMBs as 0 and the corrupted ones as 1.

It is important to mention that, unlike in the previous chapter in which the packet was lost in the first P frame of the GOP; in this case, the packet can be removed from any frame within the GOP. In other words, the script calculates the evolution of the sMBs for any

frame within the GOP where a packet has been lost, except the last P frame since there is no evolution of sMBs to be calculated.

Once the lost packet has been reconstructed, for the following P frame, the script calculates 2 movement arrays with 36x44 size. These arrays or maps will store the vertical and horizontal movement of the sMBs of the frame being analyzed and, to create these maps, the structure "movement" is used. Furthermore, a third map is created which will store the number of nonzero coefficients of each sMB of the frame. The importance of this map will be explained afterwards. In order to create these maps, the script "create_mv_map.m" was created. For an extended account on the working of this script, see Appendix A subchapter C.4.

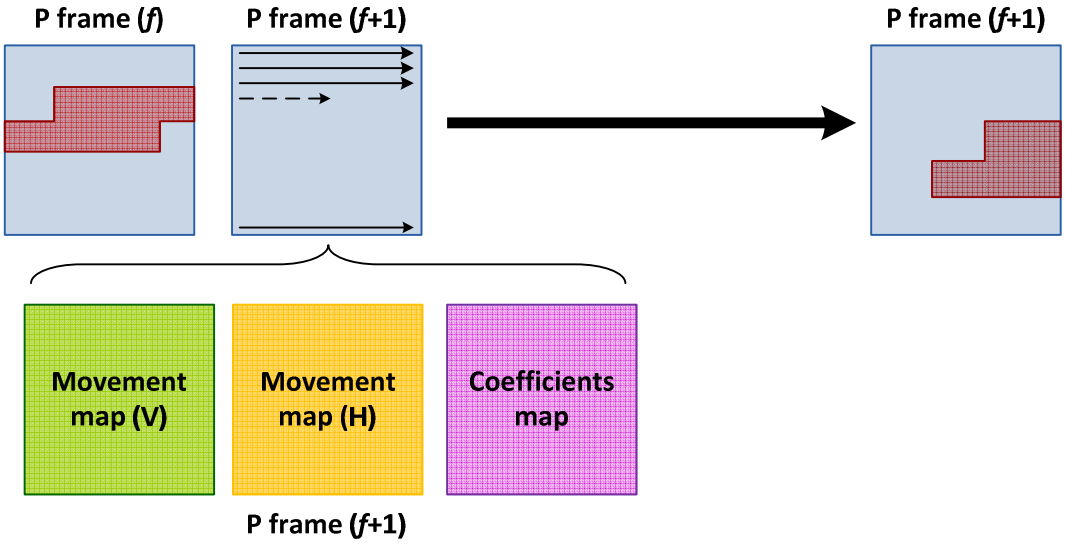


Figure 4.4 Process of calculating the amount of sMBs pointing to the corrupted sMBs in the previous frame

As Figure 4.4 shows, once the three maps have been created, the P frame being analyzed is scanned. By taking the position of each sMB and its movement, which is stored in the movement maps; it is possible to calculate to which position the sMB is pointing in the previous frame by means of equation (1). Therefore, by doing this, it is calculated the amount of sMBs that are pointing to the corrupted sMBs in the previous frame.

To carry out this process, the "cal_new_map_variable.m" script was created. This script takes the 36x44 array representing the corrupted frame, the movement maps and the PIC structure.

As a first step, the script creates a map corresponding to the frame being analyzed. Therefore, it creates a 36x44 array with all positions which represent the sMBs initialized to 0 (non-corrupted). For easier understanding, this array will be called "new frame".

Next, for each sMB, the script calculates the position to which the sMB is pointing in the previous frame. However, sMBs are not pointing concretely to a determined sMB position, but they can be pointing to an area shared by more than one sMB. Consequently, a threshold of 50 % was taken. Figure 4.5 shows the working of this approach.

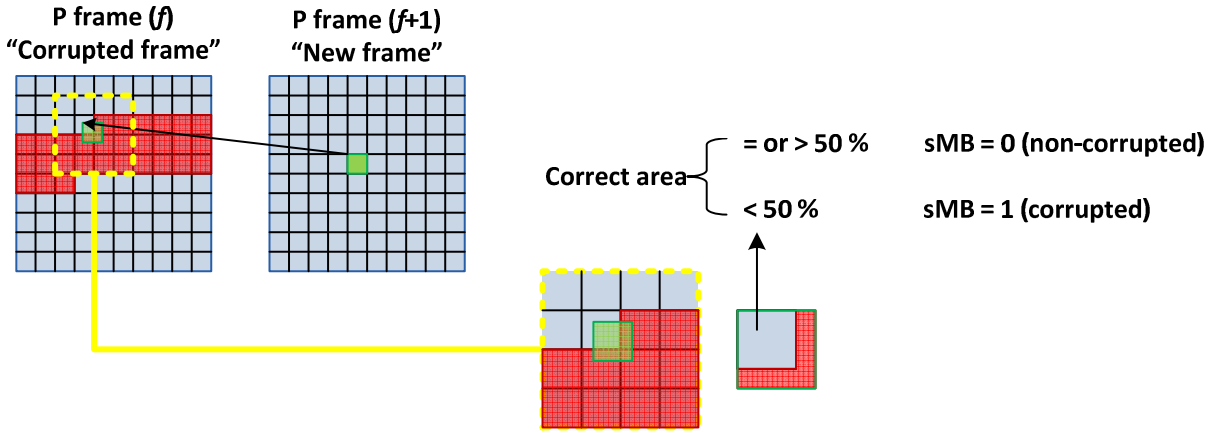


Figure 4.5 Process to decide whether a sMB becomes corrupted or not

In order to decide whether a sMB is corrupted or not, it is calculated to where the sMB is pointing. If the sMB is pointing to the position of a single sMB, then if this sMB is corrupted in the corrupted frame (previous frame), a "1" is stored in the "new frame" in the position corresponding to the sMB being analyzed. If the sMB to which is pointing is non-corrupted, then nothing is done since all the positions of the "new frame" are initialized to 0.

However, in the case in which the sMB is pointing to an area shared by more than one sMB, if some of these sMBs are corrupted and the others not, the next procedure is followed. If more than 50 % of the area of the analyzed sMB is within a corrupted zone in the corrupted frame, then the sMB is considered corrupted and a "1" is stored in its position in the "new frame".

This process is followed for all the sMBs of the frame being analyzed and, as a result, the script outputs the "new frame" where all the sMBs pointing to the corrupted sMBs in the previous frame (corrupted frame) are indicated by "1".

Next, the general script ("generate_dist_movement_map.m") takes this map and calculates the total number of sMBs that have become corrupted. In addition, the coefficients map is used in order to calculate the number of nonzero coefficients contained in these sMBs. It has been thought useful to calculate also the evolution of the number of nonzero coefficients contained within the corrupted sMBs because it is thought that it can also have an effect on the evolution of the distortion.

The process is repeated for all the following frames until the end of the GOP. As a result, 2 vectors are obtained. The first one stores per each position the amount of sMBs pointing to the corrupted area in the previous frame and the second vector stores, per each position, the amount of nonzero coefficients contained within these sMBs. Therefore, these vectors store the evolution of these 2 variables within the GOP. Finally, they are represented against the distortion in order to study if there is a relationship between them.

4.2.1. Results

4.2.1.1. Results for "foreman.qcif"

In order to check the correct working of the created scripts, before encoding a new video sequence with a real packet size, it was decided to use the already calculated and stored data for the video sequence "foreman.qcif".

Therefore the data created for the analysis with a GOP size of 10 and packet size equal to 11 was used (section 3.5.1.1). For the analysis purposes a GOP with high movement was selected. In this case, in the selected GOP, the camera is moving down. When removing the first packet in the first P frame of the GOP, Figure 4.6 shows the obtained results.

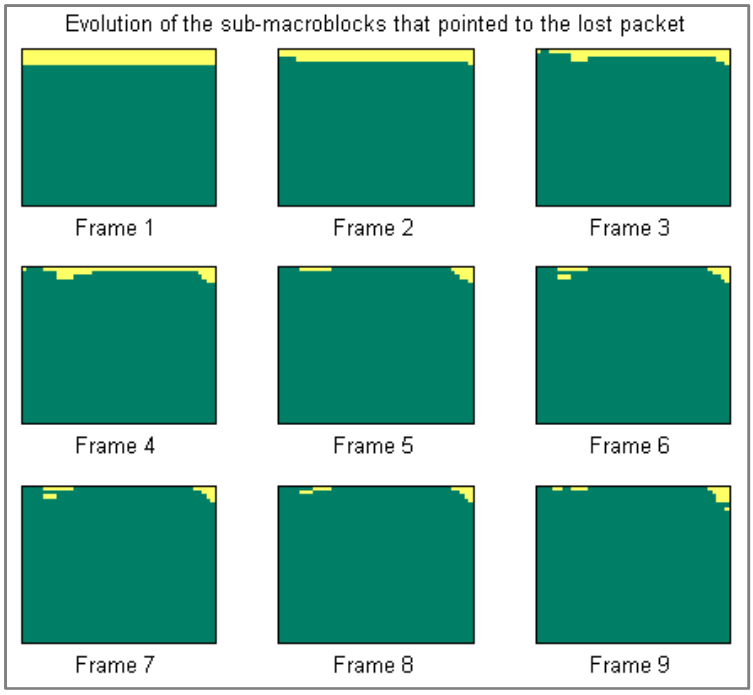


Figure 4.6 Evolution of the corrupted sMBs ("Foreman.qcif" first packet)

In this example, "Frame 1" represents the frame where the packet is lost, which is depicted as yellow. It can be appreciated that, taking into account that the packets are composed of 11 MB, the first packet corresponds to the first row of the frame. As was expected (Figure 4.1), as a consequence of the camera movement, the corrupted sMBs of the lost packet are not used as reference for the following P frames of the GOP and, as a result, the number of corrupted sMBs within the GOP decreases.

When matching this evolution of the corrupted sMBs with the real distortion in the GOP, it is confirmed that the distortion within the GOP decreased too. This can be seen in Figure 4.7 where the distortion is represented in blue and the evolution of the number of corrupted sMBs is represented in green. In this graph, the first position represents "Frame 1" and the last position represents "Frame 9". It can be appreciated that the high decreasing slope of the sMBs corresponds to the region where the distortion drops off.

Figure 4.8 represents the evolution of the nonzero coefficients (green line) against the distortion (blue line) within the GOP. In this case, the first position has no coefficients since it corresponds to the frame where the packet is lost and, consequently, there is no information available for this packet. Therefore, the number of coefficients is 0. Since the number of corrupted sMBs decreases rapidly, the number of nonzero coefficients decreases too. Therefore, it can be concluded that since both of them decrease, the distortion within the GOP decreases too. However, it has to be mention that although it is supposed that the coefficients have an effect on the evolution of the distortion; it is believed that its effect is smaller than the effect of the sMBs. Coefficients can have an effect but the behavior of the distortion is believed to be explained mostly by the evolution of the sMBs.

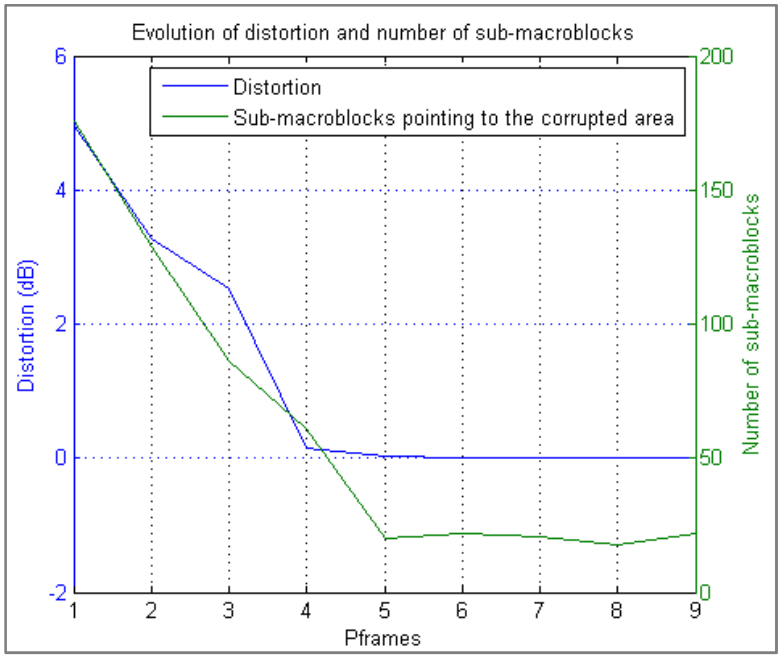


Figure 4.7 Relationship between the distortion and the corrupted sMBs ("Foreman.qcif" first packet)

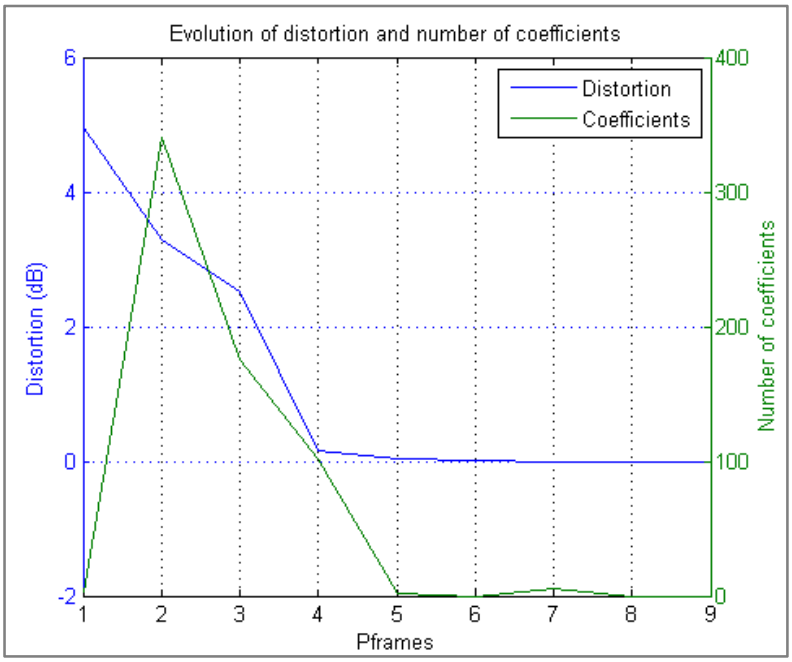


Figure 4.8 Relationship between the distortion and the nonzero coefficients ("Foreman.qcif" first packet)

Figure 4.9 shows the results in the case of removing the last packet in the first P frame of the GOP. It is observed that the amount of sMBs becoming corrupted increases as was expected taking into account Figure 4.2. This can also be seen in Figure 4.10 by the green line. It is appreciated that the distortion in the case of removing that packet (blue line), as was expected, follows an increasing behavior.

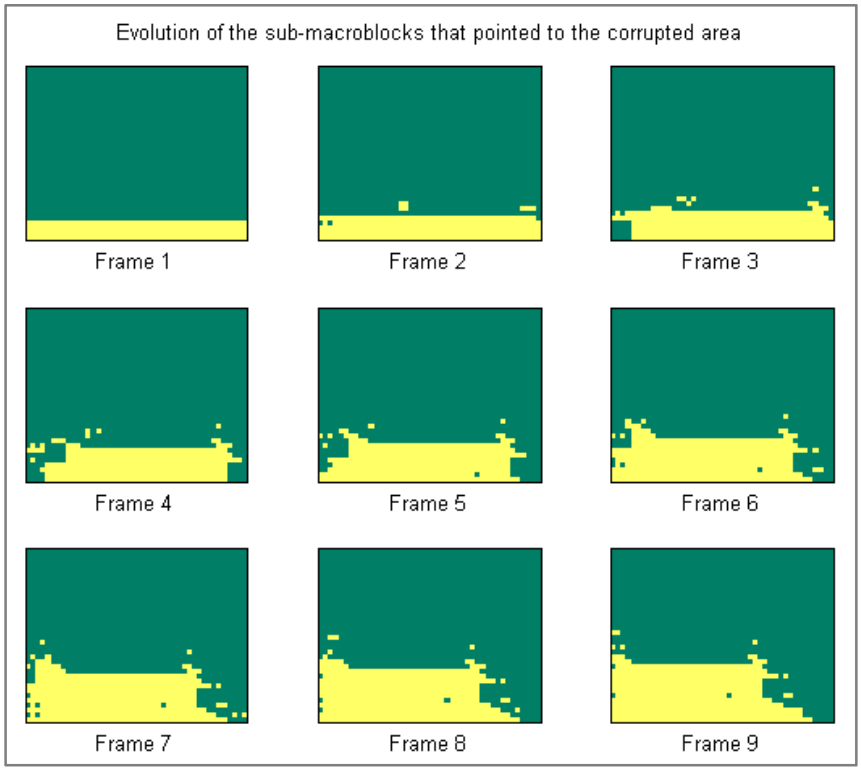


Figure 4.9 Evolution of the corrupted sMBs ("Foreman.qcif" last packet)

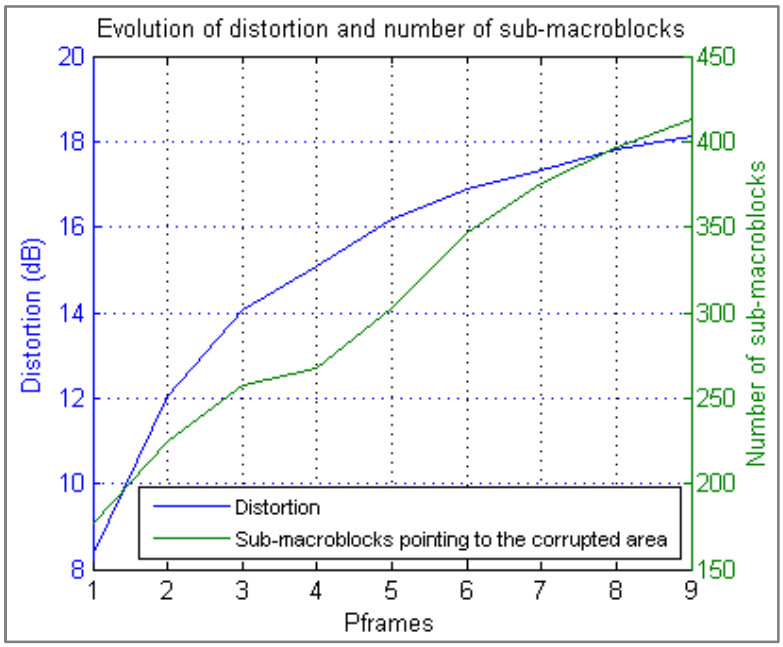


Figure 4.10 Relationship between the distortion and the corrupted sMBs ("Foreman.qcif" last packet)

In the case of the evolution of the nonzero coefficients contained within the corrupted sMBs, Figure 4.11 shows that their number decreases within the GOP. This can contribute to make the distortion increase within the GOP since more sMBs are becoming corrupted but the number of nonzero coefficients applied to them decreases, having as a result that the prediction of the sMBs in the current frame relies more on the reference, which is corrupted.

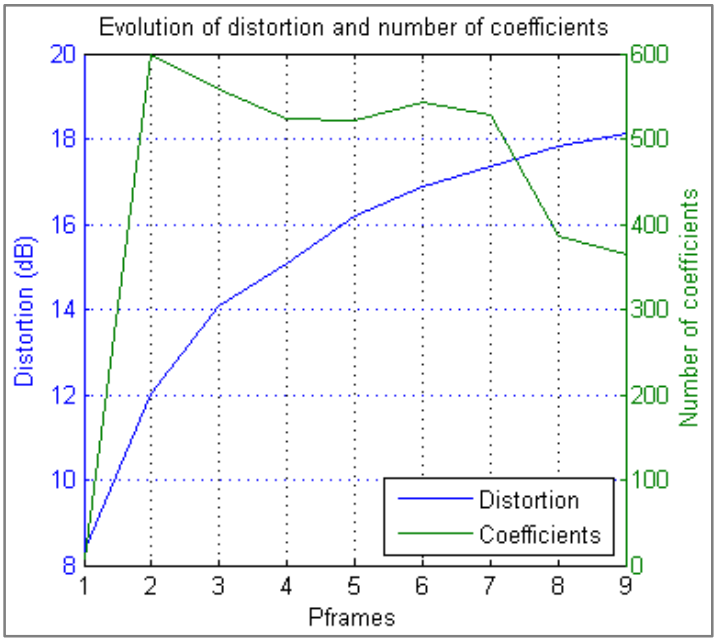


Figure 4.11 Relationship between the distortion and the nonzero coefficients ("Foreman.qcif" last packet)

4.2.1.2. RESULTS FOR "HUSKY.QCIF"

The previous analysis was repeated but, in this case, with a real packet size. Therefore, a new video sequence was selected which, taking into account the interest of this analysis, had high camera movement. The selected video sequence is called "husky.qcif" (250 frames) and was encoded with a packet size of 1200 Bytes, GOP size equal to 10 and QP for I and P frames equal to 36 which implies that more coefficients will be set to 0 after the quantization process.

Since in the previous section, the analysis was done in the case of vertical camera movement, it was decided to select a GOP with horizontal camera movement in order to check whether the observed results were valid also for this type of camera movement. Therefore, in the selected GOP, the camera is moving left and the first packet of the first P frame is removed. As a consequence of this movement, Figure 4.12 shows that the sMBs on the right of the lost packet disappear in the following frames and, consequently, are not used as reference for the following P frames.

However, it was expected that the new information appearing on the left side of each P frame became also corrupted since it was appearing within an area which was corrupted in the previous frame. In other words, it was expected that the number of corrupted sMBs was constant within the following P frames since, on the one hand, the corrupted sMBs were disappearing on the right side of the frames and, on the other hand, new sMBs that appear on the left side were supposed to become corrupted by appearing in a corrupted area.

In this case, taking into account the high camera movement of the video sequence, the JM encoder had encoded some of the MBs within the P frames as I MBs. Thus, this type of MBs do not become corrupted despite they appear on an area which was corrupted in the previous frame. In the example, the I MBs can be clearly seen in "Frame 2" and they correspond to the new MBs appearing on the left side of the frame in the area corresponding to the lost packet. Consequently, as some of the new information appearing on the left side of each P frame is encoded as I MBs and the other information appearing is

taking as reference these I MBs, the effect is that the corrupted sMBs are disappearing under the right side of the frame. As a result, the number of corrupted sMBs decreases within the GOP.

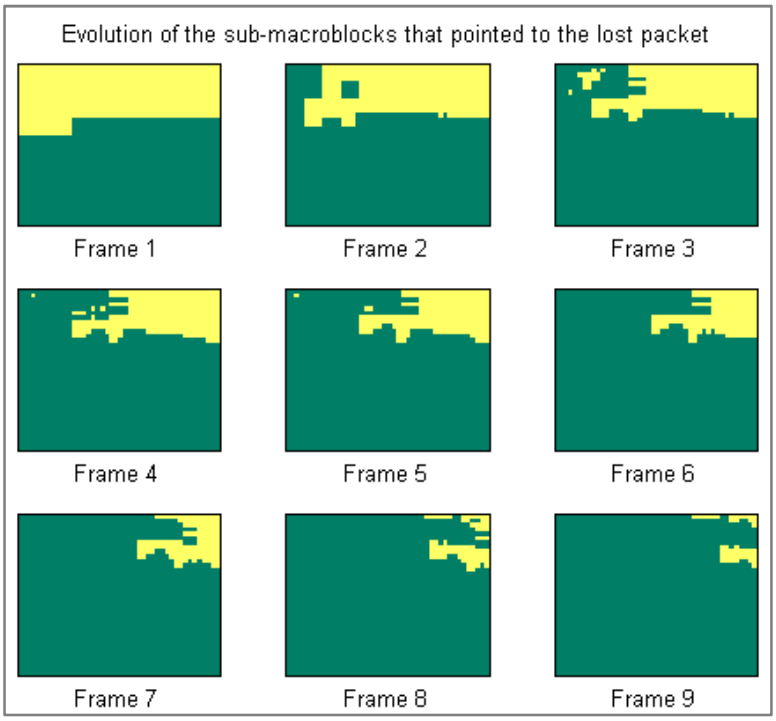


Figure 4.12 Evolution of the corrupted sMBs ("Husky.qcif" first packet)

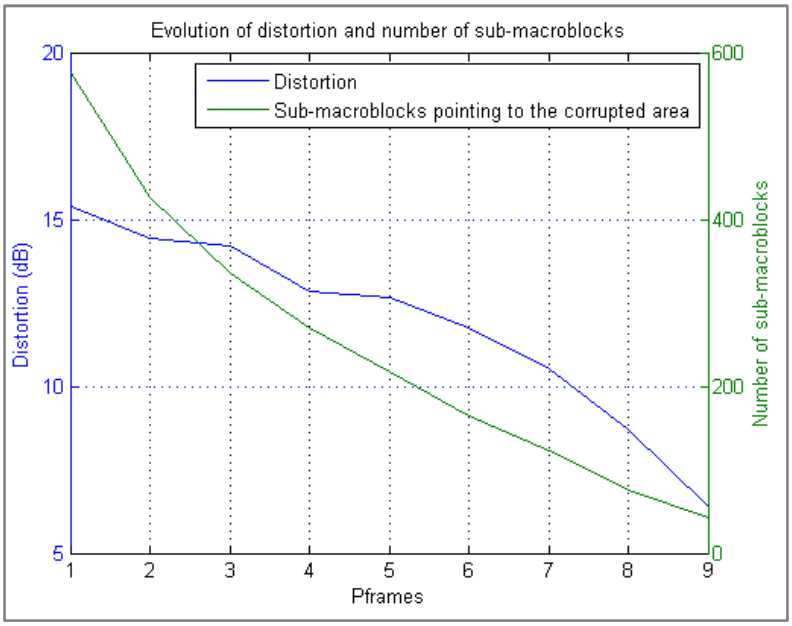


Figure 4.13 Relationship between the distortion and the corrupted sMBs ("Husky.qcif" first packet)

The distortion within the GOP decreases as depicted by the blue line in Figure 4.13 which is explained by the decreasing behavior of the number of corrupted sMBs represented by the green line. In the case of the nonzero coefficients contained within the corrupted sMBs, Figure 4.14 shows a behavior similar to the one observed in Figure 4.8: since the number of corrupted sMBs decreases, the number of their nonzero coefficients also decreases.

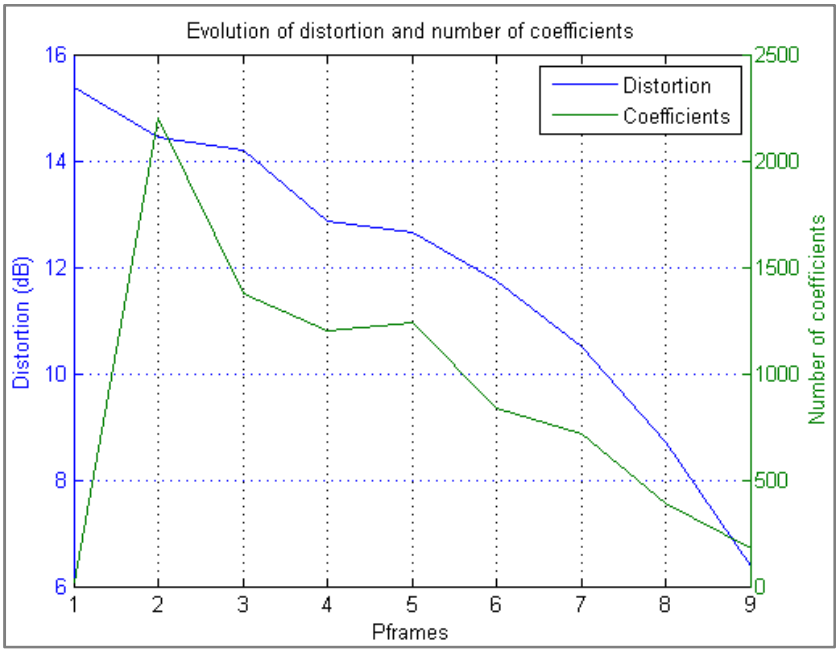


Figure 4.14 Relationship between the distortion and the nonzero coefficients (“Husky.qcif” first packet)

5. RANKING MODEL

Once it was studied the relationship between the evolution of the distortion and the evolution of the number of corrupted sMBs and the nonzero coefficients contained within them, it was proceed to estimate the distortion within the P frames of a GOP in order to create the ranking model.

The following subchapters explain the different steps carried out in order to create this ranking model as well as the tests done in order to check its correct working.

5.1. DISTORTION ESTIMATION (CALCULATION BASIS)

In order to estimate the evolution of the distortion in a GOP, the following set of variables of the GOP were used:

- Number of corrupted sMBs per each P frame following the error
- Number of nonzero coefficients contained within the corrupted sMBs
- Size of the lost packet
- Distortion in the frame where the packet is lost

The first two elements of the list correspond to the variables studied in the previous chapter. In addition, the size of the lost packet is used since it was observed that bigger packets, if they are lost, tend to cause higher distortion than smaller packets. Finally, the distortion in the frame where the packet is lost is needed in order to have a starting point to calculate how this distortion will evolve in the following P frames. It can be thought that a needed parameter should be the index of the P frame where the packet is lost, but in the following subchapter it is going to be showed that this parameter does not play so an important role.

Once the previous parameters have been calculated for all the possible lost packets (the ones belonging to P frames) within the GOP, the distortion caused by each lost packet is estimated by following the next procedure:

1. Calculate the increment or decrement of the number of corrupted sMBs (equation (2)) and nonzero coefficients (equation (3)) as a percentage:

$$\Delta sMB = \frac{sMB_{TOT} - sMB_{FINAL}}{sMB_{TOT}} \quad [\%] \tag{2}$$

Where:

- sMB_{FINAL}: Number of corrupted sMBs in the last P frame of the GOP
- sMB_{TOT}: Size of the lost packet in sMBs
- ΔsMB: Increment or decrement of the number of corrupted sMBs as a percentage of the total number of corrupted sMBs (size of the packet)

$$\Delta Coef = \frac{Coef_{INIT} - Coef_{FINAL}}{Coef_{INIT}} \quad [\%] \tag{3}$$

Where:

- $Coef_{INIT}$: Number of nonzero coefficients within the corrupted sMBs of the next P frame following the error
- $Coef_{FINAL}$: Number of nonzero coefficients within the corrupted sMBs of the last P frame of the GOP
- $\Delta Coef$: Increment or decrement of the number of nonzero coefficients within the corrupted sMBs as a percentage of the total number of nonzero coefficients (in this case $Coef_{INIT}$).

By doing this, it is possible to discriminate better which packets have a higher increment or decrement of the number of corrupted sMBs and nonzero coefficients.

2. Estimate the slope of the distortion within the GOP as a percentage:

In this case, Figure 5.1 shows the procedure to estimate the slope of the distortion (*m*) within the GOP in order to, afterwards, estimate the value of the distortion in each position (P frame) by means of interpolation.

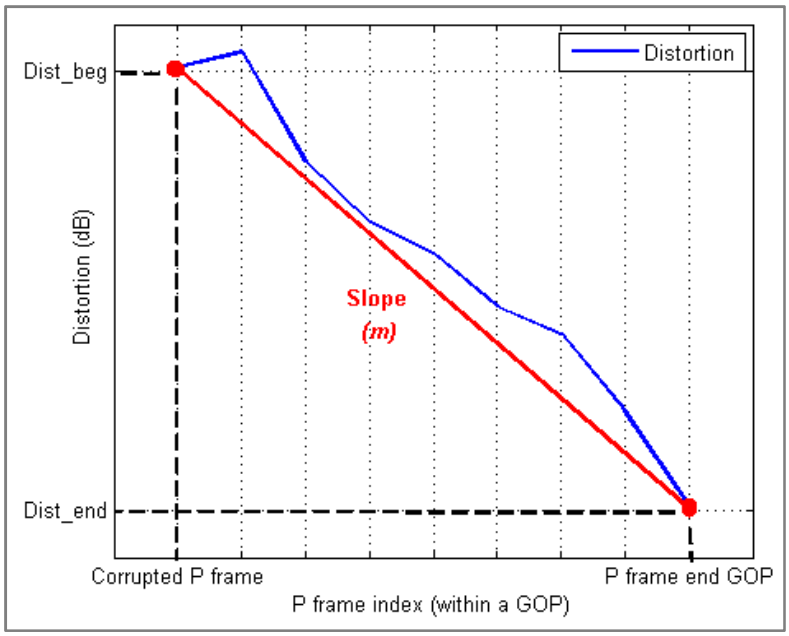


Figure 5.1 Example of the estimation of the increment/decrement of the distortion

In this case, *m* is defined as showed in equation (4). Therefore, it can be seen that *m* corresponds to the percentage of the total increment or decrement of the distortion within the GOP.

$$m = \frac{Dist_{BEG} - Dist_{END}}{Dist_{BEG}} \quad [\%] \tag{4}$$

Where:

- $Dist_{BEG}$: Distortion in the frame where the packet is lost

- $Dist_{END}$: Distortion in the last P frame of the GOP

In order to estimate *m*, 3 parameters are calculated:

$$Param1 = \Delta sMB / \max(\Delta sMB) \quad (5)$$

$$Param2 = \Delta Coef / \max(\Delta Coef) \quad (6)$$

$$Param3 = \frac{(1/sMB_{TOT})}{\max(1/sMB_{TOT})} \quad (7)$$

Param1 and *Param2* correspond to *ΔsMB* and*ΔCoef* respectively but moved to the range [-1, 1]. For instance, in the case of *Param1*, the packet with the highest decrement of corrupted sMBs will have a 1. *Param3* does the same function but in this case with the size of the lost packet.

Once these 3 parameters have been calculated, *m* is estimated as showed in equation (8). The estimated slope is defined as *n*.

$$n = \frac{Param1 + Param2 + Param3}{3} \quad [\%] \quad (8)$$

3. Estimate the total distortion caused by each packet in the GOP.

The first step consists in calculating the absolute distortion slope which is calculated as:

$$N = \frac{Dist_{BEG} \cdot n}{\text{length}(GOP) - i\_Pframe} \quad (9)$$

Where:

- *N*: Absolute distortion slope
- Length(GOP): Size of the GOP in P frames
- *I_Pframe*: Index of the P frame where the packet is lost

Then, the distortion in each of the following P frames is estimated by interpolation, therefore using the equation of a straight line. In this case, the calculation is based in equation (10). Finally, as showed in equation (11), the total distortion caused by a packet is calculated as the sum of the distortion in each P frame starting from the frame where the packet is lost to the final P frame of the GOP.

$$Dist_{ESTIM}(idx\_Pf) = N \cdot idx\_Pf + Dist_{BEG} \quad [\text{dB}] \quad (10)$$

$$Dist_{ESTIM\_TOT}(idx\_pack) = \sum Dist_{ESTIM} \quad [\text{dB}] \quad (11)$$

Where:

- $Dist_{ESTIM}$: Estimated distortion caused by a packet at P frame level
- *idx_Pf*: Index of the P frame within the GOP
- $Dist_{ESTIM\_TOT}$: Total estimated distortion caused by a packet at GOP level
- *idx_pack*: Lost packet index

5.2. RANKING MODEL CREATION

In order to create the ranking model, a script called “dist_in_GOP_2.m” was created. Taking into account the definition of the ranking model made in subchapter 2.3, the ranking model has to work by GOP basis and estimate the possible distortion that each packet can cause within the GOP.

Therefore, the script starts a loop in which it will estimate the distortion caused by the loss of the first packet of the first P frame within the GOP, of the second packet of the first P frame within the GOP until the last packet of the last P frame of the GOP.

In order to do so, for each packet, the script calls the “generate_dist_movement_map.m” script and calculates the evolution of the corrupted sMBs and the evolution of the nonzero coefficients contained within them. In addition, the “dist_in_GOP_2.m” script stores per each packet the index of the P frame where the packet is lost, the size of the lost packet in sMBs and the distortion measured in that frame.

Then, once it has calculated and stored these values for all the packets of the P frames within the GOP, the script follows the procedure explained in the previous section in order to estimate the total distortion caused by each packet. In the case of the packets of the last P frame of the GOP, the total distortion caused by the packet corresponds to the distortion measured in the frame.

Next, the packets are sorted from the packet causing the lowest distortion to the packet causing the highest distortion. Finally, depending on the number of logical channels available, the packets are divided in as many groups as available logical channels.

5.2.1. RESULTS

A GOP with large camera movement, concretely the frame range [170-179], of the encoded video sequence “husky.qcif” used in section 4.2.1.2 was selected for this analysis. Therefore, this analysis was done with real packets.

Figure 5.2 shows the comparison between the real total distortion (yellow bars) caused by each packet of the P frames within the GOP and the total estimated distortion (green bars) caused by each packet and calculated by the model. In this graph the x axis corresponds to the packet index and they are ordered in such a way that the first position corresponds to the first packet of the first P frame of the GOP and the last position corresponds to the last packet of the last P frame of the GOP.

It can be appreciated that, in almost all the cases, the estimated distortion is close to the real one. This graph shows an example of the reason why the index (within a GOP) of the P frame where the packet is lost is not so important compared to the parameters used in the distortion estimation. It can be supposed that if a packet is lost in the first P frame of a GOP, then the increase or decrease of the distortion will be higher than if it is lost in the fifth P frame since, in the first case, there are more P frames until the end of the GOP for the number of corrupted sMBs to increase or decrease.

However, in Figure 5.2, it is observed that there are packets in the last P frames of the GOP that, if they are lost, cause higher distortion than packets belonging to P frames in the middle of the GOP. In addition, in the last 5 positions (which correspond to the packets of the last P frame) it is seen that estimation and real distortion are equal since the total distortion in these cases corresponds to the distortion at picture level.

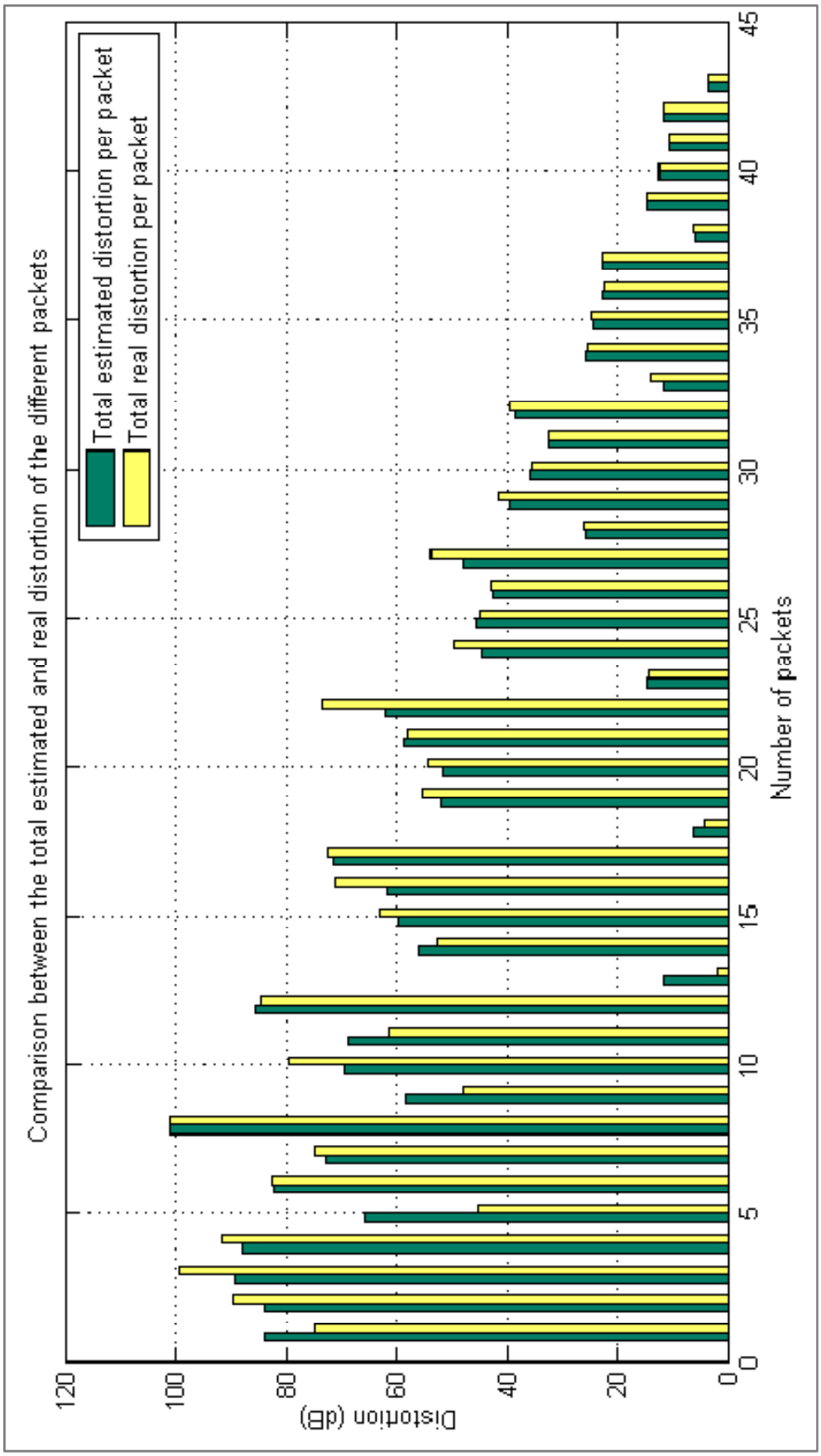


Figure 5.2 Estimated distortion versus real distortion (“Husky.qcif”)

After the packets were sorted from the one estimated to cause the lowest distortion to the one estimated to cause the highest distortion, the result ("estimated ranking") was compared to the "real ranking", which means measuring the real distortion and sorting the packets in the same way. The result of the comparison is shown in Figure 5.3.

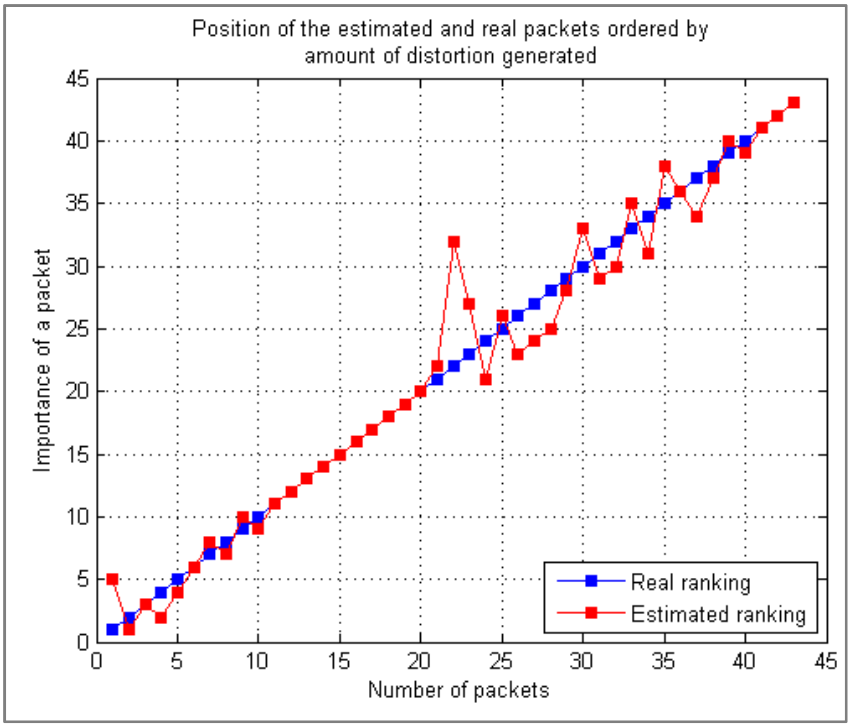


Figure 5.3 Comparison between the "real ranking" and the "estimated ranking" (GOP [170-179])

As it has been mentioned, the first position in the x axis corresponds to the packet causing the lowest distortion, and the last position corresponds to the packet causing the highest distortion. Although there are some differences, it can be appreciated that almost all the positions within the "estimated ranking" are very close to their corresponding positions in the "real ranking" and, when they do not, in almost all the cases the difference between the positions of the real and the estimated ranking is about 3 positions.

Next, the packets were divided in groups. In this case, it was supposed the existence of 2 available logical channels, one with a higher QoS and the other with a lower QoS. Therefore, the packets were split in 2 groups: those packets from the first position in the estimated ranking to the one being in the middle of the ranking were put in a group called "Group of packets causing low distortion" and the rest of packets were put in a group called "Group of packets causing high distortion".

When comparing how many packets were correctly classified in each group according to the "real ranking", Figure 5.4 shows the obtained results. In this case, the blue bars represent how many packets should be in each group according to the "real ranking". The red bars represent the amount of packets that were correctly classified in each group. It can be seen that 20 out of 21 where correctly classified within the "Group of packets causing high distortion" (abbreviated as "GHD") and 22 out of 21 where correctly classified in the "Group of packets causing low distortion" (GLD). This test was carried out for other GOPs of the video sequence and it was observed that approximately 90 % of the packets were correctly classified in each group (see Appendix E). Taking into account the obtained results

for the GOP analyzed in this section, the percentage of correctly classified packets increases to approximately 93 %.

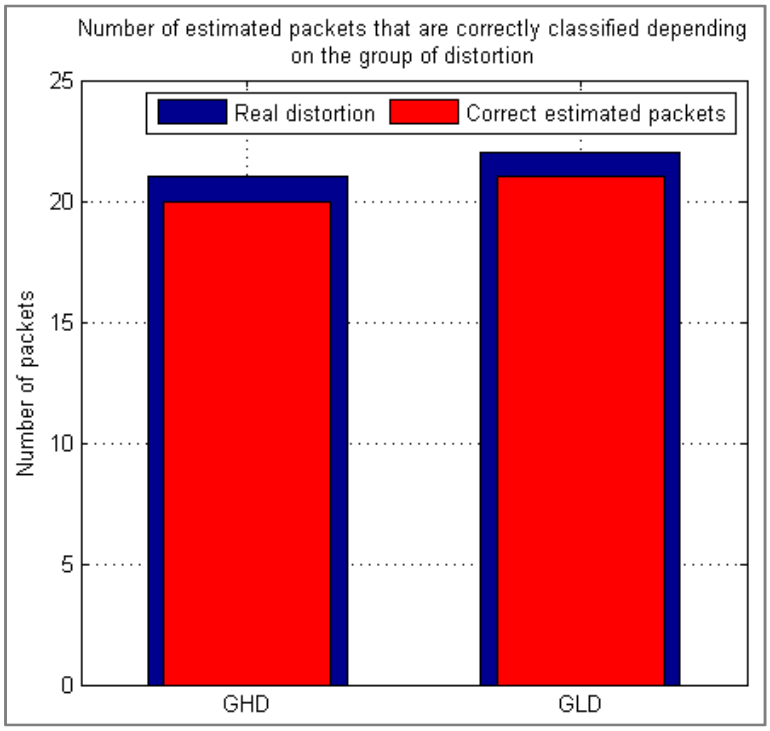


Figure 5.4 Amount of correctly estimated packets in each group (GOP [170-179])

5.3. SIMULATIONS

Once the correct operation, in terms of classification and ranking of packets, of the ranking model was checked according to Figure 5.4, simulations were carried out in order to check whether the quality of the decoded video sequence improves when the ranking model is working or not.

Therefore, the simulations were carried out for 2 different scenarios. The first scenario or "scenario nº 1", depicted in Figure 5.5, corresponds to simulate a standard transmission. In this case the H.264/AVC encoder encodes the packets and sends them through a channel with a 5 % of packet loss rate.

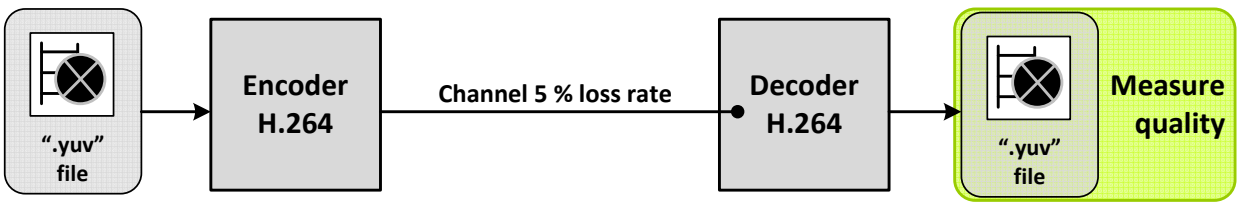


Figure 5.5 Standard transmission (scenario nº 1)

Then, in the second scenario or "scenario nº 2", which is depicted in Figure 5.6, it is simulated that the ranking model is enabled. In this scenario there are 2 available logical channels, one with a packet loss rate of 7.5 % (low QoS channel) and the other of 2.5 % (high QoS channel). Hence, the packets will have to be divided in 2 groups.

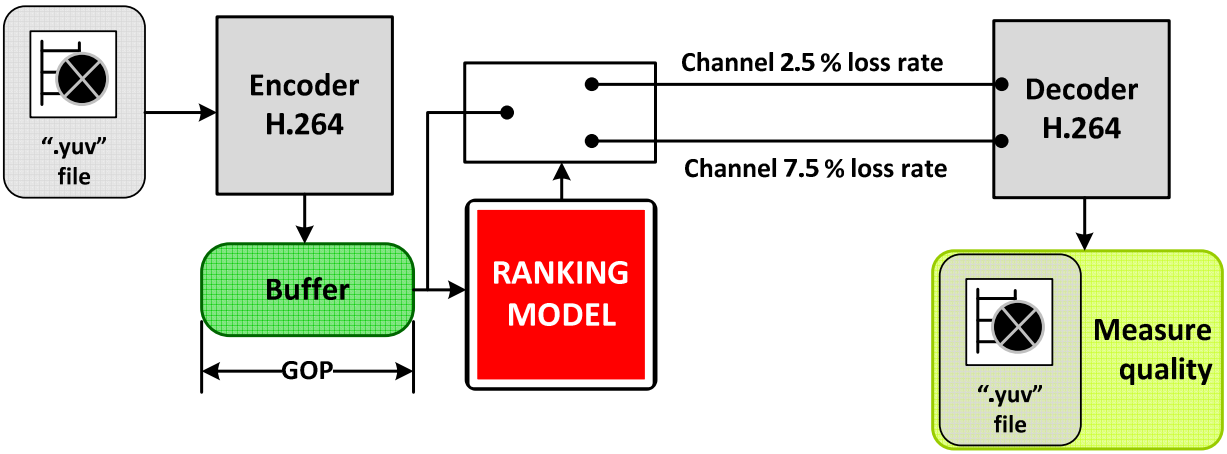


Figure 5.6 Ranking model working (scenario nº 2)

Since in this project, those packets estimated to cause higher distortion are considered to be more important; therefore, the ranking model decides that those packets being in the "Group of packets causing high distortion" are going to be sent through the channel with a 2.5 % of packet loss rate. Then, the packets being in the "Group of packets causing low distortion", they will be sent through the channel with a packet loss rate of 7.5 %. It has to be pointed out that, although the packets are being sent through 2 different logical channels, both channels have in average the same packet loss rate than the channel of the "scenario nº 1".

In the end, the quality of the decoded sequence in each case is measured and the results are compared between them.

5.3.1. Simulation procedure

In order to carry out the simulations, a total of 4 GOPs (maintaining the GOP size to 10) with high vertical and horizontal movement of the video sequence "husky.qcif" were selected. They corresponded to the frame ranges of [170-179], [190-199], [210-219] and [220-229]. For each of them, a video sequence was created. In this case the video sequence corresponded to the GOP repeated a total of 60 times. Therefore, the result was a video sequence of 600 frames where the GOP was repeated. Consequently, 4 video sequences of 600 frames were obtained.

Next, each GOP was analyzed by the "dist_in_GOP_2.m" script and the packets were ranked and split in 2 groups in the same way as mentioned in section 5.2.1. Then, the indexes of the packets and the P frames they belong to were stored for each group and saved in ".mat" files. Therefore, for each GOP (".mat" file), 4 vectors were stored:

- "Group of packets causing low distortion" and "Group of packets causing high distortion"
    1. Indexes of the packets
    2. Indexes of the P frames to where the packets belong

After that, the effect of sending the packets through a channel with a determined packet loss rate was carried out by following a procedure similar to the one explained in section 3.3.2. Depending on the scenario, a percentage (5 % or [2.5 %, 7.5 %]) of the total amount of packets was skipped from the video sequence.

Therefore, each video sequence was encoded with the same parameters used for "husky.qcif" in section 4.2.1.2. As a result, according to Figure 3.1, a ".264" file was obtained for each of them. Each ".264" file was processed by a script called "simul_skip_pack.m" which is similar to "skip_n_packet_P.m" (section 3.3.2.1).

In this case, the "simul_skip_pack.m" script reads the ".264" file and creates a structure containing all the packets of the video sequence. Then, it starts reading each packet and, in the case of the "scenario nº 1", it decides with a probability of 5 % to skip the packet or to copy it in a new empty structure. The result is that the new structure contains approximately 95 % of all the packets in the original structure. Therefore, approximately 5 % of the packets have been skipped (unless the packets from I frames since they are scheduled more rarely than P frames and, therefore, are supposed to not be affected by the channel). Finally, the script writes the new structure in a new ".264" file.

However, in the case of the "scenario nº 2", the script reads each packet and takes its packet index and P frame index. With this data, depending on the video sequence, it checks in the corresponding ".mat" file to which group the packet belongs. If it belongs to the "Group of packets causing high distortion", then the script decides with a probability of 2.5 % to skip the packet or not. If, however, the packet belongs to the "Group of packets causing low distortion", then with a probability of 7.5 %, the script decides to skip the packet or copy it in the new structure. Next, the structure is copied in a new ".264" file and the result is that approximately 2.5 % of all the packets belonging to the "Group of packets causing high distortion" are skipped and 7.5 % of all the packets belonging to the "Group of packets causing low distortion" are skipped. In average, approximately 5 % of all the packets (except the ones from I frames) are skipped.

Subsequently, in order to have simulations of around 10000 frames for each scenario and video sequence, this procedure is repeated 17 times per video sequence and scenario. Therefore, for each video sequence (600 frames), 34 ".264" files are obtained: 17 corresponding to the "scenario nº 1" and 17 corresponding to the "scenario nº 2". Consequently, a total number of 136 ".264" files are obtained.

The ".264" files are decoded and the result is stored in text files in the same way as it is done in section 3.3.2.2. Then, the quality values (YPSNR) stored in each text file are extracted by means of the "batch_measure_distortion.m" script. Finally, the average quality of each simulation (text file) is calculated and the results are compared in the following way:

- Video sequence GOP 170:
  - Average quality file #1 "scenario nº 1" against average quality file #1 "scenario nº 2"
  - …
  - Average quality file #17 "scenario nº 1" against average quality file #17 "scenario nº 2"
- …
- Video sequence GOP 220:
  - Average quality file #1 "scenario nº 1" against average quality file #1 "scenario nº 2"
  - …

- Average quality file #17 “scenario nº 1” against average quality file #17 “scenario nº 2”

5.3.1.1. RESULTS

The following set of figures (Figure 5.7, Figure 5.8 and Figure 5.9) corresponds to examples of the results obtained. Each of them shows the results obtained for a determined comparison between simulations and video sequence. In these cases, the average quality measured in the case of simulating the “scenario nº 2” is depicted as green and in blue it is depicted the average quality when simulating the “scenario nº 1”.

It can be appreciated that, in all of them, the average quality of the video sequence increases in the case of simulating the “scenario nº 2” (optimization enabled) than when simulating the “scenario nº 1” (standard transmission).

In fact, in more than 90 % of the simulation comparisons carried out, it was checked that the quality of the video sequences, in average, improved when simulating that the optimization was enabled (ranking model operative) than when simulating the standard transmission. In those cases, the improvement was of 1 dB in average. In addition, the opposite case (average quality when simulating “scenario nº 2” lower than average quality when simulating “scenario nº 1”) corresponded to approximately 1 % of the total number of comparisons between simulations. In the rest of the comparisons, the quality in both cases was approximately the same value.

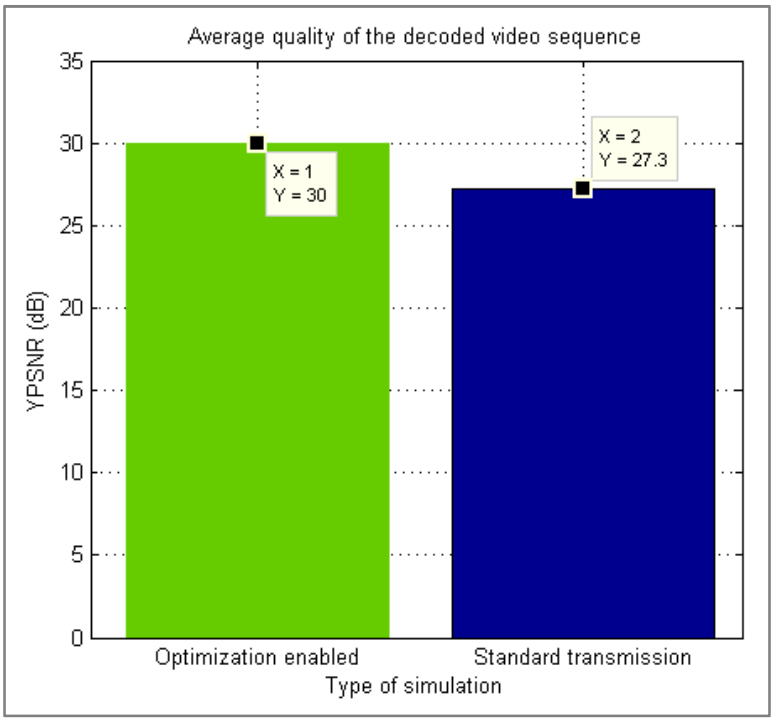


Figure 5.7 Simulation results for GOP [170-179]

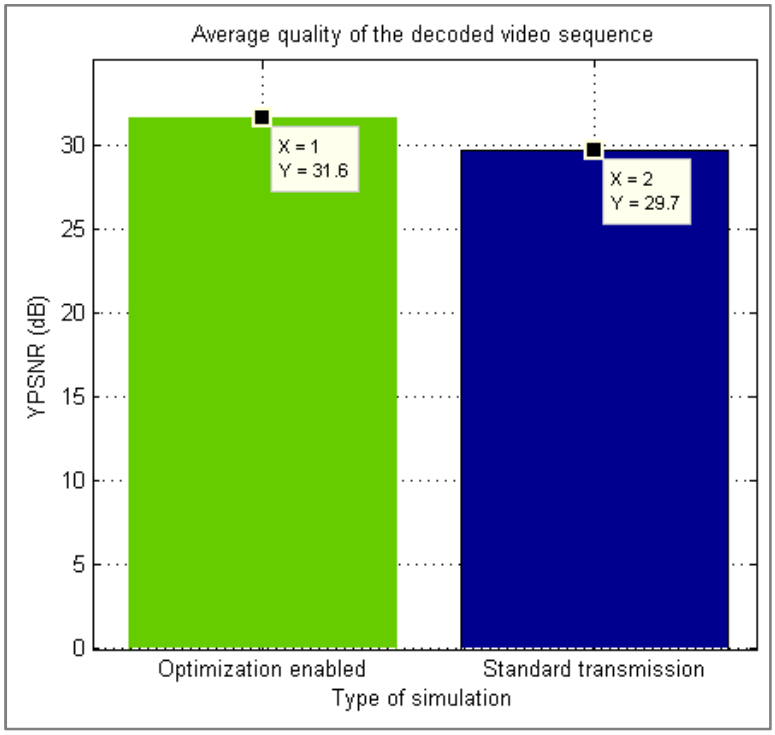


Figure 5.8 Simulation results for GOP [210-219] (I)

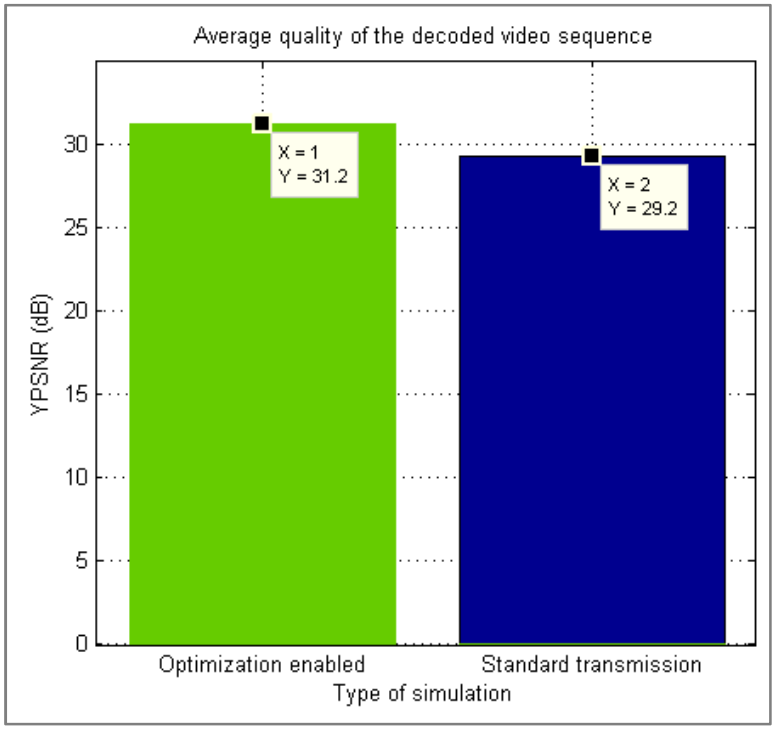


Figure 5.9 Simulation results for GOP [210-219] (II)

6. CONCLUSIONS

The scope of this project consists in the implementation and testing of a method in order to increase the quality of H.264/AVC Baseline profile encoded video sequences with QCIF (144x176 pixels) resolution transmitted over error-prone IP-based channels.

Errors, whether they are bit inversion errors or packet losses, are detected by the lower layer entities and result in packet losses. Packets can contain in this case VCL NAL units which consist of data associated to a video slice or non-VLC NAL units which include parameter sets and supplemental enhancement information. Since the amount of VCL NAL units is much higher than the number of non-VCL NAL units, therefore it is much probable that a VCL NAL unit is lost as the result of a transmission error. This type of loss produces visual artifacts on the video sequence which can be spatially (I frames) and/or temporal (I and P frames) propagated depending on the frame to which the lost packet belongs. Since in the streaming scenario I frames are scheduled more rarely than P frames, therefore it is more probable that, as a consequence of a transmission error, a packet of a P frame is lost. As a buffer of size 1 is considered, which means that the current frame is using the previous frame as reference, the error will propagate from the P frame where the error occurs until the last P frame of the GOP.

In order to increase the quality of the transmitted video sequence, the strategy followed is based on the protection of those packets which are estimated to cause higher distortion if they are lost. In this case, the distortion that a packet can cause is estimated by means of the features of the packet. Therefore, the method proposed works by GOP basis since the effect of an error in a P frame propagates until the last P frame of a GOP. The method then analyzes the features of the packets of the P frames and estimates the total distortion that each of them can cause within the GOP if they are lost. Finally, depending on this total estimated distortion, the packets causing the highest distortion are better protected by being transmitted over high QoS logical channels.

The estimation of the total distortion caused by a lost packet is split in 2 calculations. When the error occurs, the quality of the affected P frame is degraded and, due to the temporal prediction exploitation by the encoder, the degradation propagates to each of the following P frames until the last one of the GOP (temporal error propagation). Consequently, in order to estimate the total distortion, it is needed to estimate the distortion in the frame where the error occurs (distortion at picture level) and the evolution of the distortion in the following set of P frames because of the temporal error propagation.

In the first case, some packets of a video sequence were removed. Then, by modifying the H.264/AVC JM 15.0 (FRExt) decoder, the features of the removed packets were outputted and then they were matched with the distortion measured in the decoded video sequence when removing those packets. The results show that there is a relationship between the features of the removed packets and the distortion caused by them. Even though all of them show a relationship, there are some features like the number of nonzero coefficients or the number of sMBs containing nonzero coefficients which show a stronger relationship with the distortion than the rest of features. In these cases, the results show that the higher the number of nonzero coefficients or sMBs with nonzero coefficients that a packet has; the higher the distortion that this packet can cause in the decoded video sequence if it is lost.

Therefore, this preliminary analysis proved the existence of a relationship between the features of the lost packets and the distortion caused by them. The next step was to estimate

the distortion by means of these features. The strategy followed to estimate this distortion was by means of interpolation. Therefore, a tool called PCA (Principal Component Analysis) was used in order to reduce the set of features to a number suitable in order to carry out the interpolation. PCA allows re-expressing the set of variables as a lower number of components which are a linear combination of the original set and are uncorrelated between them. Therefore, by means of PCA, the coordinates of the original set of variables could be transformed into component coordinates and then, the distortion at picture level would be estimated by means of interpolation.

In the case of the evolution of the distortion, from practical observations it was observed that the evolution of the distortion within a GOP was affected by the camera movement. As a consequence, the number of corrupted sMBs within a GOP increased or decreased. The distortion in these cases was observed to increase and decrease, respectively. Taking into account that the number of nonzero coefficients of the lost packets affect the distortion, it was also calculated its evolution within a GOP. The obtained results show a relationship although the main feature that describes the evolution of the distortion is the number of corrupted sMBs.

The implemented method takes the previous data and other additional parameters for each of the packets of the P frames of a GOP and the distortion at picture level and estimates the distortion in each of the following P frames of the GOP. Then, it sums all of them together in order to estimate the total distortion caused by each packet.

In this case, the aim is not that the model perfectly estimates the distortion caused by the loss of each packet in the decoded video sequence, but that the packets are correctly ranked from the packet causing the lowest distortion to the packet causing the highest one. Then, supposing the existence of 2 logical channels, the packets are divided in 2 groups. Results show that more than 90 % of the packets were correctly classified into each group.

The results of the simulations carried out in order to check the improvement introduced by the method proposed, show that in more than 90 % of the simulations, the average quality (YPSNR) of the decoded video sequence increases 1 dB in average when simulating that the method is working than when simulating a standard transmission in even conditions.

Since, due to time schedules, the PCA stage was not finished, future work with this method would imply the correctly estimation of the distortion at picture level by means of the PCA. In addition, taking into account that this project defines a first approximation of the method, it would be interesting to improve the scheme used to estimate the total distortion in order to improve the ranking results.

Taking into account that this method, in order to improve the quality of the transmitted video sequence, takes advantage of the information of the transmitted packets without adding any extra information; this implies a method of smaller computational complexity than other methods in the literature of the H.264/AVC standard. Therefore, future work with this method would also imply the study of how this method can help to improve the results of already implemented error resilience methods.

7. BIBLIOGRAPHY

1. **Heinrich-Hertz-Institut, Institut Nachrichtentechnik.** H.264/AVC Software Coordination. [Online] [Cited: March 03, 2010.] http://iphome.hhi.de/suehring/tml/.

2. **Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer and Thomas Wedi.** Video coding with H.264/AVC:Tools, Performance, and Complexity. [Online] 2004. [Cited: October 27, 2009.] http://www.img.lx.it.pt/~fp/artigos/H264_final.pdf.

3. **Gary J.Sullivan, Pankaj Topiwala and Ajay Luthra.** The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. [Online] August 2004. [Cited: October 29, 2009.] http://www.fastvdo.com/spie04/spie04-h264OverviewPaper.pdf.

4. *Overview of the H.264/AVC Video Coding Standard.* **Thomas Wiegand, Gary J.Sullivan, Gisle Bjøntegaard and Ajay Luthra.** 7, July de 2003, IEEE Transactions on circuits and systems for video technology, Vol. 13.

5. **Ikuno, Josep Colom.** Performance of an Error Detection Mechanism for Damaged H.264/AVC Sequences. [Online] December 2007. [Cited: September 5, 2009.] http://www.nt.tuwien.ac.at/fileadmin/topics/mobile_communications/finished_theses/Master_J.COLOM_IKUNO.pdf .

6. *H.264/AVC Over IP.* **Wenger, Stephan.** 7, July de 2003, IEEE Transactions on circuits and systems for video technology, Vol. 13.

7. *H.264/AVC in Wireless Environments.* **Thomas Stockhammer, Miska M. Hannuksela and Thomas Wiegand.** 7, July de 2003, IEEE Transactions on circuits and systems for video technology, Vol. 13.

8. **ITU-T.** Recommendation H.264: Advanced video coding for generic audiovisual services. [Online] 03 2009. [Cited: February 15, 2010.] http://www.itu.int/rec/T-REC-H.264-200903-I/en.

9. **Shlens, Jonathon.** A tutorial on Principal Component Analysis. [Online] March 25, 2003. [Cited: January 31, 2010.] http://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf.

10. **Jolliffe, I. T.** Introduction. *Principal Component Analysis.* Second Edition. s.l. : Springer-Verlag New York, Inc., 2002, pp. 1,2,6,7.

11. **Scholz, Matthias.** PCA - Principal Component Analysis. [Online] [Cited: January 31, 2010.] http://www.nlpca.de/pca.html.

APPENDIXES

APPENDIX A

This appendix shows and explains the code introduced in the JM reference H.264/AVC decoder in order to output the needed information for the processes described in subchapter 3.2. Next, the code is presented and a brief explanation of the functions that the code is carrying out is given.

A.1. MODIFIED CODE OF THE JM REFERENCE H.264/AVC DECODER

For the reasons explained in section 3.2.1, instead of modifying the JM encoder, it is modified the JM decoder. Since the information is wanted to be outputted as text files; therefore, first it was needed to define the text files within the code. Hence, in the "global.h" file of the "ldecod" project, which is the project corresponding to the decoder, it was added the following code where the pointers to the files are defined.

```
FILE *p_mb_INFO;
FILE *p_mb_mvd_info_H;
FILE *p_mb_mvd_info_V;
FILE *p_abs_MV_H;
FILE *p_abs_MV_V;
FILE *p_mb_coef_levarr_info;
FILE *p_mb_coef_runarr_info;
```

Then, in the source file "ldecod.c" it was added, for each of the previous pointers, the following code:

```
if ((p_mb_mvd_info_H=fopen("mb_mvd_info_H.txt","w"))==0)
// append new statistic at the end
{
  snprintf(errortext, ET_SIZE, "Error open file %s!",TRACEFILE);
  error(errortext,500);
}
```

Once this was done, it was proceed to insert the code in order to output the parameters of the video sequence. Therefore, as the information wanted to be outputted corresponds to information at MB level, the source file "macroblock.c" of the project "ldecod" was modified. In order to output the information, it was needed to detect where in the code each of the variables was being read. Once this was detected, the specific code for each case was added.

Next, it is presented the code written where the MV are being read by the decoder. In this case, the code is reading the variables that store the relative MV which correspond to the error between the predicted MV and the original ones. There are two variables, one for the horizontal direction and the other for the vertical one. For each variable, the code reads per each MB the 16 values corresponding to the motion of each 4x4 block of samples of the MB and then writes it in a text file.

At the end, the code outputs 2 files storing, per each MB of the video sequence, a vector of 16 values corresponding to the relative motion of the blocks of 4x4 samples in which the MB is divided.

```
//This part of the code outputs the relative motion vectors (H and
V) for each macroblock
```

```
  if (img->number > 0){
      fprintf(p_mb_mvd_info_H,"PIC: %d\t SLICE: %d\t MB: %d [H]:
\t",img->number,img->current_slice_nr,img->current_mb_nr);
      fprintf(p_mb_mvd_info_V,"PIC: %d\t SLICE: %d\t MB: %d [V]:
\t",img->number,img->current_slice_nr,img->current_mb_nr);
      for (i1 = 0; i1 < 1; i1++){
            for (i2 = 0; i2 < 4; i2++)
                  for (i3 = 0; i3 < 4; i3++)
                        for (i4 = 0; i4 < 2; i4++) {
                              if (i4 == 0)

    fprintf(p_mb_mvd_info_H,"%d\t",currMB->mvd[i1][i2][i3][i4]);
                              if (i4 = 1){

fprintf(p_mb_mvd_info_V,"%d\t",currMB->mvd[i1][i2][i3][i4]);}}

            fprintf(p_mb_mvd_info_H,"\n");
            fprintf(p_mb_mvd_info_V,"\n");
            fflush(p_mb_mvd_info_H);
            fflush(p_mb_mvd_info_V);
      }

  }
```

Subsequently, the absolute MV which correspond to the real ones are read. The previous process is repeated in the case of these 2 variables and, at the end, 2 new files storing the motion of the MBs in vertical and horizontal directions are obtained.

```
// This part of the code outputs the absolute motion vectors (H and
V) for each macroblock
  if (img->number > 0){

      currMB_row = floor(img->current_mb_nr / (img->width/16));
      currMB_col = img->current_mb_nr - (img->width/16) *
currMB_row;

      fprintf(p_abs_MV_H,"PIC: %d\t SLICE: %d\t MB: %d [H]: \t",img-
>number,img->current_slice_nr,img->current_mb_nr);
      fprintf(p_abs_MV_V,"PIC: %d\t SLICE: %d\t MB: %d [V]: \t",img-
>number,img->current_slice_nr,img->current_mb_nr);
      for (i1 = 0; i1 < 1; i1++){
            for (i2 = currMB_row * 4; i2 < (currMB_row+1)*4; i2++)
                  for (i3 = currMB_col * 4; i3 < (currMB_col+1)*4;
i3++)
                        for (i4 = 0; i4 < 2; i4++) {
                              if (i4 == 0)

    fprintf(p_abs_MV_H,"%d\t",dec_picture-
>motion.mv[i1][i2][i3][i4]);
                              if (i4 = 1){

fprintf(p_abs_MV_V,"%d\t",dec_picture->motion.mv[i1][i2][i3][i4]);}

                        }

            fprintf(p_abs_MV_H,"\n");
            fprintf(p_abs_MV_V,"\n");
            fflush(p_abs_MV_H);
            fflush(p_abs_MV_V);
      }}
```

Then, where the variables corresponding to the MB type and whether the MB is being skipped or not are being read, the following code was added. In this case, the code reads the variables and writes them in a text file. Therefore, the file indicates per each MB the previous information.

```
// This part of the code outputs de information associated to the
macrobloc such as if it is skipped, the type of macrobloc and the
cbp*/
  if (img->number > 0){
      fprintf(p_mb_INFO,"PIC: %d\t SLICE: %d\t MB: %d\t SN: %d\t T:
%d\t cbp: %d\t\n",img->number,img->current_slice_nr,img-
>current_mb_nr,currMB->skip_flag,currMB->mb_type,currMB->cbp);
  fflush(p_mb_INFO);
  }
```

Furthermore, another variable was read and outputted. This variable corresponded to the "cbp" which indicates in how many sMBs the MB is being divided. In this case, the sMBs correspond to groups of blocks of 4x4 samples. This variable was outputted but in the end not used in the analysis carried out in chapter 3 since, compared to the other variables, it was seen that the information it gives was not relevant enough.

Finally, where the variables corresponding to the coefficients of the each 4x4 block of samples are being read, this code was added. For more information on these variables, see Appendix B.

```
// This part of the code outputs (in the file
mb_coef_levarr_info.txt and file mb_coef_levarr_info.txt) the levarr
and rumarr coef for each subMB

    if ((img->number > 0)&&(0 <= k < numcoeff)) {

          fprintf(p_mb_coef_levarr_info,"PIC: %d\t SLICE: %d\t MB:
%d\t (%d,%d)\t",img->number,img->current_slice_nr,img-
>current_mb_nr,i/4,j/4);
          fprintf(p_mb_coef_runarr_info,"PIC: %d\t SLICE: %d\t MB:
%d\t (%d,%d)\t TOT_ZEROES: %d\t",img->number,img-
>current_slice_nr,img->current_mb_nr,i/4,j/4, total_zeroes);
          for (k = 0; k < numcoeff; k++)
          {

                fprintf(p_mb_coef_levarr_info,"%d\t",levarr[k]);
                fprintf(p_mb_coef_runarr_info,"%d\t",runarr[k]);
                           fflush(p_mb_coef_levarr_info);
                           fflush(p_mb_coef_runarr_info);

          }
          fprintf(p_mb_coef_levarr_info,"\n");
          fprintf(p_mb_coef_runarr_info,"\n");
}
```

The process carried out by this code is similar than the one carried out for the MVs. In this case, 2 files are outputted. In one of them, per each sMB, the code reads the nonzero coefficients and writes them in the file. In the case of the other file, the code reads the number of "TotalZeros" and the position of the zero coefficients and writes them in the file.

Therefore, at the end, one of the files stores per each sMB a vector containing the nonzero coefficients, and the other file stores per each sMB a vector contaning the zero coefficients and their position within the original 16 size vector of coefficients (Appendix B).

APPENDIX B

B.1. CAVLC CODING ENTROPY METHOD

Once the transformed coefficients are quantized, they are scanned in a zig-zag fashion. A typical zig-zag scan of quantized transform coefficients for a 4x4 luma block could be the following one:

8, 5, 3, 0, 1, 1, 0, 0, -1, 1, 0, 0, 0, 0, 0, 0

It can be appreciated that there are 16 coefficients and that only few of them are nonzero coefficients. In addition, there is a predominant occurrence of coefficient levels with magnitude equal to 1, so-called “Trailing 1’s” (T1’s), at the end of the scan (2).

Furthermore, the statistical distribution of the coefficients typically shows large values for the low frequency part decreasing to small values later in the scan for the high-frequency part. Based on this statistical behavior, the following data elements are used to transmit information of quantized transform coefficients for a 4x4 luma block (4).

- **Number of nonzero coefficients and T1’s**
- **Encoding the value of the coefficients:** The values of the coefficients are coded in reverse scan order, which means that in the example the first element to be coded would be 3. In the case of the T1’s, they only need sign specification since they all are equal to +1 or -1.
- **Sign information:** One bit is used to signal coefficient sign and, for T1’s, this is sent as single bits.

Finally, in order to code the positions of each non-zero coefficient the following data is transmitted:

- **TotalZeros:** Specifies the number of zeros between the last nonzero coefficient of the scan and its start.
- **RunBefore:** It specifies how the zeros between the nonzero coefficients are distributed.

APPENDIX C

C.1. CREATION OF THE STRUCTURE PIC

As it was said in the section 3.2.2, after the JM decoder was modified, as a result of the decoding of the encoded video sequence, a set of 7 text files storing the statistics of the packets of the encoded video sequence were obtained. Concretely, those files were named as follows:

- **mb_coef_levarr_info.txt:** Stored the nonzero coefficients per each sMB
- **mb_coef_runarr_info.txt:** Stored the "TotalZeros" and "RunBefore" per each sMB (see Appendix B)
- **abs_MV_V.txt:** Stored the absolute vertical MV per sMB
- **abs_MV_H.txt:** Stored the absolute horizontal MV per sMB
- **mb_mvd_info_V.txt:** Stored the relative vertical MV per sMB
- **mb_mvd_info_V.txt:** Stored the relative horizontal MV per sMB
- **mb_info.txt:** Stored the type of MB and P_Skip mode per each MB

Once these files were obtained, then it was proceed to create a series of MATLAB scripts (.m files) that extract the information within them and create the structure PIC which would store the information of the whole encoded video sequence.

C.1.1. WORKING OF THE INVOLVED MATLAB SCRIPTS

In general, the working of the scripts is shown in Figure C.1. It can be observed that the main script is that called "build_struct_info.m". This script receives as inputs all the previous files and creates the structure PIC which is, in the beginning, empty.

Then, it starts by calling the script "get_MB_other_info.m" and giving PIC and the "mb_info.txt" file as inputs. In fact, the main script gives PIC as an input for all the other scripts so the operation of filling the structure can be easier in operational and, consequently, time consuming terms.

So, the "get_MB_other_info.m" script starts reading the file "mb_info.txt" and, since this file contains statistics per MB, extracts the information about the type of MB, its index and whether it has been skipped or not. Then, it fills the corresponding fields of PIC and returns it to the main script.

As a second step, the main script calls the script "get_mvd.m" and gives as an input all the files that contain the information about the MV. It reads the files that stored the absolute horizontal and vertical MV per sMB and fills the correspondent fields of PIC (Figure 3.3) with two vectors that store these values for vertical and horizontal directions. Then, it calculates the average value of each vector and stores them in the field "Average value of

absolute MV" where the first position corresponds to the vertical component and the second one, to the horizontal component.

This last operation is repeated for the relative MV (previously reading the files "mb_mvd_info_V.txt" and "mb_mvd_info_H.txt") with the result of filling the field "Average value of the relative MV" in the same way as with the absolute MV.

Next, the main script calls the "average_num_subMB_coeff.m" script and gives as an input the "mb_coef_levarr_info.txt" file. Therefore, the script extracts the nonzero coefficients per each sMB and stores them in the corresponding position of the field "Coefficients" (remembering that it is an array and each row corresponds to a sMB of the MB). Then, it calculates the number of sMBs with nonzero coefficients within the MB and, subsequently, the total number of nonzero coefficients of those sMBs and fills the PIC fields "sMBs" and "Nonzero coefficients" respectively.

Finally, the main script calls the "get_coefficients.m" script which reads the "mb_coer_runarr_info.txt" file and calculates the complete 16 coefficients vector per each sMB since this text file contains, per each sMB, information that allows calculating the position of each nonzero coefficient (Appendix B). Then, depending on the type of the MB, the script calculates the field "Mode" as follows:

- **Mode 1:** The MB has not been subdivided (type 1)
- **Mode 2:** The MB has been subdivided in 2 (type 2 or 3)
- **Mode 4:** The MB has been subdivided in 4 (other types)

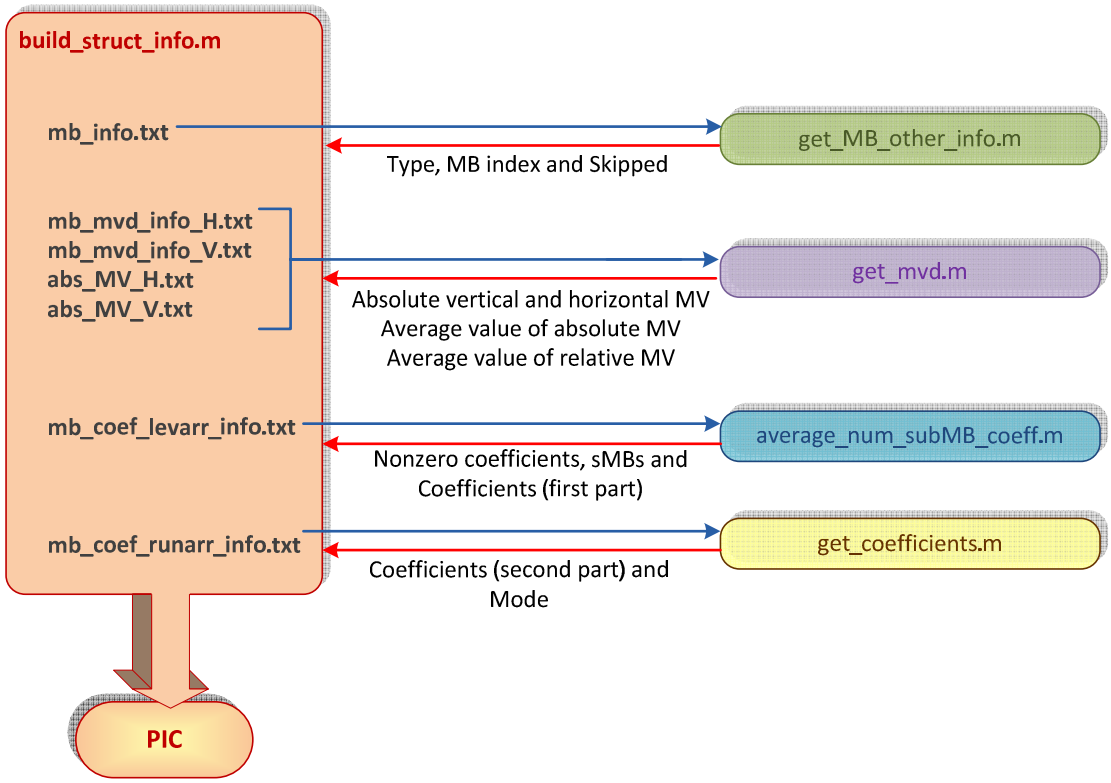


Figure C.1 Creation of PIC

C.2. AUTOMATIC DECODING OF THE "DAMAGED .264 FILES"

Once all the "damaged .264 files" have been obtained, it is needed to decode each of them. In order to do that, a different decoder configuration file (".cfg") and a different text file to store the results of the decoding process are needed (Figure C.2).

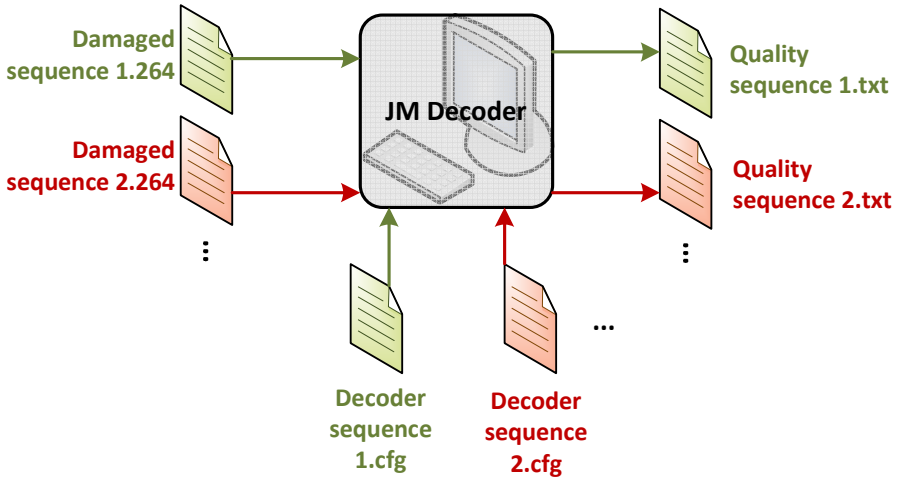


Figure C.2 Working of the decoding of the damaged ".264" files

Therefore, in order to generate the ".cfg" files, two MATLAB scripts were created: "generate_decoder_cfg.m" and "batch_generate_decoder_file.m". The last one corresponds to the main script and takes as input the following set of parameters:

- **cfg_filename_list:** Corresponds to a vector that, for each position, stores the name of the decoder file (".cfg") per each sequence. Therefore, the length of this vector is equal to the amount of sequences of removed packets. This parameter is created by means of the "generate_cfg_filename_list.m" script.

- **input_seq_list:** This corresponds to a vector that stores the names of the "damaged .264 files". In this case, this parameter is created by means of the "generate_input_seq_list.m" script.

- **output_vid_list:** Corresponds to the name that is wanted for the output of the decoding process or, in other words, the reconstructed video sequence.

- **ref_vid_list:** Corresponds to the name of the reference video that each ".cfg" file will include in order to calculate the quality of each picture. Therefore, this corresponds to the video sequence encoded in the beginning (section 3.3.2.2).

Then, with these parameters, it calls the "generate_decoder_cfg.m" script which is a copy of the syntax of the correct decoder ".cfg" file, and fills it with the information of the previous parameters. As a result, a set of ".cfg" files is obtained. Each of these ".cfg" files corresponds to a "damaged .264 file".

Next, to start the decoding process it is needed to type the following text in the "Command Prompt":

```
ldecod decoder.cfg > decoding_process.txt
```

This means that the JM decoder is configured as said in the "decoder.cfg" file and stores the results of the decoding process (the ones printed in the screen as shown in Figure 3.5) in the "decoding_process.txt" file.

Therefore, in order to automatically decode all the "damaged .264 files" and store the result in different text files, another script called "get_sim_bat.m" was created. This script creates an executable file ("sim_file.bat") that stores the previous "calling" for each "damaged .264 file". As a result, when clicking the obtained "sim_file.bat", the decoding of each of the "damaged .264 files" starts and the result is stored in a different text file.

C.3. MATCHING PACKET FEATURES WITH DISTORTION

As it was said in section 3.4.1, in order to match the features of the removed packets with the distortion generated by each of them, two scripts are used.

The first one, called "observing_features.m", takes the structure PIC where all the information of the encoded video sequence is organized and the index of the removed packet. Then, in each first P frame of each GOP (where the packet has been removed), it reads the information about the removed packet.

Since the information in PIC is organized in MBs, the script averages the information of all the MBs within the removed packet and as a result 6 vectors are obtained:

- **Av_coef:** Average number of nonzero coefficients of the packet.
- **Av_mvd_H:** Average horizontal motion of the packet.
- **Av_mvd_V:** Average vertical motion of the packet.
- **Skipped:** Since the script averages the value of the field "Skipped" (Figure 3.3) of the MB within the packet, then this vector stores the resulting average "Skipped" value of the packet.
- **Mode:** The same happens with mode, therefore this vector stores the average "Mode" (Figure 3.3) value of the packet.
- **subMB:** Average number of sMBs with nonzero coefficients of the packet.

Therefore, each position of these vectors stores the average features of the removed packet of the first P frame of each GOP. For instance, the first position will store the features of the removed packet of the first P frame of the first GOP; the second value will correspond to the features of the removed packet of the first P frame in the second GOP and so on. Consequently, the length of these vectors corresponds to the amount of GOPs in the encoded video sequence.

Then, in order to match the information within these last vectors with the distortion provoked by the removal of that determined packet, the script "analyse_max_distortion.m" is called.

For each index of the removed packet, this script calls the later script and obtains the former 6 vectors and then fills the information within them in another 6 vectors. These new 6 vectors will store the statistics of all the removed packets in the encoded video sequence (from the first removed packet to the last removed packet of each first P frame). Therefore their length will be the amount of GOPs multiplied by the amount of slices per frame.

Furthermore, it also creates a vector that will store the distortion generated in each GOP by the removal of each packet. In order to do so, it reads the structure Quality in the position corresponding to the removed packet and fills the vector with the distortion measured in each P frame of the GOP (when analyzing the error at picture level (GOP=2)) or the average distortion in the GOP (when analyzing the temporal error propagation (GOP>2)). Its length is the same as the one for the new 6 vectors.

So, at the end, 6 vectors are obtained that stored the previous features but for all the possible removed packets. Also, another vector stores the distortion generated by each of those packets. Therefore, these vectors are linked. The value stored in each of these vectors in a determined position will correspond to the statistics of a packet removed from a first P frame in a GOP and the distortion generated by the removal of that packet in that GOP.

Next, the script starts reading each of the created six vectors. In order to provide an easier understanding, the explanation will be done in the case of coefficients.

The script takes the indexes of the “coefficients” vector with a determined value of nonzero coefficients, for instance the indexes where the average number of nonzero coefficients is 0. Then, it goes to the distortion vector and averages the value stored in these indexes since these values correspond to the distortion generated by the removal of packets with 0 nonzero coefficients. It stores the calculated average distortion in the new vector and proceeds to repeat the process for all the possible values stored in the “coefficients” vector.

As a result, it is obtained a vector that stores the average distortion generated by the removal of packets with a determined average number of nonzero coefficients. Also, another vector is created but this one stores the number of packets with a determined average number of nonzero coefficients. Hence, this vector performs the function of a histogram.

These operations are repeated for each of the six vectors and, at the end, the script plots two graphics per each feature. The first one shows the previous “histogram” and the second one shows the average distortion caused by a determined value of the feature.

C.4. CREATION OF THE MOVEMENT AND COEFFICIENTS MAPS

In order to create the movement and coefficients maps needed for the analysis on chapter 4, the script “create_mv_map.m” was created. This script takes the structures “movement” and PIC as well as the index of the frame being analyzed.

With these inputs, it creates 3 arrays of size 36x44 which corresponds to the frame size in sMBs instead of pixels. The first 2 will store the motion of the sMBs in vertical and horizontal direction respectively; and the third one will store the amount of nonzero coefficients of each sMB.

Then, the script reads “movement” in the position corresponding to the frame index. By means of the structure “MB”, it calculates the position of each sMB within the new maps and also its movement in sMB units; thus dividing by 4 the value stored for each sMB within the “MB” structure which corresponded to the movement of the sMBs but in pixels, according to subchapter 4.2.

After that, it stores each calculated value (one corresponding to the horizontal direction and the other to the vertical direction) in the corresponding sMB position in each of the

movement maps. In order to calculate the number of nonzero coefficients of each sMB, the script “calculate_coef_smb.m” is called. By giving as input PIC and indicating the index of the MB within the frame and the index of the sMB, this script scans PIC and, when it finds the sMB indicated by the 2 previous indexes; it calculates the total number of nonzero coefficients of this sMB.

Then, the “create_mv_map.m” script stores the value in the corresponding position within the coefficients map. This procedure is repeated in order to fill all the positions (sMBs) within the coefficients map (frame) with the respective number of nonzero coefficients of each sMB.

APPENDIX D

D.1. PRINCIPAL COMPONENT ANALYSIS OVERVIEW

Principal Component Analysis (PCA) is a standard tool in modern data analysis characterized for being a simple, non-parametric method for extracting relevant information from confusing data sets. It provides a way to reduce a complex data set to a lower dimension in order to reveal the sometimes hidden, simplified structures that often underlie it (9).

It is generally accepted that the earliest descriptions of the technique now known as PCA were given by Karl Pearson in 1901 and Harold Hotelling in 1933.

The central idea of the PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables while retaining as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the "Principal Components" (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables (10).

Therefore, the goal of PCA is to identify the most meaningful basis to re-express a data set. Having a data set, the PCA will rotate the original data space such that the axes of the new coordinate system point into the directions of the highest variance of the data. The axes or new variables correspond to the PCs which have the following properties:

- **The PCs are ordered by data variance:** In other words, the first PC is aligned with the direction of maximum variance, the second PC in the next direction contributing to the most variance, and so on.
- **The PCs form an orthonormal basis:** They are all mutually perpendicular and have unit length. This gives PCs the useful property of being uncorrelated.
- **Each PC is a linear combination of all the original variables**

Low variance can often be assumed to represent undesired background noise. Hence, PCs with larger associated variances represent interesting structure, while those with lower variances represent noise. The dimensionality of the data can therefore be reduced, without loss of relevant information by extracting a lower dimensional space covering the highest variance. This can help to reduce the dimensionality of the data set even more while retaining the most meaningful information (9) (11).

APPENDIX E

E.1. RANKING RESULTS FOR "HUSKY.QCIF"

E.1.1. GOP 190 (FRAME RANGE [190-199])

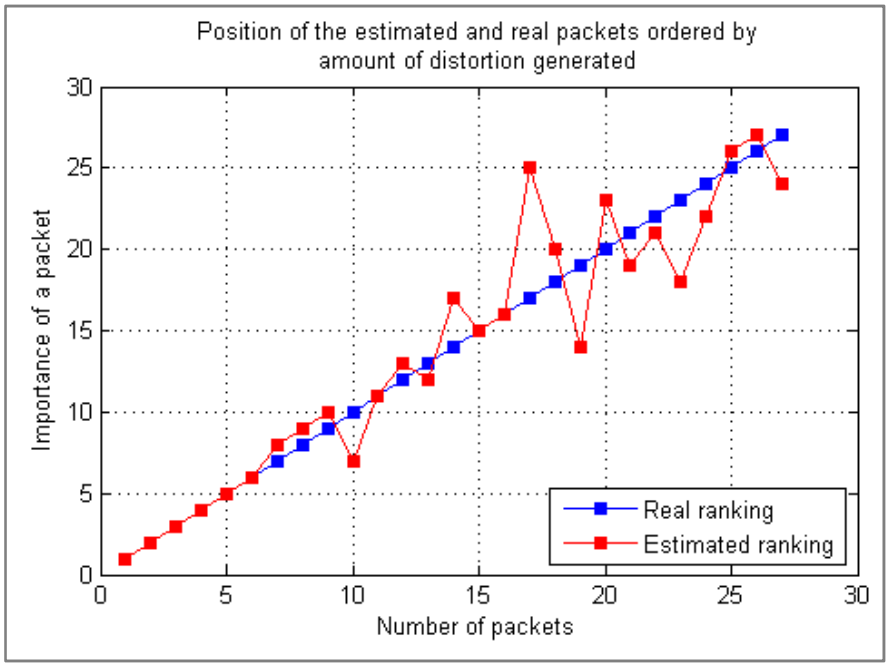


Figure E.1 Comparison between the "real ranking" and the "estimated ranking" (GOP [190-199])

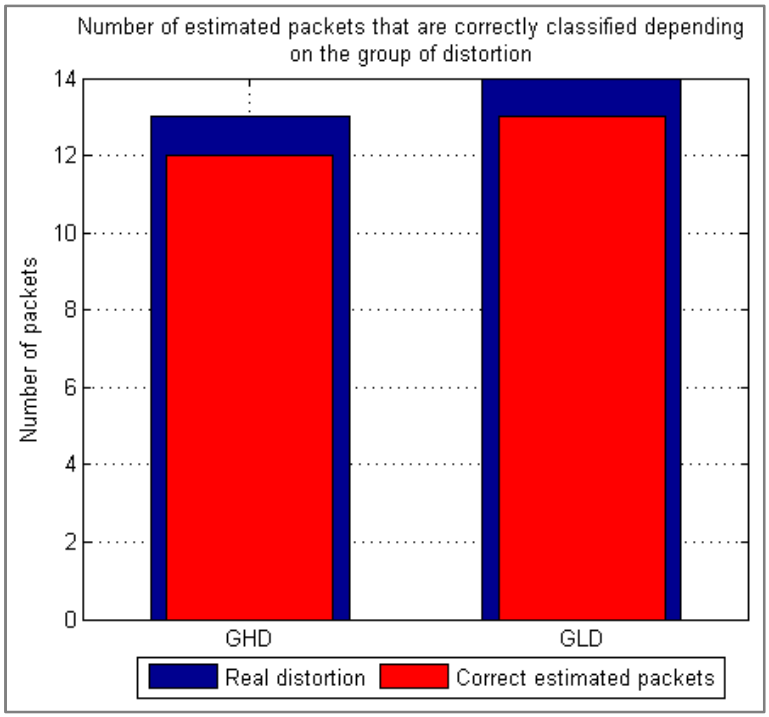


Figure E.2 Amount of correctly estimated packets in each group (GOP [190-199])

In this case, Figure E.1 shows that the differences between the "estimated ranking" and the "real ranking" are higher than in the case of the GOP [170-179] (Figure 5.3) analyzed in section 5.2.1. In Figure E.1 the "estimated ranking" follows the real one until approximately the 10th position where the differences between both start to be noticeable and, in average, they are about 3-4 positions. However, the scheme used to classify the packets in 2 groups show results (Figure E.2) comparable to the ones for the GOP [170-179] (Figure 5.4): 12 out of 13 packets are correctly classified into one group and 13 out of 14 are correctly classified into the other group. Therefore, approximately 93 % of the packets are correctly classified.

E.1.2. GOP 210 (FRAME RANGE [210-219])

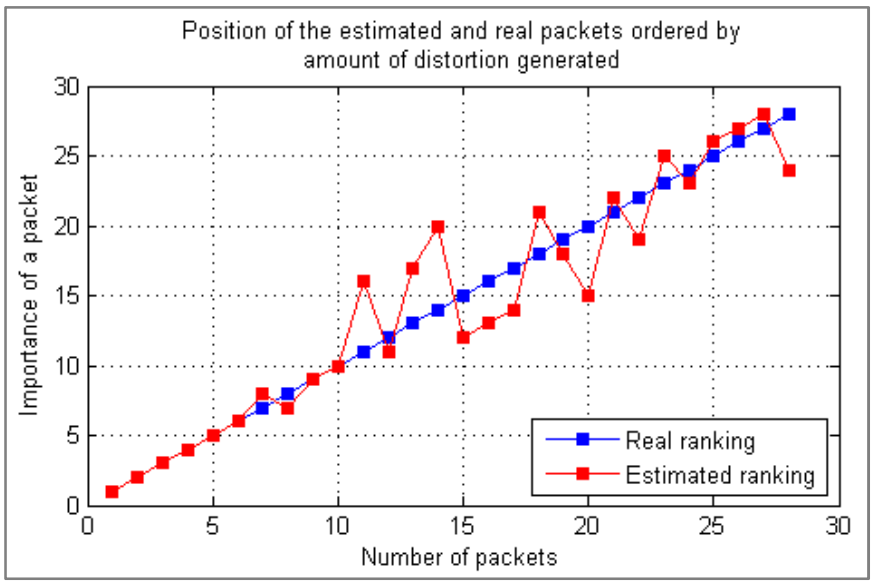


Figure E.3 Comparison between the "real ranking" and the "estimated ranking" (GOP [210-219])

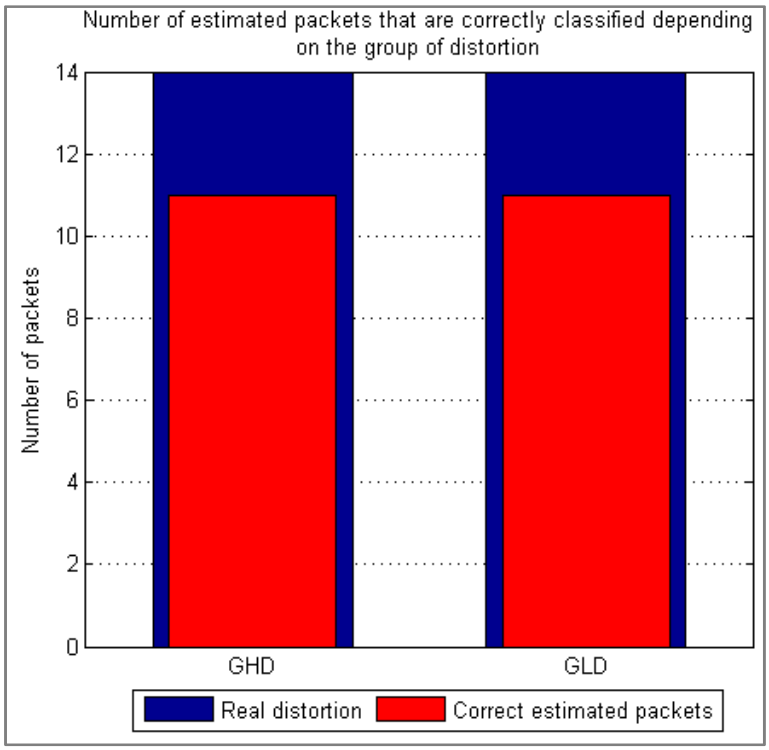


Figure E.4 Amount of correctly estimated packets in each group (GOP [210-219])

Although the differences between "estimated ranking" and "real ranking" are in average among 3 positions too as shown in Figure E.3; since the differences concentrate in the middle positions of the ranking, when classifying the packets into each group, it is observed in Figure E.4 that a smaller amount of packets are correctly classified into each group. In this case 11 out of 14 packets are correctly classified in each group, which implies that 78 % - 79 % of the total amount of packets is correctly classified.

E.1.3. GOP 220 (FRAME RANGE [220-229])

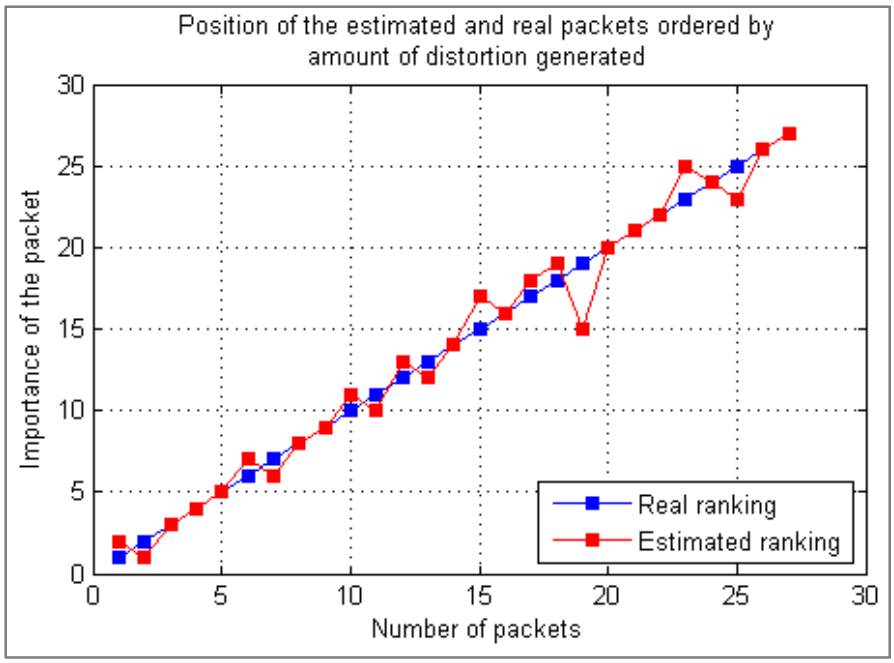


Figure E.5 Comparison between the "real ranking" and the "estimated ranking" (GOP [220-229])

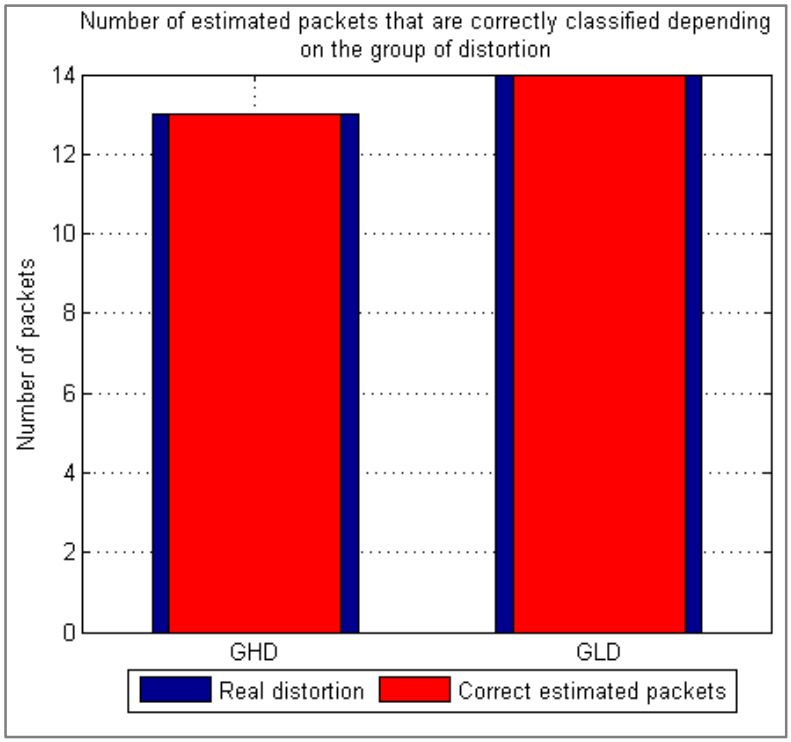


Figure E.6 Amount of correctly estimated packets in each group (GOP [220-229])

Figure E.5 shows that the “estimated ranking” in this case follows the “real ranking” with a difference of 1 position. Nevertheless, there are some cases (concentrated on the final positions of the ranking) where this difference increases to approximately 3 positions. The results of the classification of the packets show that the 100 % of the packets are correctly classified into each group (Figure E.6).

Taking into account the results in the case of the GOPs [190-199] and [210-219], approximately 90 % of the total amount of packets of these 3 GOPs are correctly classified into each group. If the results for the GOP [170-179] are also considered (Figure 5.4), then the amount of correctly classified packets increases to around 93 %.