Definition of the best strategy for buffer control of a multistream ultra low delay HDTV codec

by

Oriol Lluch Parellada

Tutor: Dr. Marco Mattavelli

EPFL (École Polytechnique Fédéral de Lausanne)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Supervisor: Dr. Renzo Posega

LiveTools Technology SA

Lausanne (Switzerland)

February 2007 - July 2007

Abstract

High-definition TV introduces new technologic challenges. Hardware platforms in charge of encoding and decoding high-definition video sequences, have to be adapted in order to process effectively the huge non-compressed bitrate that the high-definition generates.

For this reason, a parallelization technique, called multistream, has been used to encode the video sequences. The multistream technique consists on transforming a high-definition video sequence onto many standard-definition video sequences, by splitting the original frame shape into smaller parts. Each standard-definition sequence can be now encoded by a present codec.

However, the multistream coding approach generates some cutting edges into the original frame. Each portion of the original frame will be coded independently of the others, so, the motion estimation block of the codec will be affected by these discontinuities.

Furthermore, television broadcasts often require that the broadcasts must be done in real-time; that is, the delay introduced by the video coding and decoding chain should be less than 40 ms. This requirement introduces new restrictions onto the codec design, as well as, the implementation of new characteristics to adapt it to run in real-time.

The buffer in charge of transforming a variable bitrate into a constant bitrate, also known as, rate buffer, is the most important element which introduces a delay into the codec chain. So, if it is desired to reduce the delay, the buffer size must be reduced, as well. However, a small sized buffer increases the probability to fall into unstable conditions and to overflow. Therefore, the algorithm to control the rate must be modified to react quickly to rate variations.

This project has been developed in cooperation with the enterprise LiveTools S.A. LiveTools aims to implement a hardware platform for the encodage of the video sequences coming from a professional camera and its later modulation in order to be sent via a wireless link. The encoding process is based on a modification of the standard MPEG-4 Visual codec.

My project is a part of the codification part of the LiveTools project. The main objective of my project is to design one rate control algorithm that maximizes the quality of the images while preventing any undesired situation, such as, the buffer overflow or the system instability.

As secondary objectives, some characteristics have had to be implemented to adapt the codec to the multistream approach, and with an ultra low delay. As it was used the multistream approach, it has also been studied its behavior in comparison to the classical singlestream approach.

In order to reduce the codec delay, the following characteristics have been implemented: *No B-pictures GOP*, *Slice-based QP update* and *Slice-based intra-refresh.*

For the rate control algorithm, the criteria to evaluate the algorithms is as follows; the algorithm should maximize the quality, it must be stable, that is, not to fall into oscillations and it must react quickly enough to cope with rate variations. Some tests have been performed with different algorithms, and the one that best met the requirements mentioned, was elected as the algorithm to be implemented in the codec. This algorithm is a non-linear modification of a PID controller.

The comparison between multistream and singlestream has been done by encoding and decoding a set of six high-definition video sequences with both approaches. The results obtained show a close similarity in quality of the images between both approaches. However, some dependency with the nature of the video sequences can be deducted. Singlestream approach performs better when encoding sequences with a high degree of motion whereas multistream approach is better when encoding sequences with textures highly detailed.

Although the set used for the comparison between multistream and singlestream might not be big enough to come up with a conclusion, it can be remarked that, in a first approximation, using the multistream approach for the encodage of high-definition video sequences might be feasible.

This project covers the description of the characteristics implemented in the codec, as well as it mentions why they are needed and especially it states the main advantages they bring to the codec.

The algorithm chosen to be implemented in the codec is also presented, together with some graphs of its behavior. Some alternative algorithms that were discarded are also shown along with the reasons that make them to be discarded. In that chapter, the algorithms are shown following a problem-solution evolution, ending with the algorithm chosen for the codec.

It is interesting to note, that for low bitrates, 20 Mbps, in some sequences with highly detailed textures some artifacts can be visualized.

For this reason, the implementation of a blurring filter has been proposed. The filtrage should be done before the encoding process in an attempt to reduce the information of the input sequence and relax the requirements of the encoding process. One expects the codec to output video sequences with fewer artifacts at the price of a less detailed image, as the people often prefer seeing a less detailed image than seeing the artifacts.

This project allowed checking the stability and the performance of the high-definition TV ultra-low delay codec by software simulation. The next step will be to realize a hardware implementation of the codec in order to evaluate it and confirm the software simulation results.

Contents

List of Figures

List of Tables

Glossary

ATSC	Advanced Television Systems Committee
Block	Group of 8x8 pixels
CBR	Constant Bitrate
CODEC	enCOder - DECoder
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DSP	Digital Signal Processor
DTV	Digital Television
DVB-T	Digital Video Broadcasting - Terrestrial
DWT	Discrete Wavelet Transform
ETSI	European Telecommunications Standards Institute
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
GOP	Group of Pictures
HD	High Definition
HD-DVD	High Definition - Digital Versatile Disk
HDTV	High Definition Television
ITU	International Telecommunication Union
Macroblock	Square group of 16x16 luma pixels and 2 chroma blocks of 8x8 pixels
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
NTSC	National Television System Commitee
PAL	Phase Alternating Line
PID	Progressive, Integrative, Derivative
PSNR	Peak Signal to Noise Ratio
SMPTE	Society of Motion Picture and Television Engineers
SDTV	Standard Definition Television
USA	United States of America
VBR	Variable Bitrate
VLC	Variable-length Code
VOP	Video Object Plane
YUV	Color Space Model

Acknowledgments

I would like to thank Dr. Marco Mattavelli for the confidence placed on me and for allowing me to work in his laboratory, which it has been an excellent experience. I also want to thank Ali "the informatic *chef*" for his help on setting up the computers and France for helping me at any time, even before arriving in Switzerland.

I want to especially thank Dr. Renzo Posega for his constant guidance and support during this 6 months which helped me in achieving the goals of the project. I also appreciate the support received from Olivier and Serge.

I also want to thank all the friends who shared this experience with me, Agnès, Roser, Xavi, Miguel, Pablo, Andreu, César, Anna, Eva, Marina, and Lea that made the stay in Lausanne really worthwhile. Also to the friends in Vilafranca and my family for their constant support.

Chapter 1

Introduction

One of the main human being necessity is the communication. Communication is a topic in continuous evolution due to, in great measure, the increasingly necessity to communicate at large distances. First appeared the voice telecommunication, i.e., the phone, afterwards, once the voice was consolidated as a mass communication medium and the technology had advanced enough, it came the image transmissions and the birth of the analog TV broadcasts took place.

By that time digital technology was not yet developed and the only way to achieve reliable transmissions was by using analog communications techniques. We are talking about the not yet dead, NTSC and PAL systems. During all its live, analog systems contributed to think on the perception of TV broadcasts as an undetachable item of people life. In addition the advances in microelectronics and the capabilities of microcontrollers, DSPs, FPGAs, along with their affordable cost, let digital systems to emerge. Digital systems have many advantages over analog systems; as a result, nowadays, many analog systems have been replaced by digital systems.

Turning to digital systems simplified and made signal processing more flexible, as a consequence, many different applications became technically possible. The digitalization applied to video sequences allowed higher compression ratios with little loss of quality compared to the previous analog systems, this implied that on the same bandwidth of the transmission channels, better quality image could be sent and more programs can be included.

The next step towards the goal of improving the quality of the broadcasts, was to switch from the analog system to the digital system maintaining the main image characteristics, such as, width-height ratio and image resolution. Due to the better bandwidth efficiency of the digital video, more channels could be broadcasted at the same time. This technology is exploited by the standards adopted on USA and Europe, which are DTV and DVB-T respectively.

Finally, the near future turns to the viewer and tries to approach TV and cinema. This new standard is named HDTV, that stands for *high-definition television*. This technology is mainly focused on increasing video definition and visual angle at what TV is watched. Increasing the image definition will sharpen the images thus giving a more detailed image; on the other hand, an increase on the angle of vision at which the spectator watches the screen is said to give a more perceptual sensation of 'telepresence'[1]. Therefore, at the end, the overall image quality will be improved.

[1]Telepresence is the creation, by the viewer, of a sense of being physically present at an event [1].

Only recently such workload has become sustainable in real-time applications for commercial products, thanks to advances in microelectronics.

In fact, these huge bitrates must be reduced to efficently store or transmit the high-definition contents.

Compression engines push the limits of current hardware to the maximum, e.g. clock frequency, to provide enought processing power to reduce the bitrate to feasible numbers aimed to transmit over wireless links or storage on HD medias (Blu-ray, HD-DVD).

The well-known DPCM/DCT codec model that underlies MPEG2/MPEG4/AVC compression algorithms inherently generates a variable bitrate (VBR) stream, due to the variable amount of exploitable redundancy of the input video sequence, unless some mechanism are deployed to adapt compressed quality to achieve a constant bitrate (CBR) bitstream. Constant bitrate streams are more adapted to wireless transmission.

Furthermore, when wirells real-time applications require low end-to-end delay, some additional constrains arises, and the quality control algorithm becomes increasingly important.

1.1 Scenario

Sometimes it is important to know the motivations behind a specific project because, it can give a wider point of view towards it which normally helps to understand why things are done and the path followed to reach the goal.

This project is, indeed, a module within the framework of a bigger project, from now on referred as ‘FPGA project’. FPGA project aims to implement an ultra low delay HDTV encoding/decoding on a FPGA platform. FPGA project main features are as follow:

Input format SMPTE 295M/296M HD-SDI 4:2:2 up to 1080i60 and/or 720p60

Output format SMPTE 295M/296M HD-SDI 4:2:2 up to 1080i60 and/or 720p60

End-to-end delay $<$ 40ms

Compressed Bitrates 20,30 and 40 Mbps

FPGA project can be divided into three parts, *input,encoding/decoding* and *output.*

Input Accepts the raw uncompressed video data to be coded; it is responsible to store it in the memory and manage all the memory access requests done by the encoding/decoding part.

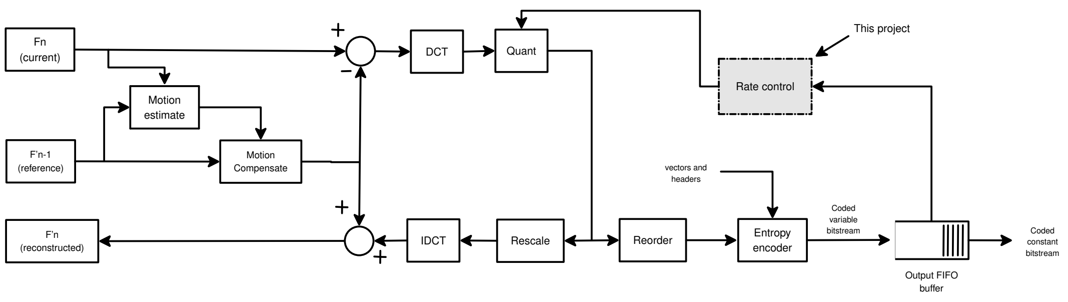
Encoding/Decoding This part is mainly an implementation of four ultra low delay MPEG4 codecs working in parallel, which will give the ability to code high definition video sequences.

Output Adapts coded data coming out from the four encoders into a suitable form to be transmitted on the physical medium. Provided packetisation and generation of MPEG-2 transport streams.

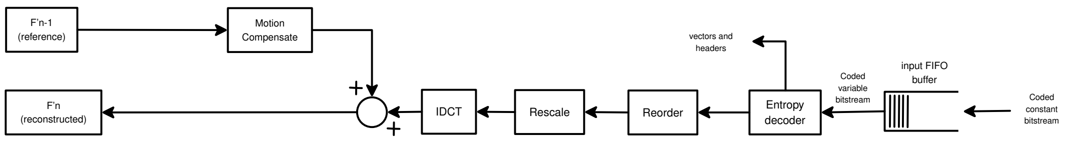
As may have been seen already, this project is enclosed within the development of the *encoding/decoding* part of the FPGA project.

Now that the project scope has been presented, let's go deep into the *encoding/decoding* part explaining where into the CODEC chain this project can be placed.

The codec used in this project is based on the well known hybrid DPCM/DCT video codec model [2]. This work is represented in figure 1.1a as an additional feedback block connecting the *output FIFO buffer* and the *quantisation block* on the hybrid DPCM/DCT video codec model.



(a) Encoder



(b) Decoder

Figure 1.1: DPCM/DCT video codec

1.2 Goal of the project

The main objective of this project is to develop an algorithm that controls the output buffer[2] of an ultra low delay HDTV multistream codec.

On the other hand, rate control algorithm depends, in a great measure, on the buffer input bitrate, and as different coding features produce different bitrates, the algorithm needs to be tested on the complete codec. So implementation of all codec features is a must to achieve the project main goal, and has consisted on important preliminary work. These features are explained in section 6.3.

Moreover, since a single encoder cannot handle the HDTV bitrate requirements, a multistream approach has been adopted. Consequently, tsts have been done to compare performance with the singlestream approach. This constitutes the second important preliminary work of this project.

The output buffer managing algorithm should maximize the quality of the coded video sequence while preventing any overflow or underflow of the rate buffer. To do so, the algorithm uses some feedback information from the buffer and adjusts the *qp parameter* accordingly.

[2]The codec is intended to be used on a constant bitrate medium, as a consequence an output FIFO buffer is required. See section 5.

Due to the ultra low delay requirement, the size of the TS buffer is quite limited, then any little variation on the input image complexity could lead to an overflow. Fast response of the algorithm is mandatory.

The buffer control strategy should prove to be stable in very different scenarios.

The complexity (in terms of exploitable redundancy) of the input video sequence can vary greatly, thus leading to a high variance of the compressed bitrate.

Since the compressed stream will be transmitted by a constant bitrate channel, this variance must be averaged by the “small” rate buffer and the QP adaptation algorithm should perform well in all situations.

1.3 Structure of this document

This document is divided in three parts. In the first part, some background concepts that conform the base of this project are given; this material is contained in chapters going from chapter 2 to chapter 5.

Next, the second part, which is the main part of this document, comprises the following three chapters. It covers the work done on this project; it starts with chapter 6 on the description of the CODEC where all the tests have been performed. Subsequently chapter 7 talks about the comparison between multistream and singlestream approaches; the tests carried out, along with the results obtained and the final conclusions are given. Chapter 8 focuses on the main goal of this project, the *rate control algorithm*; it begins introducing the quality considerations that have been adopted in order to evaluate each of the algorithms tested, then the different algorithms considered during the project are presented, and to sum up, the results and conclusions are given.

Lastly, the third part of this project, comprising the last chapter, it covers the global conclusions and the future work.

Chapter 2

Video coding

This chapter is devoted to present digital video encoding. It is meant to be a recall of some techniques used by current encoders to compress video sequences. Current encoders take advantatge of some properties of the human visual system that allows lossy compression without significantly reducing the perceived quality.

2.1 Introduction

Due to the importance of radio spectrum, transmission channels are usually bandwidth limited. As bitrate and bandwidth are proportionally, one should minimize the video sequence bitrate. The process of compacting data into a smaller number of bits is called *compression*. Video coding is the process of condensing a digital video sequence into a smaller number of bits.

Even though, prior to display, the video sequence need to be reconverted to its original source data. Therefore, the compression needs its counterpart, decompression, which transforms compressed data back into a representation of the original data. Thus, that compression is a process that comprises two parts, a compressor (encoder) and a decompressor (decoder). The encoder/decoder pair is often described as a *codec*.

Data compression is achieved by removing all the information that is not useful for a perceptually correct representation of the data; this useless information is called *redundancy*. Redundancy can be *statistical*, which can be effectively removed applying lossless compression techniques, i.e. the output representation can be reconstructed back to its original form, without any loss of information. However, *lossless compression* of video sequences gives only a moderate compression ratio. Therefore, *lossy compression* is necessary in order to achieve higher compression ratios, but at the expense of a reduced video quality. Lossy compression is based on the principle of removing *subjective* redundancy, i.e. remove information that is not useful for the receiver sensor, in video sequences, the human eye. As a result, the output decompressed data is not an identical representation of the original data.

A three dimensional space conforms traditional video sequences, two spatial dimensions and time. Most video coding methods exploit both *temporal* and *spatial* redundancy to compress video data. It can be seen that video frames that are close together in time are highly correlated, i.e. they have little differences. In addition, inbetween the same frame, pixels that are close to each other are also highly correlated.

2.2 Video codec model

The codec transforms the original video sequence following a *model*, i.e. a coded representation useful to reconstruct an approximation of the video data. In the following paragraphs the codec model is explained, it begins with the encoder and then it follows with the decoder.

Many encoders can be represented by three functional units: a *temporal model*, a *spatial model* and an *entropy encoder*. These parts can be seen in figure 2.1. The original uncompressed video sequence is the input to the temporal model. Then, the temporal model tries to reduce temporal redundancy exploiting similarities between neighbour video frames, normally, by predicting the current video frame from one or more previous or future frames. The output data of the temporal model is a set of 'vectors' that represent the prediction and a set of residual samples that represent the difference between the predicted frame and the original frame.

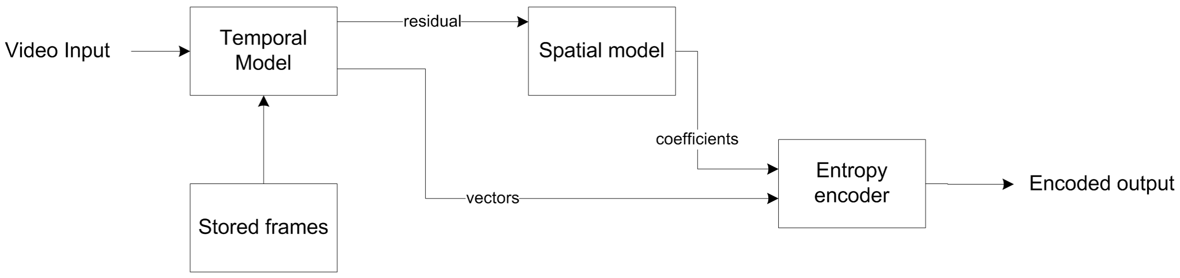


Figure 2.1: Video encoder diagram

Afterwards, the residual samples are piped into the spatial model. The spatial model transforms the residual samples to another domain where is represented by 'coefficients'. In the transform domain it is easier to delete the useless information by zeroing some of the coefficients; this process is called 'quantization'. It should be noted that 'quantization' is the point in the encoder where some information is lost and, as a result, the quality is degraded. At this point, the output parameters are the remaining 'coefficients' coming from the spatial model, and the 'vectors' from the temporal model, are given to the entropy encoder that will remove the statistical redundancy and will produce the compressed bitstream suitable for transmission or storage.

The decoder does the inverse operation, it starts with the compressed bitstream, which is given to the entropy decoder, then transformed coefficients are transmitted to the spatial model, whereas the vectors are given to the temporal model. Finally, the residual information that produces the spatial model is used by the temporal model, along with the vectors, to reconstruct the predicton done at the encoder and add to it the residual information in order to produce the original frame. Note that as some information is lost during the quantification process, the reconstructed frame at the decoder will not be the same as the original frame.

2.2.1 Temporal redundancy

One of the simplest methods to remove temporal redundancy is to perform the difference between the current frame an the previous frame in the video sequence. Then as the two images are highly correlated the difference will have much less energy that each frame alone. So the number of bits required to code the frame difference, from now on referred as residual frame, is less than the bit requirement for a video sequence frame. However, the decoder must receive information about the prediction that has been performed, and the way it has been made, in order to let it inverse the process.

Even when many temporal redundancy hab been removed, there's still some energy in the residual frame that could be removed if it is taken into account that most of the remaining energy comes from the motion in the video sequence. Then, if some kind of motion compensation is done instead of pure consecutive frames difference, a better predicted frame will be achieved thus reducing the difference between frames and, as a consequence, minimizing the energy that remains in the residual frame, achieving better compression.

In most modern codecs, motion prediction is done by two main functional units, motion estimation (ME) and motion compensation (MC). Usually a block based method is used when a rectangular region of the current frame is compared with a neighbour region of the reference frame, this neighbour area is called 'searh area'. The process that tries to find the best new localization for each block of the reference frame is called motion estimation. On the other hand, MC constructs a new frame with the blocks of the reference frame but reordered as the ME actually said. This motion compensated frame is further substracted from the reference frame to obtain the residual frame.

By changing block size and searching area dimension many different levels of accuracy can be achieved. For further information, see [2].

2.2.2 Spatial redundancy

At this point, video sequence data has gone throught a temporal redundancy remover unit, therefore there is a great part of information in the residual frame. The main techniques used to remove spatial redundancy are described in the following.

Predictive Image Coding A technique that exploits similarities between previous encoded pixels. Figure 2.2 shows a pixel X that is to be encoded. If the frame is processed in raster order, then pixels A,B and C are available in both the encoder and the decoder (since these should already have ben decoded before X). The encoder forms a prediction for X based on some combintion of previously-coded pixels, substracts this prediction from X and encodes the residual, i.e. the result of the substraction. The decoder forms the same prediction and adds the decoded residual to reconstruct the pixel.

Transform Coding Is the process that converts some input data to another domain, called the transform domain. In the transform domain the data should meet some requirements:

1. Data should be decorrelated
2. The conversion should be reversible
3. The conversion should be computationally feasable

There are many transforms that meet the above requirements but almost all of them can be classified into two groups: block-based and image-based. One example of a block-based transform is the *Discrete Cosine Transform* DCT, where the transformation is done in blocks of $N \times N$ samples and hence the image is processed in units of a block. Block-based transforms have low memory requirements and are well-suited to

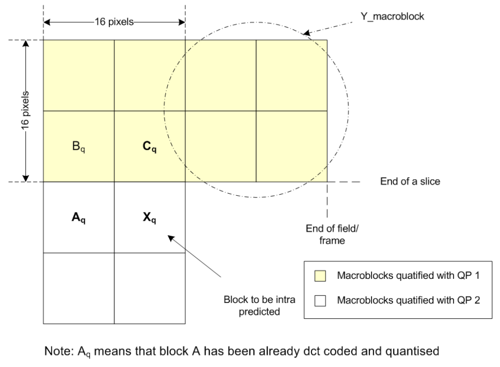

16 pixels
16 pixels
Y_macroblock
B_q
C_q
A_q
X_q
End of a slice
End of field/ frame
Block to be intra predicted
Macroblocks quatified with QP 1
Macroblocks quatified with QP 2
Note: A_q means that block A has been already dct coded and quantised

Figure 2.2: Intra prediction

compression of block-based motion compensation residuals. In contrast, image-based transform perform better for still images. Image-based transforms operate on an entire image or frame. An example of an image-based transform is the *Discrete Wavelet Transform* DWT.

In most modern codecs, DCT is widely used because of its good performace when motion-compensation is used in the temporal model.

Detailed information on the mathematics of the transforms is ommitted as it goes beyond the scope of this document. Further information can be found in [2],[3] and [4].

Quantisation It is the action that consists in mapping a range of values A to a reduced range of values B. A *scalar quantiser* maps one sample of the nput signal to one quantise output value and a *vector quantiser* maps a group of input samples to a group of quantised values. In this process, as the range of output values is smaller than the input range, the recovery of the original input value is not possible leading to a loss of information; as a result, quantisation is a lossy process. The bigger the range of values of B the better the approximation to the original range A thus preserving more information, that means quality, but the worse compression achieved. The number of values in rage B is controlled by a parameter called *step size* or simply *QP parameter*.

Reordering and Zero Encoding The quantification process, by its nature, generates a high number of zero valued transformed coefficients. As it is necessary to compact, as much as possible, the information in order to reduce the number of bits to represent it, transformed coefficients are reordered in orther to pack all zero coefficients toghether so that they could be more efficently represented prior entropy coding. Further details can be found in [2].

2.2.3 Entropy coding

This functional block works with the data that comes from the temporal and the spatial units. At this point, the data that represents the video sequence is supposed to be free of the majority of the spatial and temporal redundancy, but now the data at the input of the entropy coding can be modeled as an information source with some statistical redundancy, that is, some data values are more probable than others. So, source coding techniques should be applied prior transmission or storage to minimize the bitrate. Some of the methods are described in the following.

Variable-length Coding A variable-length encoder maps input symbols to a series of code-words (VLCs). Frequently occurring symbols are represented with short VLCs whilst less common symbols are represented with long VLCs. Over a long period of time, the statistical properties of the data are better approximated thus achieving the desired compression of the data. An example of a variable-length encoder is the *huffmann coding*. See [2] for further details.

Arithmetic Coding An arithmetic encoder converts a sequence of data symbols into a single fractional number. This approach can approximate better the optimum number of bits[1] to represent each symbol that is usually a non integer number. See [2] for further information.

[1]To achieve optimum compression, each value should be represented with exactly $\log_2\left(\frac{1}{p}\right)$ bits where p is the probability of occurence of the value

Chapter 3

HDTV

In this chapter, what is the concept laying behind the *high definition* word is first explained. Then it follows with a brief description of image resolution and image definition. Next, some transmission standards are presented for high definition video. On the next section, MPEG4 Visual Profiles and Levels are described in order to show what to expect on the coding sheme. Finally, an important part to mention is the problems that HD will bring to the codec architecture.

3.1 Introduction

Nowadays, video coding techniques, along with advances in electronic technology, makes feasible to increase the quality of the video sequences that the people see at their homes. So this opportunity to go a step forward is has not being ignored by the big consumer electronics companies and the TV contents providers. This new technology is known by *High definition tv (HDTV)*. The point is to know what is, technically speaking, behind this name. Further information can be found in [1], [5] and [6].

3.2 Standards defining High-Definition TV

After many years[1] of discussions, standard proposals, recommendations and disagreements, the ITU finally could reach some kind of an agreement for a HDTV standard which is specified in ITU-R BT.709-3.

Actually ITU did not manage to achieve the goal of a unique standard, and there are two systems which are considered into the standard, one from the United States of America, where 1125 scanning lines and a refresh rate of 60Hz are considered, and the counterpart system, promoted by Europeans, with 1250 scanning lines and a refresh rate of 50Hz. In contrast, there is an agreement on a *common image format* for HDTV, which is 1920×1080 pixels per picture width$\times$heigth.

Apart from ITU, regional standerization organizations, such as ATSC and ETSI, promoted its own standards for HDTV adding a new standard with less spatial resolution. Standards adopted in Europe under the DVB projet are detailed in table 3.1.

[1]The first discussion about HDTV standarization was held in 1986

System	720p	1080i	1080p
Resolution (pixels)	1280 × 720	1920 × 1080	1920 × 1080
Aspect ratio	16:9	16:9	16:9
scan mode	progressive	interlaced	progressive
Chroma sampling	4:2:0	4:2:0	4:2:0
Frame-rate (Hz)	24,25,30,50,60	25,30	24,30

Table 3.1: HDTV standards characteristics

By the way, nothing about the encoder method has been mentioned. The reason is that at the time of approval of the standards, only MPEG-2 video codec was available so the standard was entitled to use MPEG-2. Afterwards, MPEG-4 and H.264 reached maturity, and due to the much better compression ratio is probable that the H.264/AVC will be used instead of the now overcomed MPEG-2. For further information, see [1],[6] and [8].

3.3 High Definition new constrains

This project is software-oriented work, but is tied to the implementation of a hardware version of the codec. So the hardware limitations need to be taken into consideration. HDTV demands for a cutting edge hardware technology, and so, as not yet available for production, some different approaches must be taken into account.

Normally, a simple data flow system can be definined by three component parts, *input*,*processing* and *output*, where an input is handled, processed and outputed.

A compromise between time and resources arises in the way the process is done. One might do the processing in a sequential manner, where all the process tasks are done one after the other, or in a parallel manner, where different and independently tasks are carried out at the same time. As many tasks are done at the same time, parallel approach requires less time to perform the processing than the sequencial approach but more resources are required. Then if faster response is required, more resources will be needed and viceversa.

Something similar happens with video coding. When input raw video sequences bitrate increases, e.g. sequences with greater resolution and/or frame rate, encoder throughput must be increased in order to handle that bitrate. That is the case of the HDTV video sequences, that they have a much bigger bitrate compared to actual SDTV video sequences.

The easier solution is to increase the clock frequency at which the encoder works, due to the direct relation between encoder troughput and working clock frequency. But the clock cannot be freely increased and often this approach is not a feasable solution. Then, it might be useful to switch to parallelization of the codec itself. In other words, try to run different, not interdependent funcional units at the same time. This solution has been studied in recent years and all of the current codecs implement some kind of parallization.

However, the encoding methodology has by its own nature, some parts which are not parallelizable, which limits time reduction. Therefore, if the encoder is already parallelised and the clock frequency is at its maximum, but the encoder is still not fast enough, what else can be done? Again, parallelization is the response, this time one should increase its field of view and go for input parallelization. That means that the input, e.g. video frames, should be splitted into several sub-frames and coded in parallel by many encoders.

In this project, the encoder parallelization is achieved by the FPGA itself which gives great parallelization facilities; in contrast, FPGA is limited when it comes to clock frequency. When FPGA technology is used to handle HDTV video sequences it is not capable of handle them due to the high demanding throughput of the input sequence. It is time to move into input parallelization, also referred as multistream coding, which is explained in detail in next chapter.

Chapter 4

Multistream coding

This chapter is devoted to an encoding parallelization technique where the encoding of a video frame is not done by a unique encoder but for many, thus an input singlestream is splitted in various streams, and each of them is encoded independently.

Firstly, introductory part covers a detailed description of the multistream approach and the problems that motivated its use. Next, the frame partition plan is given. Subsequently, the advantatges and disadvantatges of the multistream implementation for video encoding are pointed out. To sum up, some conclusions are given. Some similar work on multistream can be found in [9].

4.1 Introduction

At present a hardware IP core of a single stream video codec is available. It can handle higher input throughputs by increasing the clock frequency. However, HDTV input bitrates are so high that the IP core clock should be pushed further its limit.

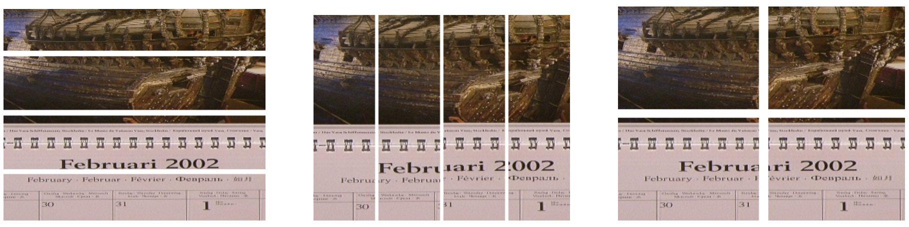
Therefore, if a unique codec is not capable of handling HDTV input bitrates, some parallelization must be applied. The parallelization consists on splitting the input image frame and send each portion of the frame to a singlestream codec. All codecs will encode its portion independently of the others and at the same time. As a result, to the transmission medium, more than one encoded bitstream will be sent.

At the decoder side, each encoded bitstream will be decoded by a singlestream decoder. Then the decoded output will be joined, in order to obtain the original frame size.

4.2 Image partition

Image partition refers to the action of splitting a frame of a video sequence in smaller parts, let's call them, subframes. There are many options on how to split the image, it can be done in rows, i.e. horizontal cuts, or in columns, or it can be cut in square regions by cutting each time at half the picture height and/or half the picture width. In figure 4.1 different splitting types are shown.

The image partition shown in figure 4.1a it performs well in case the majority of the movement present on the sequence is horizontal, that is because the motion prediction would be better. On the other hand, figure 4.1b will perferom well in environments with a lot of vertical movement, it has to be said though that, there is more horizontal than vertical movement in natural sequences. Finally, figure 4.1c shows a balancing approach between horizontal and vertical movement.



(a) Rows (b) Columns (c) Squared

Figure 4.1: Image partitions

4.3 Advantatges / Drawbacks

The main advantatge of the multistream approach is that it allows to encode a real-time video sequence that a singlestream approach could not encode due to the high demaninding technological requirements.

However, multistream approach has a performance drawback when used with encoders based on motion prediction, because when a frame is splitted, the ‘motion searching area’ is always reduced near the cutting edges, thus preventing motion estimation between subframes and increasing the probabilities that a macroblock near the edge is intra encoded, lowering the compression performance.

4.4 Conclusions

Since the prevalent movements in natural video sequences is due to horizontal moving, the row splitting approach of figure 4.1a offers the best compression results. Anyway, this approach will raise the minimum end-to-end delay to at least 3/4 of frame period, due to pipelining constraints of the four encoders.

The less efficient column splitting approach, see fig.4.1b, offers the lowest end-to-end delay.

The approach of figure 4.1c has been referred to achieve the best possible compression efficiency while suffering of a latency limited to about half of a frame.

Chapter 5

Delay and Rate Control

5.1 Introduction

The information of a video sequence is in its nature, variable in time. That means that an encoder with constant parameters, e.g., quantization step size, motion estimation search area, etc. will produce more bits when the input sequence has more information, i.e, more detail or higher motion, and less bits otherwise. Therefore, one might expect a variable output bitrate from any generic encoder. Furthermore, the bitrate variation not only comes from the input sequence, but also from different encoded pictures types, e.g. I-picture, P-picture and B-picture, that will require a different amount of bits to code the same source image. For further information and different studies about rate control, see [10], [11], [12] and [13].

However, some scenarios might require the encoder to output a constant bitrate. For example a circuit-switched channel, which is a constant bitrate transmission medium. Storage medias usually have a fixed storage capacity, so they require to control the variable bitrate, up until the point to be able to store the desired video sequence into the media.

Consequently, it is desireable to 'smooth' the variable bitrate. The trick can be achieved by buffering the data prior transmission. A FIFO buffer is employed. Figure 5.1 shows the typical arrangement where a variable bitrate is converted into a constant bitrate.

The arrangement is such that the encoder variable bitrate is piped to the buffer; then the buffer is emptied by the channel at a constant bit rate.

At the other end, another FIFO buffer is placed at the input of the decoder and is filled at the channel constant bitrate. The decoder buffer is emptied, by the decoder, at a variable bitrate (since the decoder extracts P bits to decode each frame, and P varies).

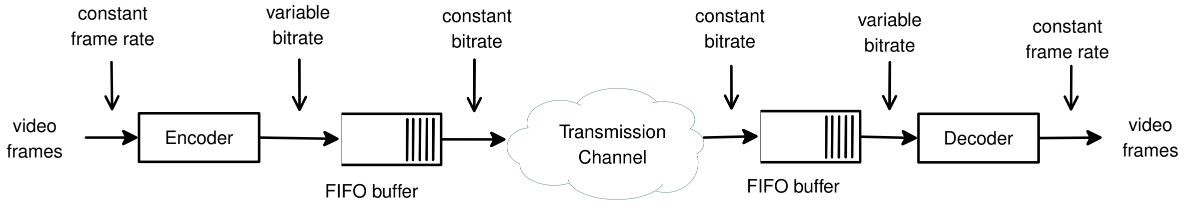


Figure 5.1: Variable to contant bitrate conversion

5.2 Delay

It has been seen, that a pair of FIFO buffers are needed in order to adapt a variable bitrate to a constant bitrate and viceversa.

However, introducing FIFO buffers adds an extra delay to the encoding/decoding process, which is function of the chosen CBR and the buffer size, as stated in equation (5.1). In other words, if a constant delay is required, then buffer size is dependant on the desired bitrate. In figure 5.2 delay chain is detailed.

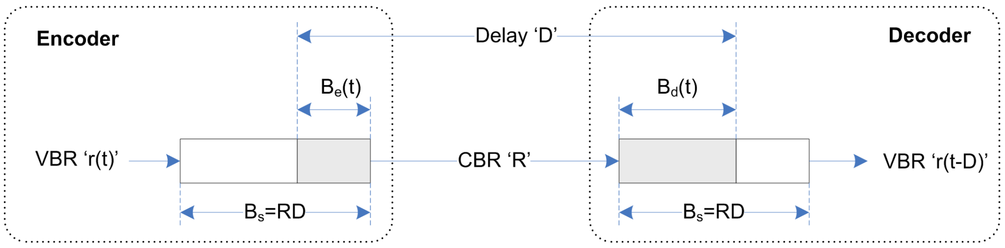


Figure 5.2: FIFO buffers delay

Where buffer capacity B_s, as well as buffer fullness $B_e(t)$ and $B_d(t)$ are given in ‘bits’, delay D is given in ‘seconds’ and all rate values, $r(t)$, $r(t-D)$ and R are given in ‘bps’.

If the encoder and decoder buffers are well synchronized, it is possible to prove that the sum of their fullness is a constant B_s equal to the buffer size [10], as shown in figure 5.2 and stated in equation 5.1.

$$\begin{aligned} B_e(t) + B_d(t+D) &= B_s \\ R \cdot D &= B_s \end{aligned} \tag{5.1}$$

Note that the delay introduced by the encoding/decoding process it has not been considered. As it is normally much smaller than the delay introduced by the FIFO buffers.

Anyway, to evaluate the total end-to-end delay, the latency given by the multistream approach should be added (half frame period).

5.3 Rate control

In the pasts sections, it has been shown that a variable bitrate can be adapted to a constant bitrate transmission channel by adding a buffer in both encoder and decoder. However, the inclusion of the buffers comes at the expense of increased storage size and delay. It can be noted that the bigger the bitrate variation, the bigger the buffer size needed. Furthermore, as the bitrate variation comes in a great part from the source video sequence, the buffer size would have to be adapted to the worse case, resulting in impractically big buffer sizes and delays.

As a consequence, as the encoder output bitrate can be adopted, it should be desirable to implement some kind of feedback mechanism that would act on the encoder quality, in order to prevent the buffer from over- or under-flow. The modification of the encoding parameters to adapt the encoder output bitrate to a target bitrate is known as *rate control.* The most obvious parameter at which one should point out, is the QP parameter, because it has a direct relation with the final output bitrate.

On the other hand, the algorithm can obtain more information apart from the buffer occupancy, and that could give the algorithm the quality to predict the bits, that will require to code the current frame. As an example, some information can be extracted from the motion estimation block, evaluating the motion vectors for some randomly chosen macroblocks; if no valid motion vector have been extracted, then it is more probable that those macroblocks will be intracoded, thus increasing the output bitrate.

The rate control algorithm should be adapted to the encoder target application, as it will have different restrictions on each case. For example:

Offline encoding for storage. In this case the time is not a constrain, so the rate control algorithm can be as complex as needed, and is often used to achieve the best results.

Real-time encoding for streaming. In this particular scenario, time becomes the main actor. As the delay must be kept small, the buffer needs to be also small, so a fast algorithm is advised in order to react, to bitrate changes.

5.4 Low Delay characteristics

The delay in a codec is a mesure of the time elapsed from the moment when the input video sequence is delivered to the encoder, until it exits the decoder. For this consideration the channel has been assumed to deliver data instantanously with no loss of generality.

In the following the nomenclature for different time delays will be presented. The names were adopted along with the appearence of new video applications that need to reduce the delay of the generic standards, for example, the video-conferencing often requires a delay less than 500 miliseconds, and to differenciate it from the standard delay, the name *low-delay* was adopted. Afterwards, the real-time applications needed even less delay, in particular less than 40 miliseconds, so a new name was given to this new delay threshold, *ultra low delay.* These values are shown in table 5.1.

Delay	Value
Normal	≥ 500 ms
Low	< 500 ms
Ultra low	< 40 ms

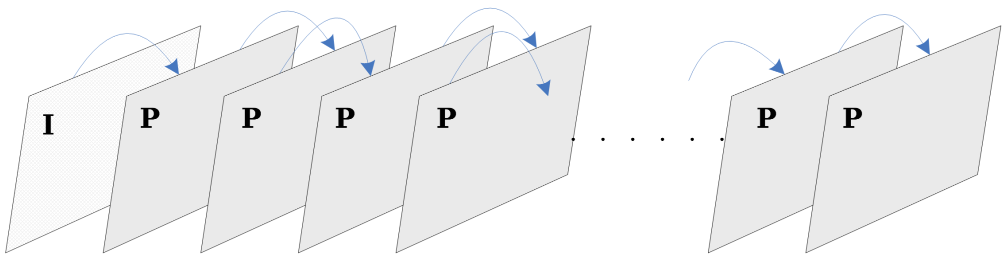
Table 5.1: Some delay values

5.4.1 Techniques used to implement an ultra low delay codec

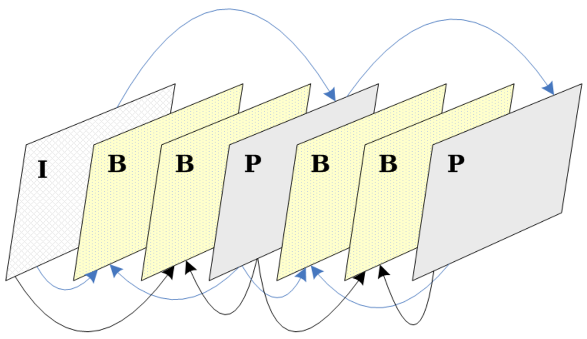
First of all, is worth recalling that fixing a maximum delay directly determines the buffer size. The less delay the smaller the buffer. So, for low and ultra-low delay, the buffer is smaller than the generic codec ones, but the encoder bitrate must be accomodated into the buffer in all cases. Therefore, the techniques that will be exposed subsequently will try to reduce the encoder bitrate, e.g., GOP structure with just one I-frame, and to store, as less information as possible, for the encoding process, e.g., no B-pictures.

Next, the techniques are described in the paragraph below along with the advantatges given and/or the problems that they solve.

No B-pictures The B-pictures are a type of a bidirectionally predicted images. A future frame in the display sequence, and a prior frame are used to predict B-pictures. In consequence, both prior and future frames must be previously encoded and stored, increasing the storage needed and the delay. In the example of figure 5.3b, the two B-pictures following the first I-picture cannot be encoded unless the first P-picture is arrived and has been encoded. Thus, a latency of at least 2 frames (and accounting for encoder processing time, often 3 frames) raises from this scheme. The main advantage of not using B-pictures is that *only the prior frame must be stored and the delay is minimized*. Refer to figure 5.3a.



(a) No B-pictures



(b) Standard

Figure 5.3: GOP structures

Intra refresh slicewise If an error occurs during the transmission, then the only way to recover from it is to code the erroneous region with intra macroblocks. Normally, this refresh is done framewise, that means that after a fixed number of frames, an intra frame, i.e. I-frame, is coded. Figure 5.3b shows the generic GOP structure. However, this approach is not feasible when it comes to small buffer sizes, as an I-frame generates a higher bitrate than the predicted pictures. With a very reduced rate buffer, this would easily cause quality fluctuations and even buffer overflow. Alternatively, what it is done, is to spread the intra refresh among consecutive predicted pictures, each of them refreshing different portions of the frame. After some time, the whole frame will be intra refreshed, obtaining the same result of the other approach, in terms of error recovery. As it has been seen, slicewise intra-refresh must be applied in a low delay codec as *it reduces the variance of the encoder output bitrate*.

Slice qp update This technique is a *direct consequence* of the *reduced buffer size.* In other words, as the buffer is so small, the little bitrate variations that occur during one frame coding could lead to a buffer overflow, so the bitrate should be tuned more often than once per frame, in order to be able to react to buffer variations in an effective way.

Chapter 6

Description of the CODEC

Most of the video compression international standards, e.g. MPEG-4 Visual, H.264,. . . , have been conceived with the idea of performing well in the vast majority of possible applications, i.e. to make it, as generic as, possible. However, there are still some specific applications, where the generic standard, can be adopted.

This chapter is devoted to the description of the features, that where modified or added to the baseline codec, in order to meet the requirements presented in section 1.1.

This chapter aims to describe the platform[1] where the project has been carried out. It describes the CODEC in a temporal evolution manner, so it starts with the source images characteristics, see section 6.2; then it points out the features of the CODEC at the beginning of the present project, see section 6.3.1; then it's the final CODEC features turn, i.e. mentioning the improvements added to it, trying to explain clearly what are the main reasons that motivated its inclusion and what are the problems appearing on each new feature addition.

6.1 Introduction

To achieve the current project goal[2], it was needed a codec that met the requirements for an encoding and decoding of high-definition images with very low latency. This project exploits an MPEG-4 simple profile IP core, which has been modified and adapted to a multistream scenario. Therefore, one of the first steps of this project was to implement the features that would convert the current codec into a multistream ultra-low delay HDTV codec; in particular, slicewise qp update and new intra-refresh schemes. Once every new feature was implemented and tested to ensure good performance, the tests covered in chapters 7 and 8 could start out.

Finally, note that section 6.3.3 is intended as a quick review on the CODEC's main characteristics, describing the way they have been implemented. Once the chapters 7 and 8 will be reviewed, this section will help on answering many of the questions that could arise.

The characristics of the video sequences used for the tests should be carefully analyzed. Since the statistics and complexity of video sequences are greatly variable, some sequences

[1]All the actors that play a role in our testbench CODEC. e.g.source sequences,input-output bitstreams,codec itself

[2]please refer to section 1.2 for more information

have been chosen in order to stress different aspects of the encoding chain. For example, it must be noted that some sequence will highlight the performance of the codec when processing slowly-moving, highly detailed sequences, e.g. calendar, or the performances when encoding fast moving detailed and less predictable sequences, e.g. parkrun.

6.2 Source video sequences

The characteristics of the selected video sequences are specified in appendix B along with a snapshot of each sequence.

6.3 CODEC

6.3.1 Starting point. MPEG4-Visual

At the very beginning, the codec was a partial implementation of the MPEG-4 Visual Simple profile ITU standard.

For more information on the Visual MPEG4 standard, pleses refer to [8].

The most relevant features for this project that where already implemented will be explained in the following.

The paradigm of the encoder architecture has changed, moving from a monolithic approach, where all the parts that constitute the codec, were done by one entity, to a client-server structure, where functions that are performed before the coding itself, such as, image partition, have been moved to the server part, while keeping the client part as the encoding entity. Figure 6.1 shows the server-client architecture adopted with the functional blocks implemented in each part.

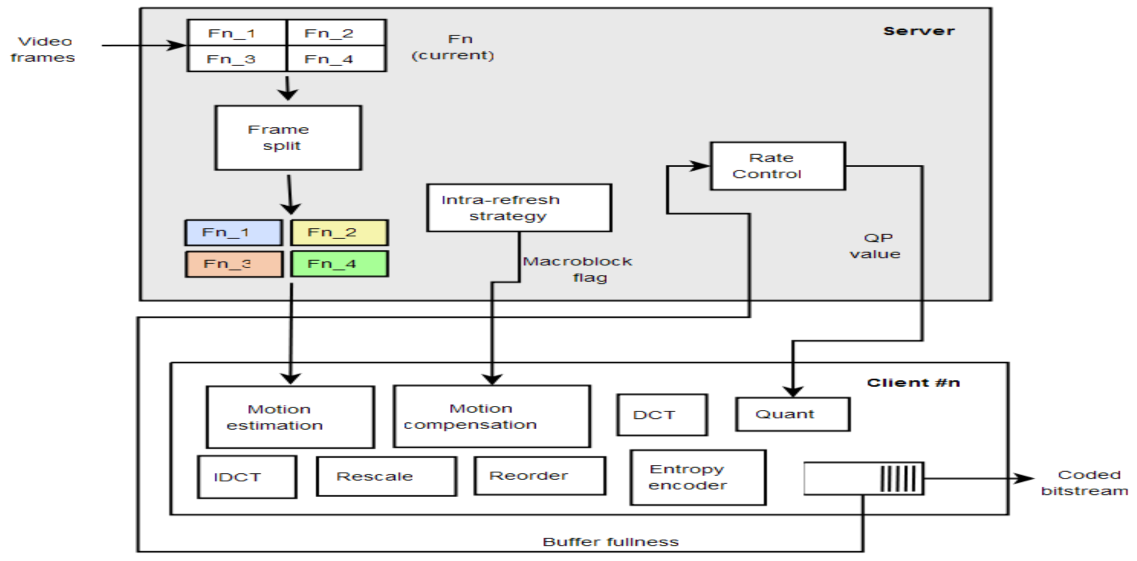


Figure 6.1: Server-Client architecture. Functional units.

The input images are splitted in four parts, as previously described in figure 4.1c.

6.3.2 Features implemented

The features that were implemented during this project are detailed in the following.

Slicewise quantization The QP should be updated every slice instead of every frame to guarantee fast response to buffer fullness changes.

Slicewise intra refresh The limitations of the buffer size prevent using intra frames to avoid error propagation, so the intra refresh will be splitted in time, i.e. consecutive frames will intra-refresh a different part of the frame until its completion after a fixed number of frames; that nuber will vary depending on the way the intra-refreshed is done.

FIR filter Implements a preprocessing low-pass filter, intended to reduce the artifacts, e.g. blockiness, that appear at low bitrates.

6.3.3 CODEC's final features. Summary

Multistream

The multistream codec is constituted by several singlestream codecs working in parallel. The number of codecs working in parallel will be fixed by the image partition schema chosen, having one codec per image part. For the current multistream codec, the number of singlestream codecs has been fixed to four.

However, if we take into consideration that the transmission medium is the same for both the singlestream and the multistream, there's still the constrain in bitrate that the medium will impose, and that constrain is applied to the sum of the n streams, being n the number of codecs, so what is pointed out is that it can be kept the output buffer as a unique buffer, now with n inputs and only one output. As there is only one buffer, it can be drived by a unique rate control strategy which will equally drive the n codecs.

This approach will allow to mantain the overall quality of the frame as constant as possible and thus, preventing the human eye to notice any frame partition. Furthermore, maintaining a unique rate control strategy performs better as it will be given information of n different parts of the frame at the same time, thus taking advantage of image statistics given in chapter 4.

Due to the requirements of global functional blocks, the codec has been implemented with a server-client architecture, where the server is in charge of the image partition, the rate control strategy and of fixing the intra-refresh strategy, and the client does all the remaining job. In figure 6.1 it can be seen the functional blocks of both client and server and in figure 6.2 the information that flows from one to the other.

To sum up, the implemented characteristics of the multistream approach are enumerated in the following.

- *Four encoders*
- *Square image partition*
- *Server-clients architecture*
- *One global rate control strategy. Independent of the number of clients*

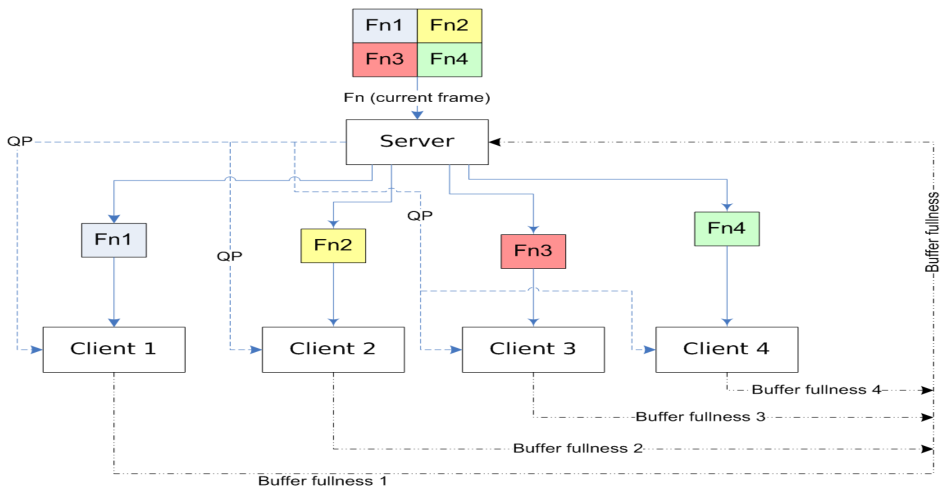


Figure 6.2: Server-client architecture. Data flowchart.

GOP stucture. No B pictures

The GOP structure of the codec is such that no backward prediction is performed, thus, no 'B' pictures are used. As it can be seen if figure 5.3a the GOP starts with an 'I' frame and then it is indefinitely followed by forward predicted 'P' pictures. This is referred as "infinite GOP". This feature is already present in the MPEG4-Visual so it is not necessary to modify the baseline codec.

Slicewise quantization

The slicewise quantization name can be a bit confusing. It refers to the fact that each slice of the first client the 'qp parameter' is updated. It means that with four codecs working at the same time, the 'qp parameter' will be updated every two slices of the original, not partitioned, frame.

The QP update is done by the rate-control strategy which is placed in the server, but the information of the bits used to code the current slice is on the client. So each time that an slice of the first client is encoded, the client sents to the server the number of bits that has been used to code the past slice. Meanwhile the server is waiting for each client to sent him the same information, and once received, it uses the rate-control algorithm described in chapter 8 to output a new QP value to be used for the clients in the next slice. Finally, at the beginning of the next slice, before any macroblock is processed, the new QP value to be used is sent to all the clients, as shown in figure 6.2.

It should be noted that QP parameters not only act on quantisation block, but also on AC/DC prediction. Slice-based QP update would introduce prediction errors in presence of intra-encoded blocks, unless this problem is not accounted for. A detailed explanation of this problem along with the solution given can be found in appendix A.

Slicewise intra refresh

For the intra refresh, the slice unit refers to a slice of the original, not splitted, frame. But again, some matisation should be done. This time, the confusing terms comes from the fact that now, the slice term is adopted as a matter of differentiation from the frame based generic intra refresh.

The intra refresh strategy adopted for the current project is such that half of the slice of the original frame (field in case of interlaced input) is intra refreshed per frame/field. Figure 6.3 shows this procedure. An example illustrates the intra refresh period for a $1080i50$ video sequence.

Half-slice intra-refresh → 68 fields needed to intra-refresh one complete field

$$T_{ir} = \frac{68\ fields}{50\ Hz} \approx 1,36 sec$$

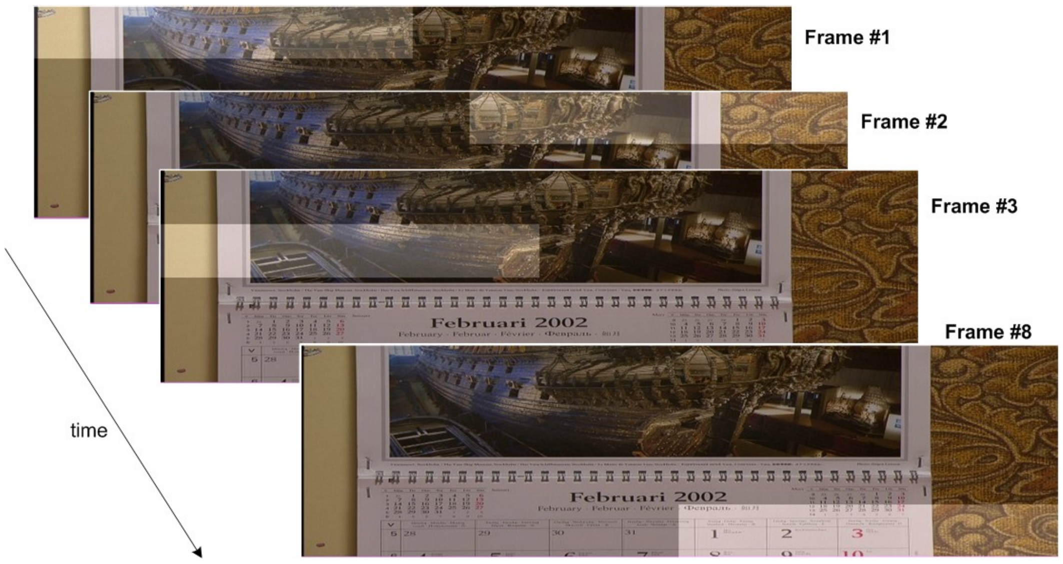


Figure 6.3: Half slice intra-refresh. Frames are supposed to have four slices.

Ultra low delay rate buffer

The capacity of the output buffer has been fixed to allow a maximum delay of 40 miliseconds. Fixed the maximum delay, the buffer's capacity becomes a function of the desired output bitrate, according to the following equation[3].

$$C_{buff} = R \cdot d$$

Where C_{buff} is the buffer capacity in bits, R represents the maximum output bitrate in bits per second (bps) and d is the maximum allowed delay in seconds.

Pre-filtering

A five tap zero-phase low-pass FIR filter has been adopted as it gives enough flexibility in design. The delay that adds to the global encoding-decoding chain can be ignored. In

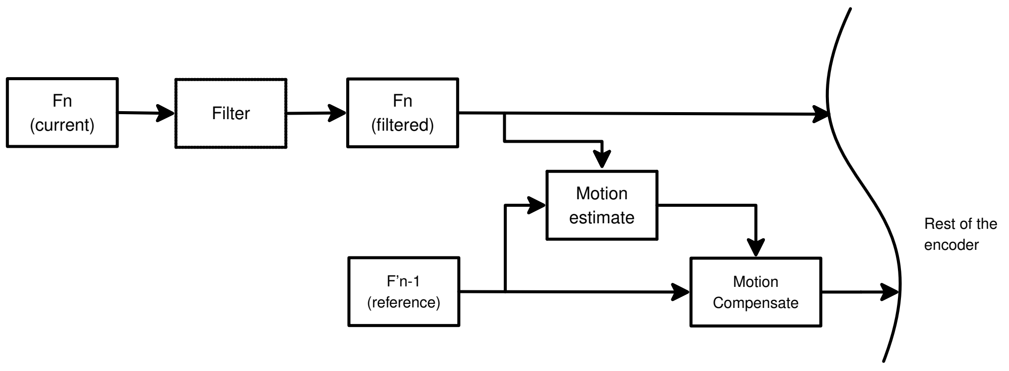


Figure 6.4: Pre-filter into the DPCM/DCT hybrid model

addition, such a low order filter is not very computationally demanding, and fits well into current's project requirements.

Filtering operation will only be performed to the luminance component and along the horizontal axis, so it is a one dimensional filtering. If we think on the background concept of the DCT transformation, we can conclude that if the high frequency components are filtered out the DCT transformation, along with the zig-zag scan, will end with less non-zero coeficients and thus will relax the next step in the coding chain, the quantization block. That means that a smaller QP value could be applied and thus less artifacts will appear. See 6.3, [14], [15], and [4].

In figure 6.4 it can be seen the encoder block diagram with the newly added filtering block.

[3]Further information in section 5.2

Chapter 7

Multistream performance

This chapter is devoted to test the performance of the multistream approach in comparison to the singlestream. The video sequences used for the tests are included in section 7.2.

7.1 Introduction

Theoretically, signlestream is a better approach than multistream, as the image partition introduces some artifacts that motion estimation might not be prepared for. In the case of a not correct motion estimation, some increase in compressed bitrate mightn be expected, which will have to be corrected by the rate control algorithm, reducing the image quality.

7.2 Description of the tests done

To test the multistream performance the same testbench was ran by both singlestream and multistream approach. Within the same testbench several tests have been performed, mostly, at different bitrate. The bitrates considered are $20Mbps$, $30Mbps$ and $40Mbps$. These values represent what is considered low-bitrate, medium-bitrate and high-bitrate for the target application of the codec.

In the beginning a testbench which used the former rate control strategy was used and by the results obtained it could be seen that the results were not useful. The results of this first testbench, referred as 'testbench A', which are given in appendix C showed that the *former rate control algorithm* was not stable as oscillations appeared in some of the testbench tests. For this reason, while the results are given as a first approximation, a new testbench, from now on 'testbench B' with the *final rate control algorithm* was conceived. This time the results, given in section 7.3, were satisfactory and the comparison between multistream and singlestream can be done properly.

Detailed information of the 'testbench B' is given in table 7.1.

7.3 Results

For the comparison of the PSNR values, to find a general response, that means, a response as independent as possible from the input sequences, many tests have been done with different

Input sequences	Mixed, stockholm, pedestrian, bluesky, riverbed, shields
Bit rates	20,30,40 (Mbps)
Intra-refresh	1 slice per frame. Top to botom. Starts on the first P-frame
Rate control algorithm	latest ($qp[n+1] = qp[n] + 0.5 sign(y[n]) y^2[n] + 8x_{med}[n]$). see ch.8

Table 7.1: Testbench B. Description

input sequences and the average value has been taken. Tables 7.2, 7.3 and 7.4 show the results for the different bitrates.

Subsequently, some graphs are shown, where the QP fluctuations and the stability of the buffer could be observed.

It might be worth noting, that the *buffer* is *not overflowing* or underflowing in any of the tests realised.

7.3.1 Testbench B

Sequence	Singlestream	Multistream	Difference[a]
Mixed	30.86782	30.80729	0.06053
Stockholm	35.36594	35.37244	-0.00650
Pedestrian	40.89761	40.77497	0.12264
Bluesky	34.32328	34.39364	-0.07036
Shields	35.25591	35.28144	-0.02553
Riverbed	32.81515	32.80387	0.01128
Average	**35.34211**	**35.32595**	**0.01615**

[a]Difference = Singlestream-Multistream

Table 7.2: Average PSNR (dB) @ 20Mbps

Sequence	**Singlestream**	**Multistream**	**Difference**[a]
Mixed	32.70705	32.75464	-0.04759
Stockholm	36.86103	36.84353	0.01750
Pedestrian	42.12789	42.01397	0.11392
Bluesky	36.46578	36.57548	-0.10970
Shields	37.03164	37.02330	0.00834
Riverbed	35.09102	35.06788	0.02314
Average	**36.71407**	**36.70980**	**0.00426**

[a]Difference = Singlestream - Multistream

Table 7.3: Average PSNR (dB) @ 30Mbps

Sequence	**Singlestream**	**Multistream**	**Difference**[a]
Mixed	34.12411	34.18261	-0.05850
Stockholm	37.77626	37.73105	0.04521
Pedestrian	43.02206	42.88118	0.14088
Bluesky	38.12300	38.20997	-0.08697
Shields	38.13910	38.19418	-0.05508
Riverbed	36.73850	36.70836	0.03014
Average	**37.98717**	**37,98455**	**0.00261**

[a]Difference = Singlestream - Multistream

Table 7.4: Average PSNR (dB) @ 40Mbps

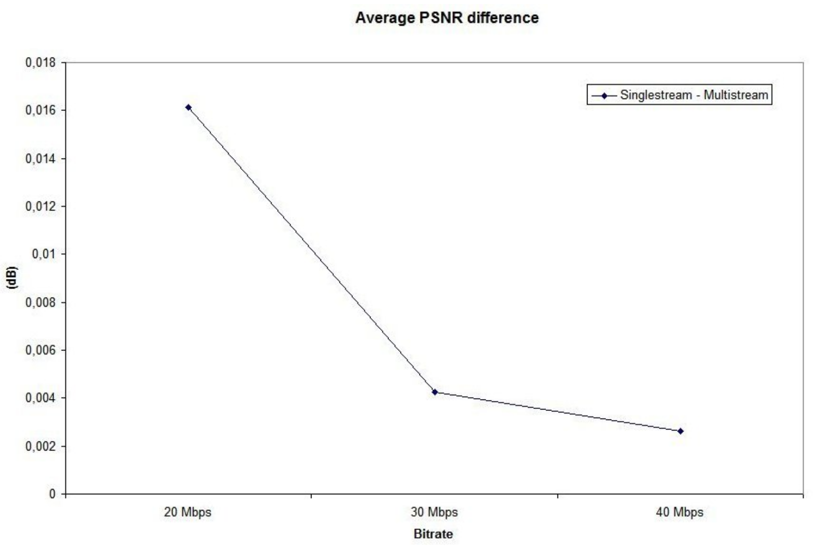
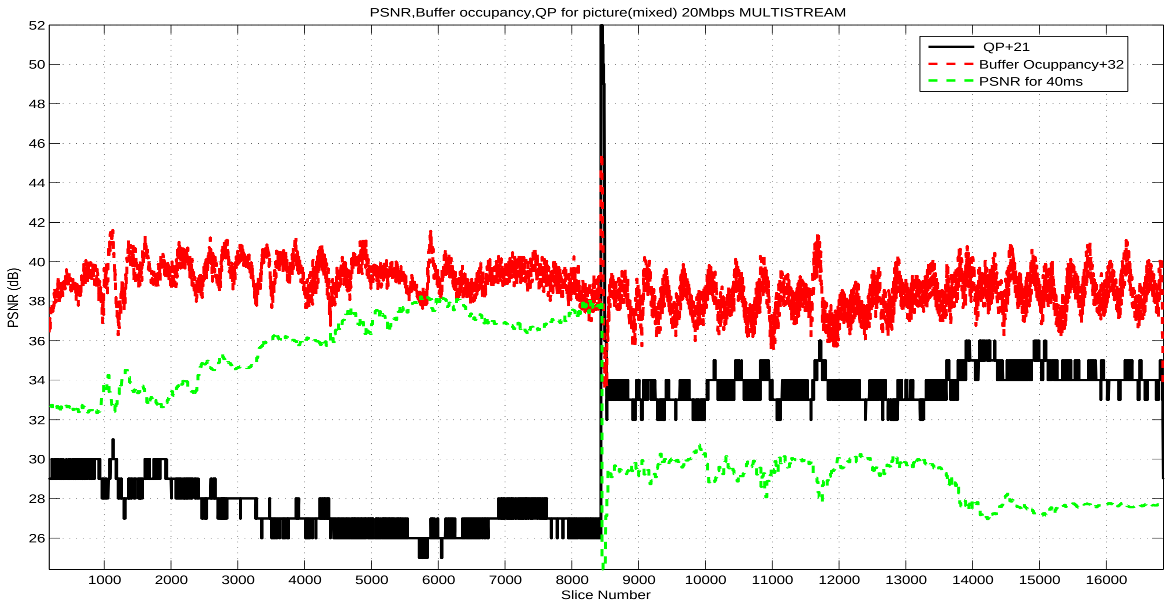
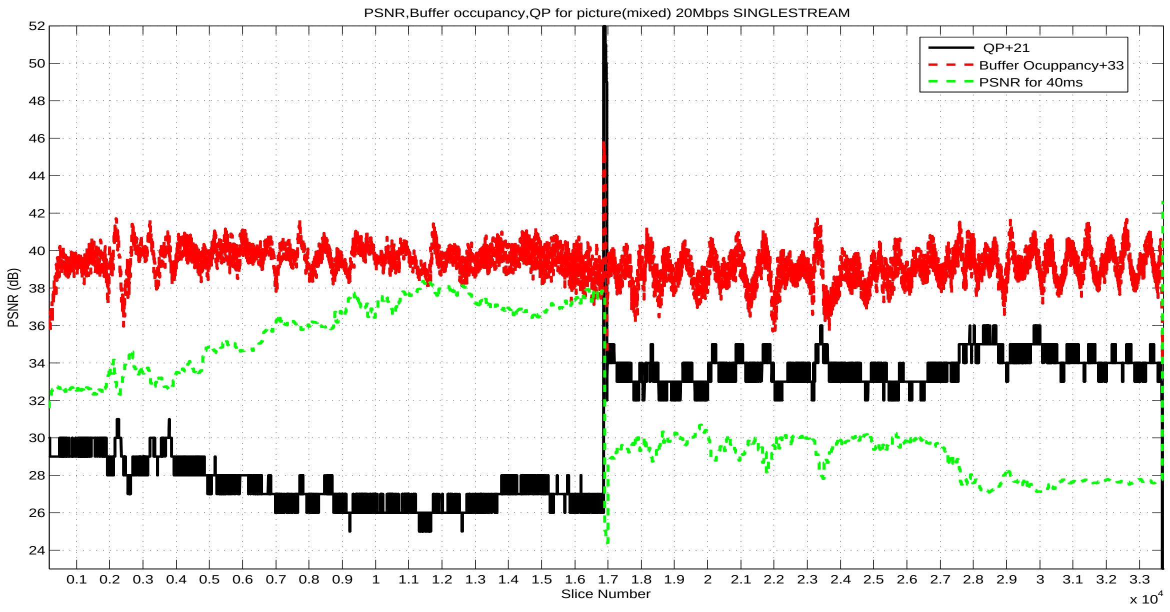


Figure 7.1: Average PSNR difference

It has been considered that *mixed sequence* is the one more important for the comparison, as it has many details, a scene change and the quite demanding parkrun sequence, and represents well the hardest challenge a codec will face. Therefore, the results of *mixed sequence* are presented for each bitrate and streaming scheme, whereas only selected results of the other sequences are given.

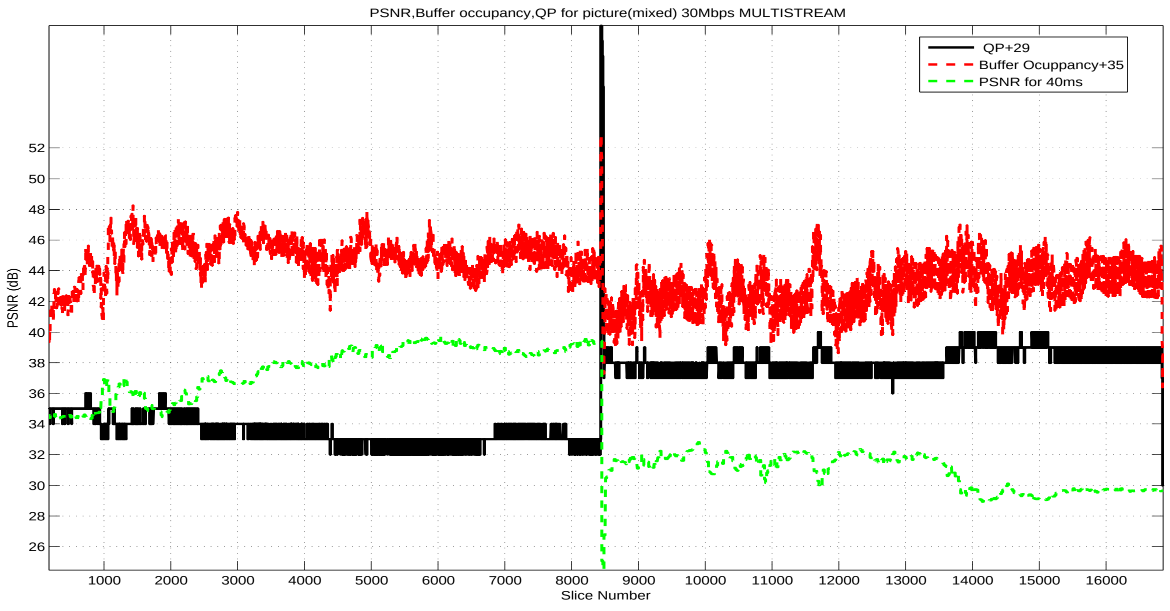


(a) Multistream

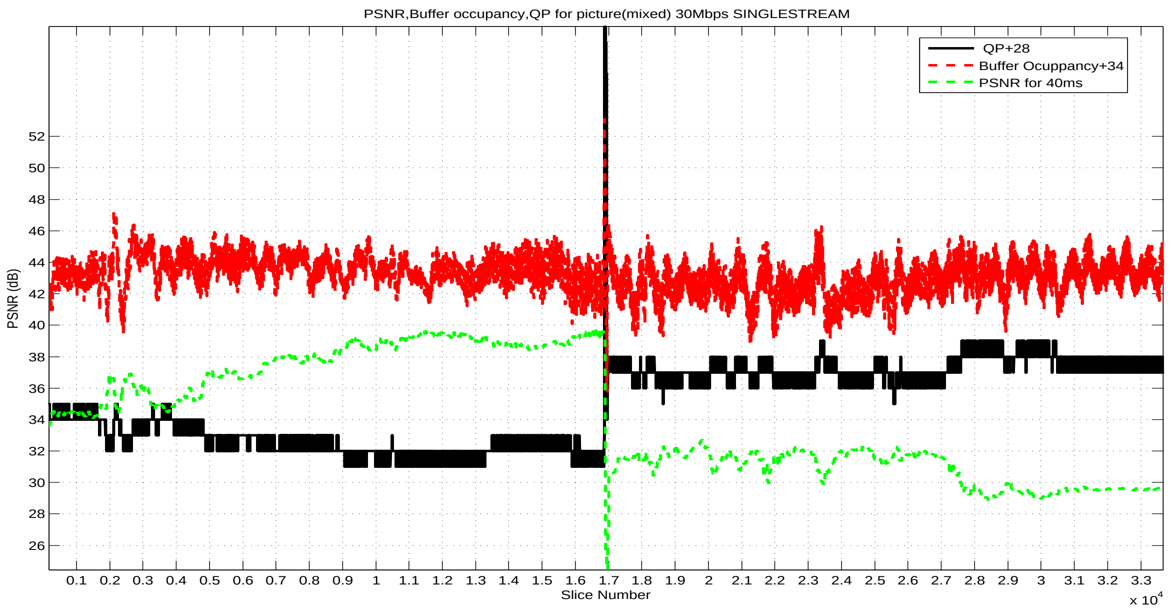


(b) Singlestream

Figure 7.2: Mixed sequence @ 20 Mbps

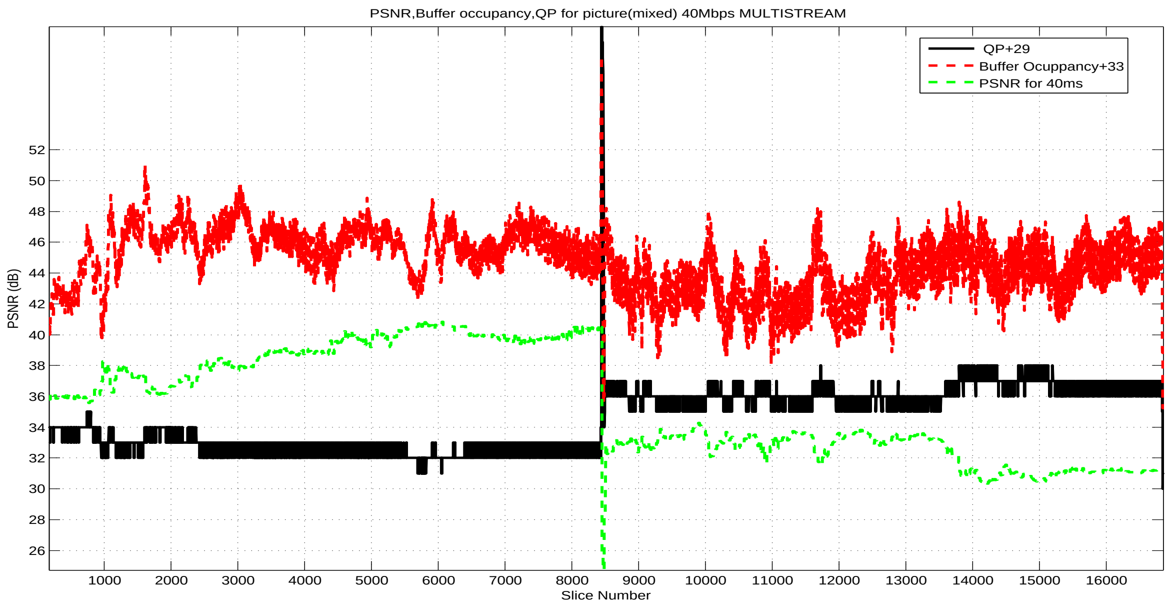


(a) Multistream

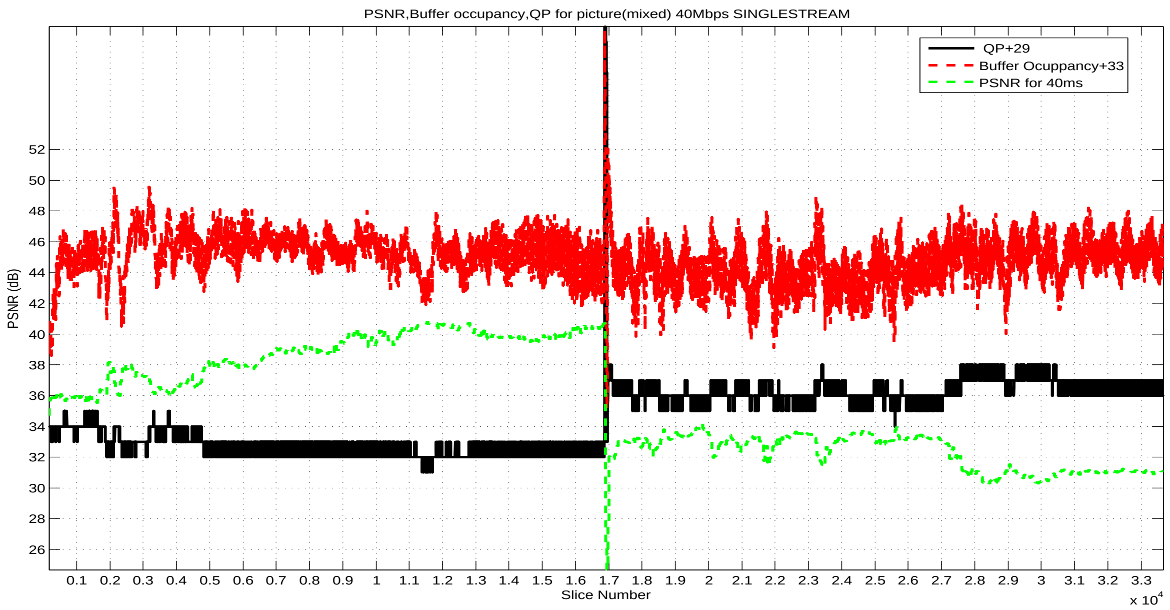


(b) Singlestream

Figure 7.3: Mixed sequence @ 30 Mbps

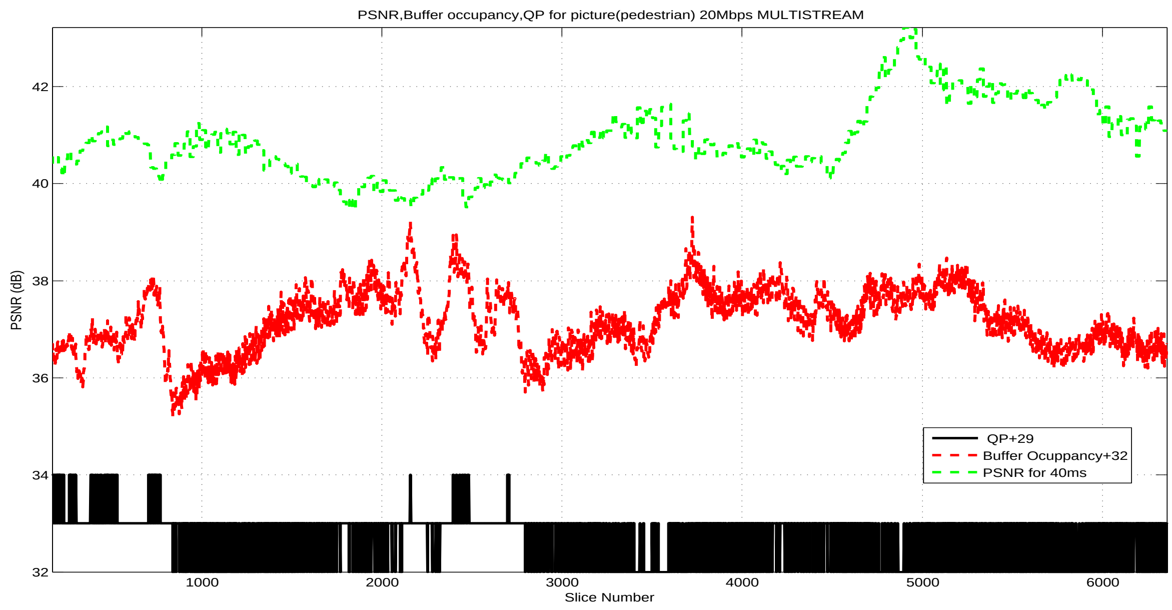


(a) Multistream

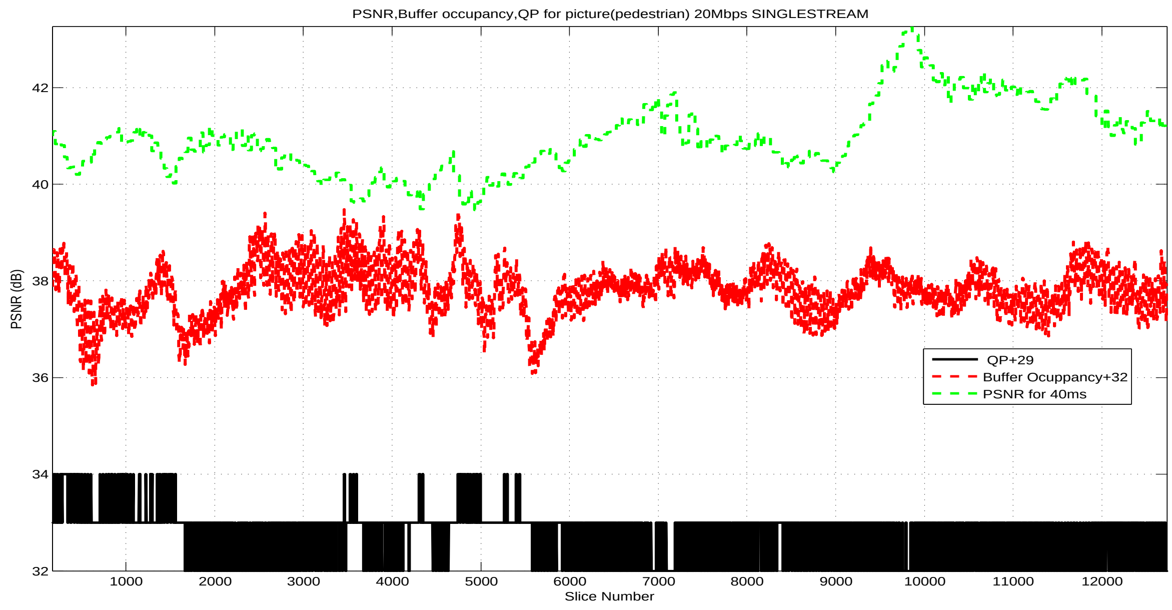


(b) Singlestream

Figure 7.4: Mixed sequence @ 40 Mbps

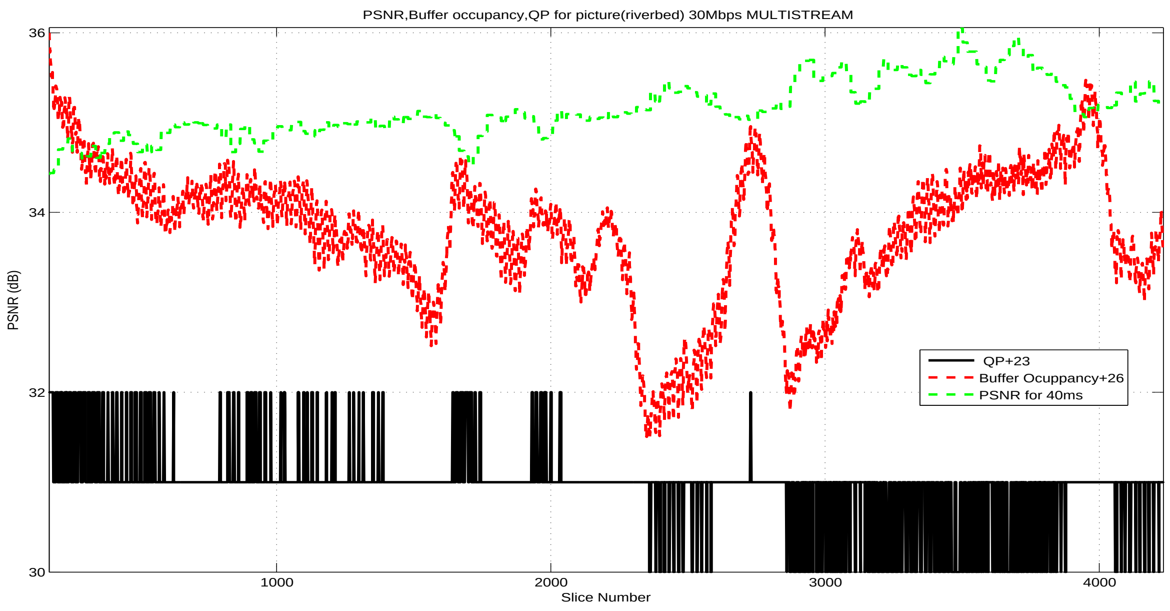


(a) Multistream

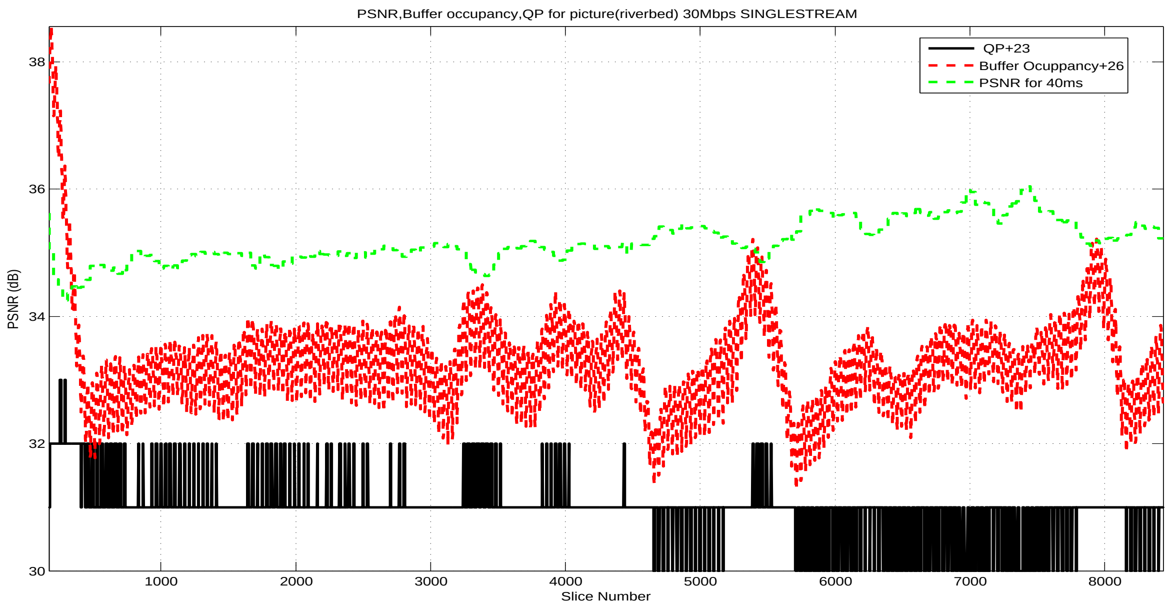


(b) Singlestream

Figure 7.5: Pedestrian sequence @ 20 Mbps

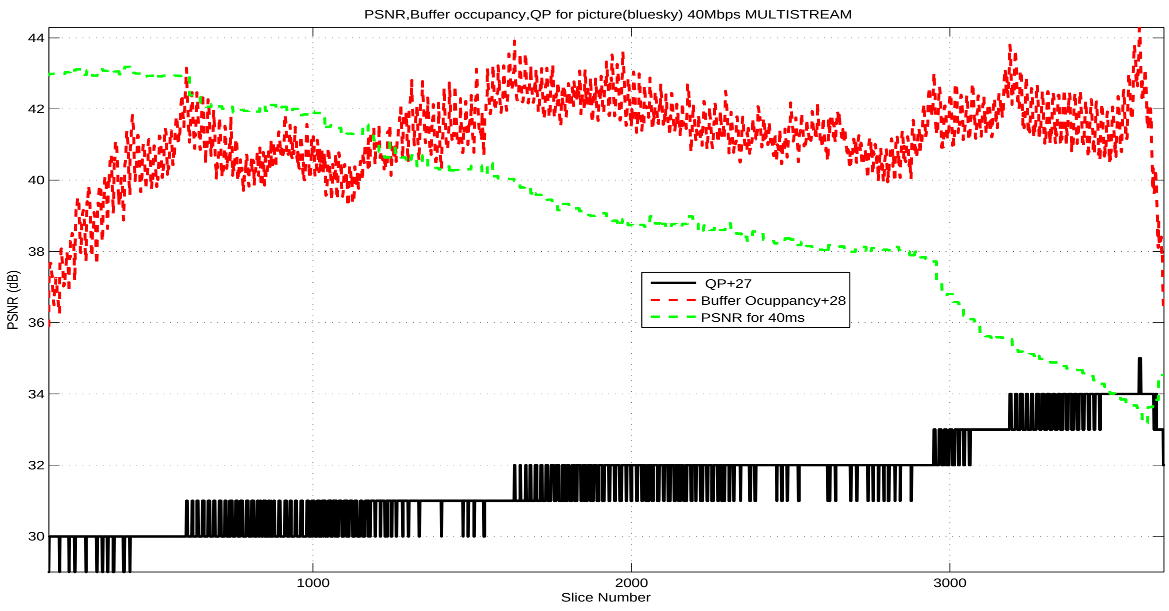


(a) Multistream

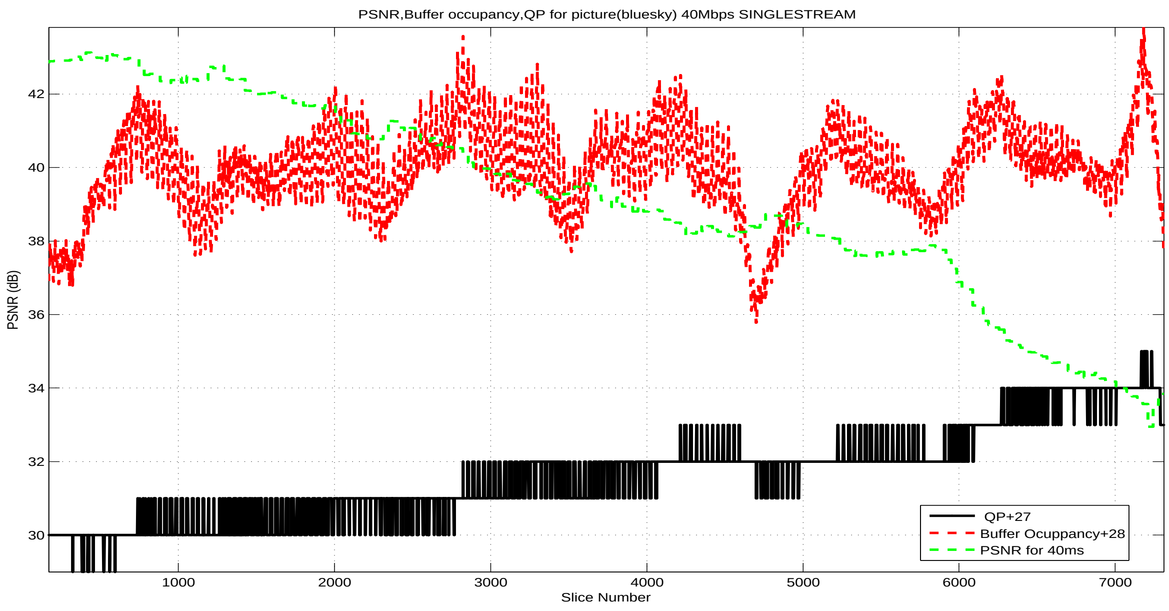


(b) Singlestream

Figure 7.6: Riverbed sequence @ 30 Mbps



(a) Multistream



(b) Singlestream

Figure 7.7: Riverbed sequence @ 40 Mbps

7.4 Conclusions

The results of the average PSNR difference for each bitrate shows that, the difference between multistream and, is much smaller than the smallest PSNR increment considered to be noticeable (0.5 dB). As a consequence, it can be said that,in average, *multistream* performs as *well* as singlestream in terms of quality. Due to the little set of the images used for the average, some more video sequences should be tested in order to give a more accurate result.

However, tables 7.3 and 7.4 show in Pedestrian sequence, which has lots of horizontal moving, that singlestream is better than multistream. Whereas, for Bluesky sequence, where texture detail is more frequent than movement, multistream shows better performance. These results make me think that the results should be presented depending on the application targeted for each codec, because it has been said, for some applications where the predominant aspect is motion, multistream might not be worth, in contrast to applications where highly detailed texture sequences will be used, where multistream can overperform singlestream. So, in my opinion, more tests should be carried out in this direction, to find out the exact impact of motion estimation with multistream.

For the other important parameters, i.e, QP fluctuations and buffer stability, the graphs show that both approaches have a similar behaviour of buffer evolution, and very similar QP fluctuations. In conclusion, no big differences are to be remarked.

Finally, just remark that in the decoded video sequence, the image partition realised in the multistream approach is not visible.

Conclusions can be summarised with the following points:

- Multistream performs as well as Singlestream, in average
- No visual partition seen on decoded video sequence

Chapter 8

Rate control strategy

This chapter main goal is to describe the tests done, in order to find the best *rate control algorithm*, and to present the results obtained with them. In section 8.1 the conforming parts of the algorithm are explained. Then, it comes the evaluation criteria at section8.2. It follows the description of the different tests done until reaching the selected algorithm, see section8.3. To sum up the results and conclusions are showed; see sections 8.4 and 8.5.

8.1 Introduction

The goal of the QP control algoritm is to keep the buffer occupancy next to a predefined level (usually half-full). The buffer can fluctuate around this desired value, as long as underflow, but most importanly overflow, are avoided.

Underflow can be tolerated, but should be avoided because it would force stuffing on the output bitstream, thus wasting precious transport bandwidth. On the other side, buffer overflow must be avoided because it would cause loss of data and errors in decoding process. If on one side the buffer fullness should be allowed to vary to absorb the video sequence complexity variance, on the other side it has to be tightly controlled to avoid overflowing, specially when some streep variations of image complexity are observed (e.g. scene change). As a consequence the final algorithm comprises two parts. One for the normal condition, and the other ready to react when an exception event occurs. Nevertheless, the normal condition algorithm will be designed to avoid reaching an exception situation.

Rate control strategy parts are described in the following:

Qp update algorithm This part is responsible of the update of the ‘Qp’ value under normal conditions, i.e. the buffer is not about to overflow or underflow.

Emergency algorithm Involves setting some rules to be applied when the buffer reach a situation where is highly probable to overflow or underflow. It has a higher priority than the normal condition algorithm.

8.2 Algorithm quality evaluation

The main ideas behind the algorithm's evaluation are to maximise visual perception quality, prevent any buffer unstabilities and finally minimize the QP fluctuations. One might think, at first glance, that these three ideas are really independent, but in fact, they are tied together facing a challenging compromise, which might be better illustrated by an example.

Let's suppose the QP jitter is controled manually so that it is minimized, i.e. the QP value will be kept as constant as possible, in addition the QP value is too low for the current coded frame; in this case the bitrate will start to grow, filling up the buffer, meanwhile the QP is kept unchanged as the jitter is to be reduced, so the buffer will fill up until it reaches a threshold when the QP value will be allowed to change.

Then the problem has been moved to the threshold value, if this threshold is low, the QP jitter will not remain minimized but if it is set to high, the difference between consecutive QP values will be to big, and that will afect the final visual perception as the quality differences in between a frame will be easily seen.

In addition, a high threshold value could lead to buffer oscillations as well, for instance, if the correction applied to the QP value is too high, which is a feasable option as the video sequence information varies in time, then the buffer will begin to be emtied very fast, and the same problem arises, having now the buffer occupancy oscillating, which by means of stability, is not a desirable situation.

Thus, as reveled by this example, trying to minimize the QP fluctuations means that the QP parameter is reacting slowly, so if the input sequence has a lot of information, that could lead to buffer instabilites. As a result, the research will focus on finding an algorithm that best balaces, QP jitter, buffer stability and visual perception quality.

The evaluation of the algorithm has been done as follows:

Magnitude	***Means of evaluation***
Quality	PSNR
QP jitter	Visual perception
Buffer unstabilities	Visual perception

PSNR magnitude gives an objective quality mesure of each frame of the video sequence. So, maximising the quality means maximising the average PSNR. It should also be controlled that the PSNR average variance, in normal conditions, is not too high in order to prevent the eye to see different quality levels. This can be achieved by minimising the QP jitter and variance as it has a direct relation to the final sequence quality.

On the other hand, a noticeable reduction on QP fluctuations can be quickly seen on the QP plots, so it has been considered a good approach to directly recall on the visual criteria for the evaluation of the QP fluctuations. The same can be said for the instabilities detection; to detect oscillations, in a non-linear system, we can analyse the plots of the buffer occupancy along the sequence. The oscillations that are due to image varying information, are amplified by an unstable rate control algorithm. In this case, the visual perception is good enough to decide if the algorithm presents some instabilities or not.

8.3 Tests

At the beginning, the first approach that was proposed was a non-liniar variation of a PID controller [16]. For general information about linear systems, see [17] and [18].

The implementation of the PID controller used in this project is, in its generic way, as follows:

$$qp\left[n+1\right] = qp\left[n\right] + a \cdot y^{p}\left[n\right] + b \cdot x\left[n\right] \tag{8.1}$$

where

$$\begin{aligned}
qp[n+1] &\rightarrow \text{QP value to be used to code next slice} \quad (\textsc{Algorithm Output}) \\
qp[n] &\rightarrow \text{previous QP value} \\
B_{ocup}[n] &\rightarrow \text{Buffer occupancy (bits)} \quad (\textsc{Algorithm Input}) \\
B_{size} &\rightarrow \text{Buffer size (bits)} \\
a, b, p &\rightarrow \text{design parameters} \\
n &\rightarrow \text{slice number} \\
y[n] &= \frac{B_{ocup}[n]}{B_{size}} \\
x[n] &= \frac{B_{ocup}[n] - B_{ocup}[n-1]}{B_{size}}
\end{aligned}$$

At the beginning some tests were performed to understand which of the design parameters was more influencing our final objectives: QP fluctuations, PNSR, stability.

In a second approximation, the tests focused on balancing the algorithm behaviour between: PSNR, QP fluctuations and stability.

The final behaviour of the global rate control algoritm depens on the parameters a, b, c, p and on the emergency condition algorithm. Therefore, the tests consisted on simulate an encoding-decoding process with different sets of parameters a, b, c, p and different emergency condition algorithms. All tests are done using the *multistream* approach and *half-slice intra refresh.*

In section 8.4 the most significant tests will be described. The description of the test consists on giving the rate control algorithm used along with the results obtained and the strong and weak points of the algoritm. The tests will be presented following a problem-solution structure until reaching the *best rate control algorithm* found.

8.3.1 Emergency condition

Two approaches can be followed; the simplest one is to divide the buffer into levels and define specific rules for each level. The second one, defines some levels but relies also on another variable to define the actuation to be carried out. This second approach, has the advantage of a better rejection to false alarms.

8.4 Results

For the tests the *mixed sequence* has been used, as it has many details that stress different aspects of the rate control algorithm. Thus, it is good to highlight the weak points of the strategy.

8.4.1 Former algorithm

Normal condition algorithm

$$qp[n+1] = qp[n] + 4y^3[n] + 2x[n]$$

Emergency condition

```
if(buff_occup > 70%) --> qp = 2*qp;
if(buff_occup > 80%) --> qp = 31;
if(buff_occup < 10%) --> qp = 0.75*qp;
```

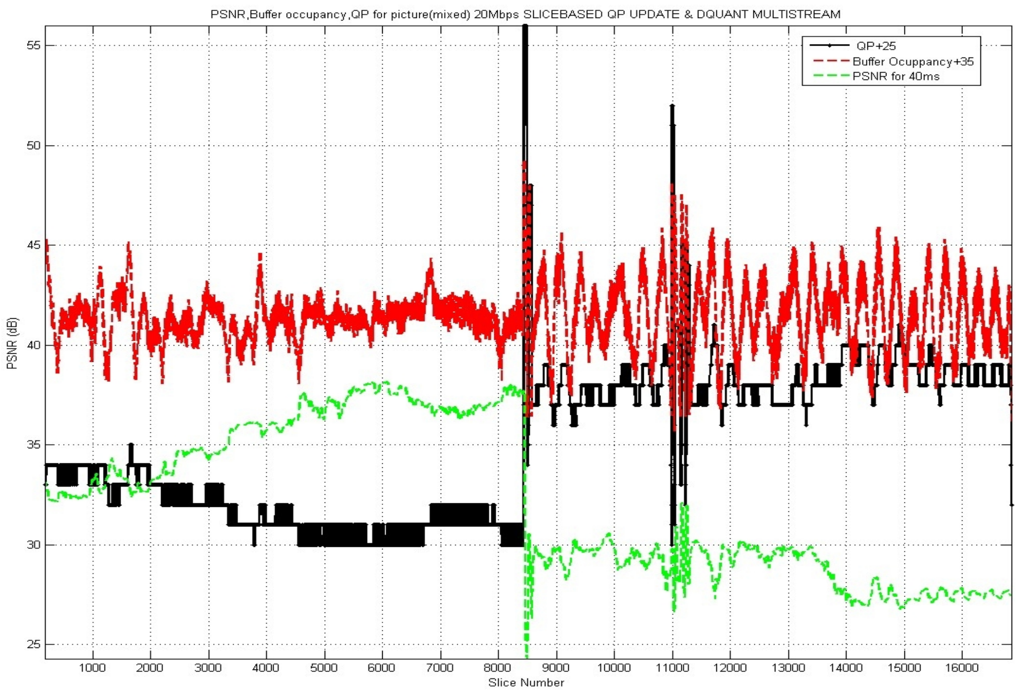


Figure 8.1: Mixed @ 20Mbps. *Former algorithm*

Problems

- Instabilities in parkrun sequence
- QP fluctuations can be reduced

Remarks

- none

8.4.2 High *'b'*

Normal condition algorithm

$$qp[n+1] = qp[n] + 4y^3[n] + 10x[n]$$

Emergency condition

```
if(buff_occup > 70%) --> qp = 2*qp;
if(buff_occup > 80%) --> qp = 31;
if(buff_occup < 10%) --> qp = 0.75*qp;
```

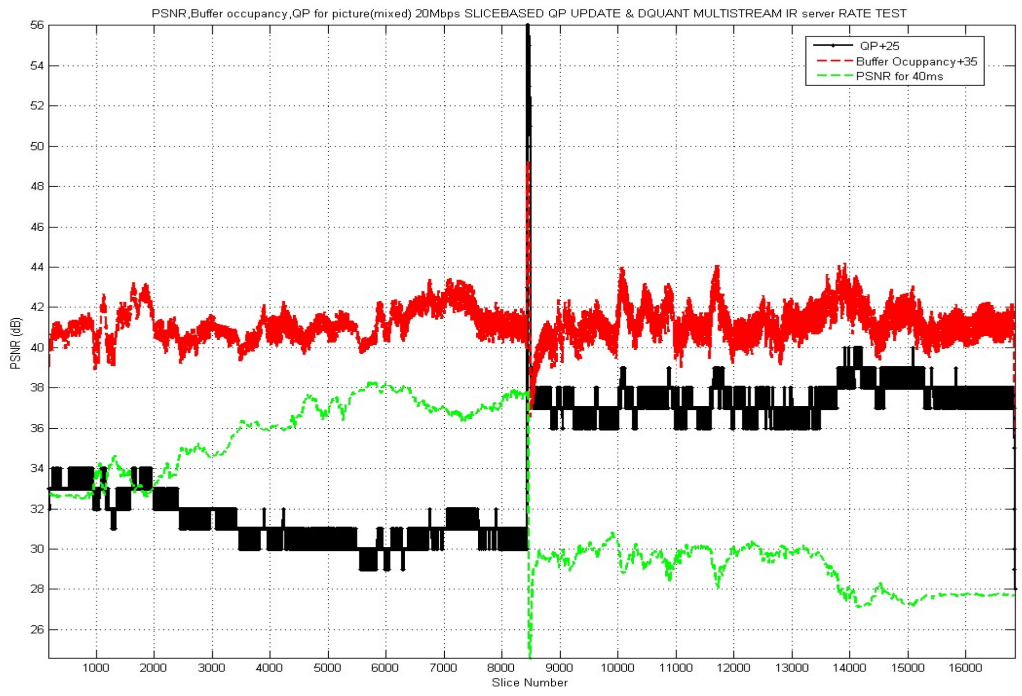


Figure 8.2: Mixed @ 20 Mbps. $(4y^3 + 10x)$

Problems

- Increased QP fluctuations

Remarks

- Stable behaviour

8.4.3 Average *‘x’*

Normal condition algorithm

$$qp[n+1] = qp[n] + 40 \cdot sign(y[n]) \cdot y^4[n] + 2x_{avg}[n]$$

$x_{avg} \rightarrow$ average of the last 4 *‘x’* values

Emergency condition

```
if(buff_occup > 70%) --> qp = 2*qp;
if(buff_occup > 80%) --> qp = 31;
if(buff_occup < 10%) --> qp = 0.75*qp;
```

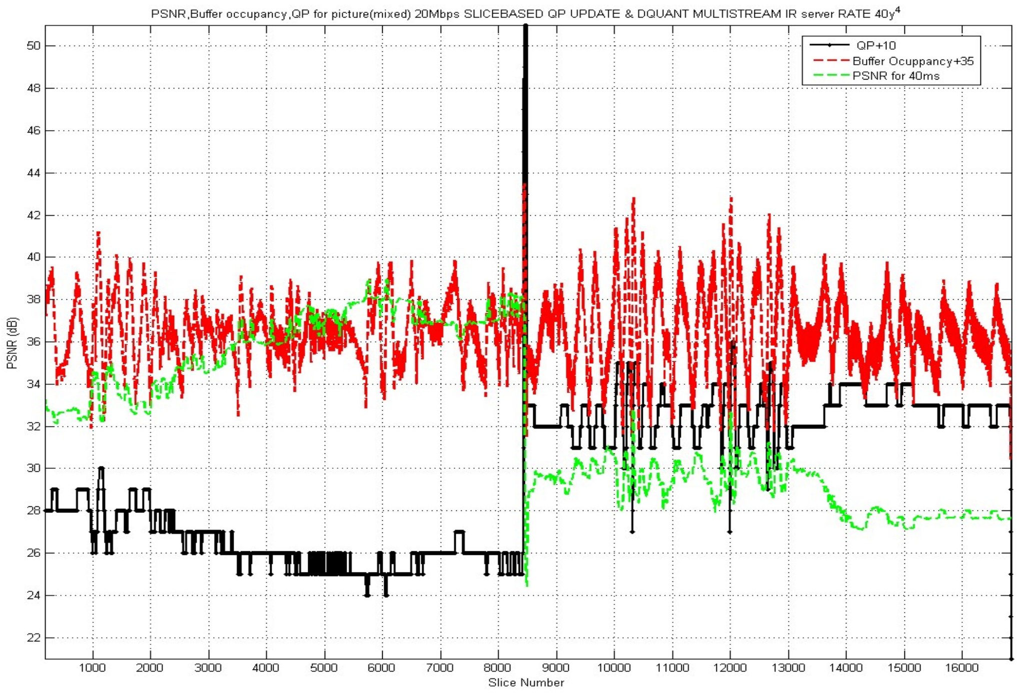


Figure 8.3: Mixed @ 20 Mbps. $(40y^3 + 2x_{avg})$

Problems

- Increased instabilities due to slow response

Remarks

- QP fluctuations also reduced

8.4.4 Modified *'p'* and *median*

Normal condition algorithm

$$qp[n+1] = qp[n]+5\cdot sign(y[n])\cdot y^2[n] + 8x_{med}[n]$$

$x_{med} \rightarrow$ median value of the last 3 'x' values

Emergency condition

```
if(buff_occup+x >= 85%) --> qp = 31;
if(buff_occup+x <= 10%) --> qp = 0.75*qp;
```

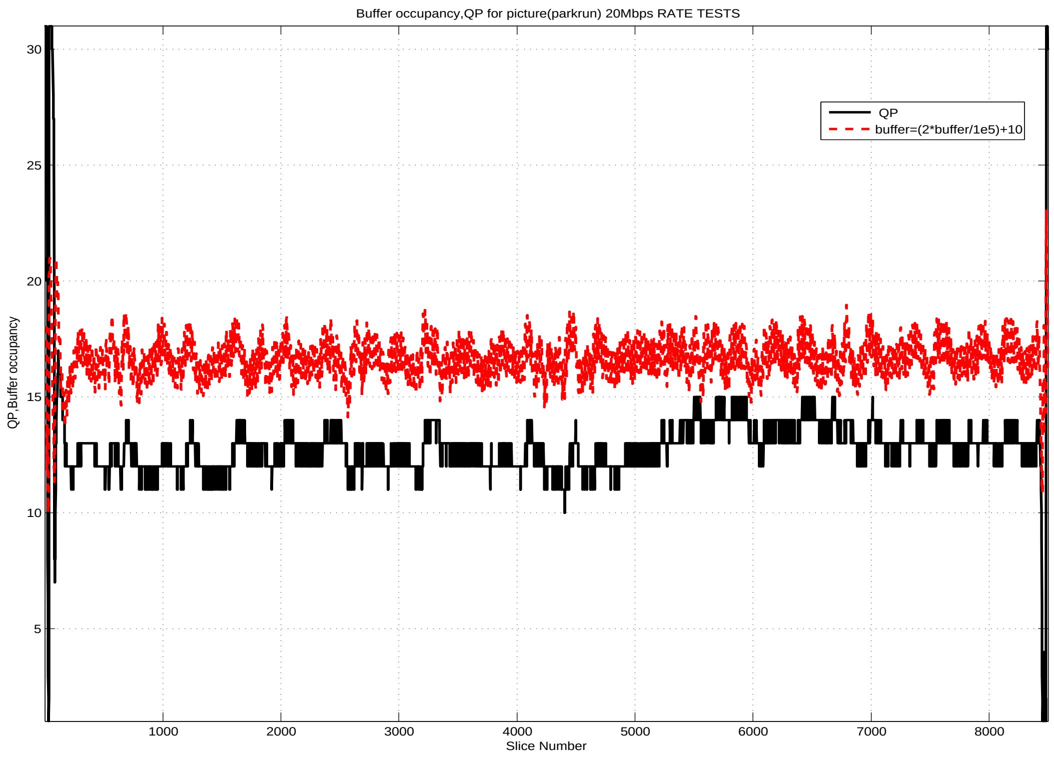


Figure 8.4: Parkrun @ 20 Mbps. $(5y^2 + 8x_{med})$

Problems

- Qp value reaching 15 too often
- Some undesired oscillations at the beginning

Remarks

- Qp jitter is somewhat reduced
- False alarms reduced by better emergency condition

8.4.5 FINAL algorithm

Normal condition algorithm

$$qp[n+1] = qp[n] + 0.5 \cdot sign(y[n]) \cdot y^2[n] \\ +8x_{med}[n]$$

$x_{med} \rightarrow$ median value of the last 3 'x' values

Emergency condition

```
if(buff_occup+x >= 85%) --> qp = 31;
if(buff_occup+x <= 10%) --> qp = 0.75*qp;
```

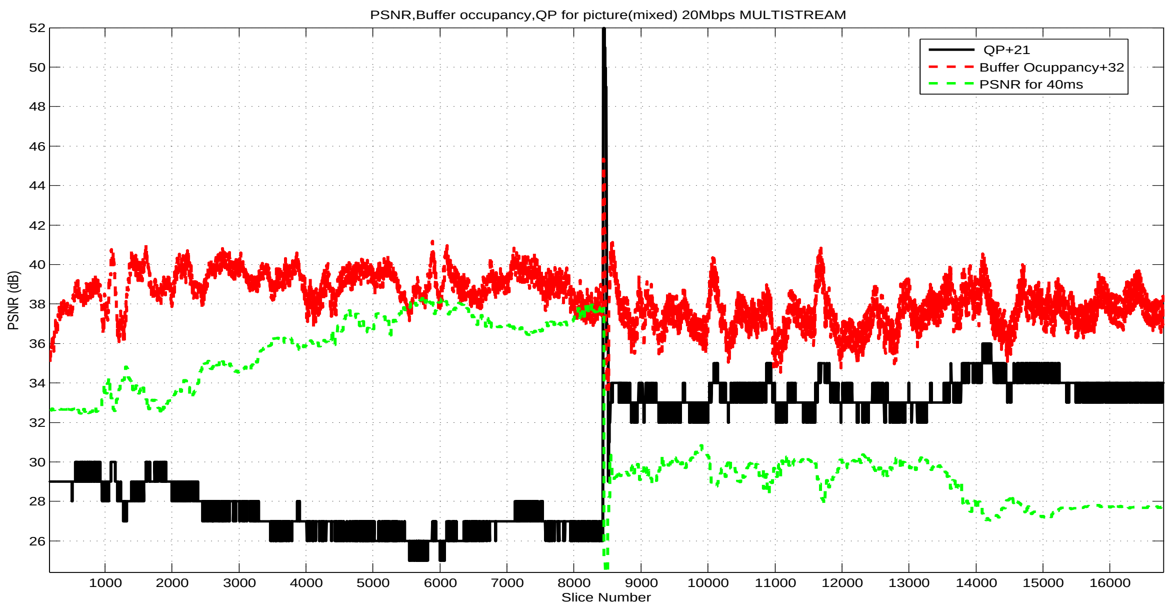


Figure 8.5: Mixed @ 20Mbps. *Final algorithm*

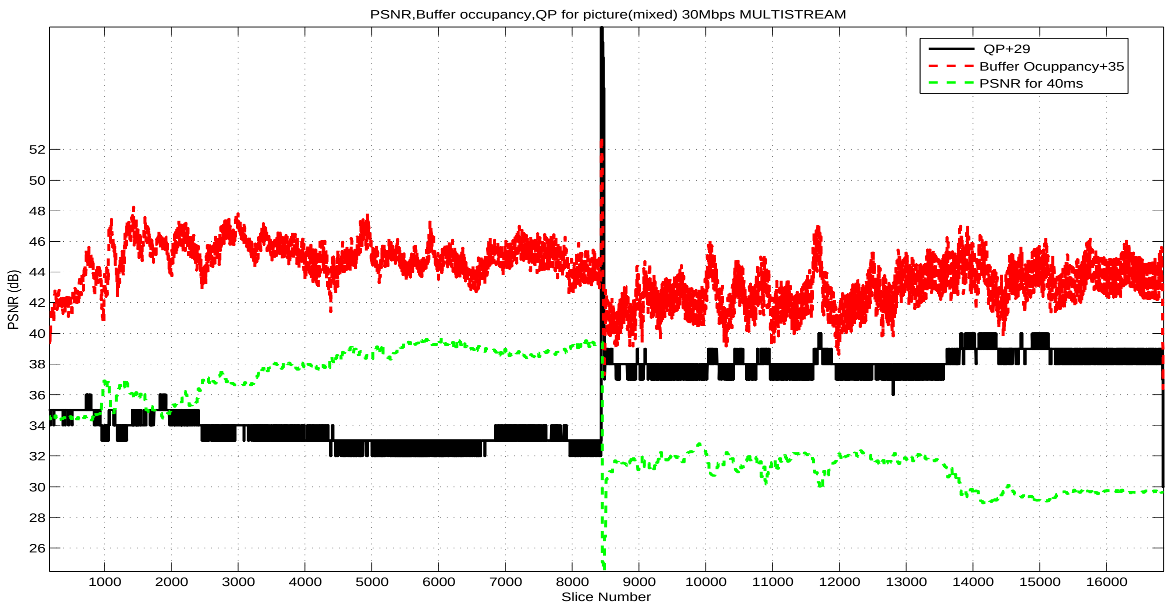


Figure 8.6: Mixed @ 30Mbps. *Final algorithm*

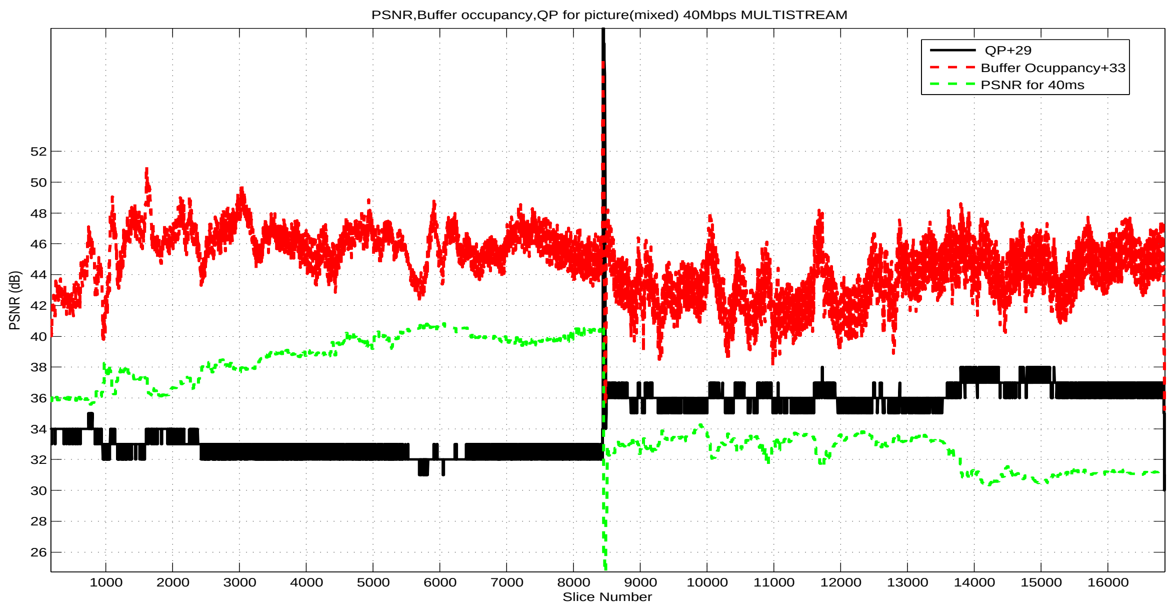


Figure 8.7: Mixed @ 40Mbps. *Final algorithm*

Problems

- none

Remarks

- QP fluctuations are mitigated by median filtering
- No oscillations ⇒ stable buffer
- False alarms reduced by better emergency algorithm
- Good performance on all bitrates

	20Mbps	30Mbps	40Mbps
Former algorithm	30.69410	32.68612	34.05209
Final algorithm	30.79729	32.77464	34.18261
Difference	+0.1	+0.1	+0.1

Table 8.1: Comparison between former and final algorithms. PSNR(dB)

For more graphs of the *final rate control algorithm*, please refer to section 7.3 where the results for different sequences have been given.

8.5 Conclusions

Firstly, one remarkable behaviour that has been found during the tests is that *QP fluctuation* and *buffer stablility* are two related magintudes. Whenever the *QP fluctuation* is reduced considerably, the *buffer occupancy* starts to oscillate; *reducing* buffer oscillations usually means increasing the *QP fluctuation.* Figures 8.4.1 and 8.4.2 shows perfectly the transition mentioned.

As a result, the QP fluctuation cannot be reduced as much as desired but up to the limit imposed by buffer stability. Consequently, eventhough figure 8.4.3 shows the algorithm which has the most reduced *QP fluct* in absolute terms, it is the final algorithm the one that it minimises it while keeping the buffer stable. Furthermore, it is worth noting that on the *final algorithm* the majority of the fluctuation is between one QP value, which it is acceptable as it is considered that not perceptual quality difference occurs between between consecutive QP values.

Figures 8.5, 8.6 and 8.7 in addition with the ones presented in section 7.3 reflect the robustness of the *final algorithm* as it perfoms well at all bitrates.

During the evolution of the tests it could be deducted the influence of the different terms of the equation (8.1). They are summarised in the following:

Equation part	Value	Consequence
y	high	buffer tightly constrained to target value but qp fluctuation increased
	low	easy to get into instabilities
x	high	very useful to stabilize the buffer. High impact on qp jitter increase
median	-	it allows to have high values on ‘x’ to keep buffer stable while reducing its impact on qp jitter

Moreover, table 8.1 shows that the stabilization of the buffer improves the average quality (PSNR). Besides the improvement in the average value, one important parameter to look at is the variance of the PSNR along time, which is desired to be as low as possible. So, by reducing the QP fluctuation we are decreasing PSNR variance.

Chapter 9

Conclusions

In this chapter, the conclusions are explained, as well as the future work that could be done.

9.1 Achievements

As a preliminary work of the project, intra-refresh and slice ‘qp’ update were implemented into the codec in order to adapt it to ultra-low delay. In addition, to make the codec capable of process HDTV sequences, multistream approach was implemented as well.

Moreover, the theoric penalty introduced by the multistream approach was tested. The results showed that multistream and singlestream comparison should be done in a sequence basis, as the results depend in great measure on the characteristics on the input sequence. Anyway, the differences between singlestream and multistream approaches are not remarkable, as they are much smaller than $0.5dB$.

The study of the rate control strategy showed a compromise between the buffer stability and the ‘qp’ fluctuation. Finally, a very robust algorithm, that stabilize the buffer and reduced the ‘qp’ fluctuations while miximasing the quality, was achived for 20, 30 and 40 Mbps.

9.2 Future work

At the end of the project, due to time restrictions, not all the desired tests, with the pre-filter enabled, could be performed, and the results obtained were not as good as expected. The video sequence got notably blurred while the blockiness was still perceivable at low bitrates in some sequences. Therefore, the filter remains useful to control emergency conditions, but further work should be done to reduce the artifacts without degrading too much image sharpiness.

The next step in the development of the project is to implement the codec, using the final rate control algorithm found in this project, into the target FPGA platform and verify final performance.

Bibliography

[1] M. Dupagne and P. B.Seel, *HDTV High-Definition TV*, ch. Global HD Production, p. 60. Iowa State University Press, first ed., 1998.

[2] I. E. Richardson, *H.264. and MPEG-4 video compression.* Wiley, 2003.

[3] A. V.Oppenheim, R. W.Schafer, and J. R.Buck, *Discrete-Time Signal Processing.* Prentice Hall, second ed., 1999.

[4] J. W. Woods, *Multidimensional Signal, Image, and Video Processing and Coding.* Academic Press, 2006.

[5] L.-I. Lundström, *Understanding Digital Television.* Elsevier, 2006.

[6] C. Poynton, *Digital Video and HDTV. Algorithms and Interfaces.* Morgan Kaufmann, first ed., 2003.

[7] http://www.itu.int.

[8] http://www.chiariglione.org/mpeg/index.htm.

[9] J. Chen and K. R. Liu, "Join source-channel multistream coding and optical network adapter design for video over ip," *IEEE Transaction on multimedia*, vol. 4, March 2000.

[10] P. Assuno and M.Ghanbari, "Buffer analysis and control in cbr video transcoding," *IEEE Transaction and systems for video technology*, vol. 10, February 2000. accepted.

[11] Y.-K. Hu and C.-B. Fong, "Mpeg-based buffer control for hdtv video coding," *IEEE*, 1993.

[12] ISO/IEC/JTC1SC29WG11, "Rate control and quantization optimization," April 1993. MPEG 2 TEST MODEL 5, Rev. 2, Section 10.

[13] Y. Sun, D. Li, I. Ahmad, and J. Luo, "A rate control algorithm for wireless video transmission using perceptual tuning," in *Proceedings of the International Conference on Informtion Technology: Coding and Computing (ITCC)*, 2005.

[14] M. Kunt *et al.*, *Techniques modernes de traitement numérique des signaux*, vol. 1. Presses polytechniques et universitairs romandes, première ed., 1991.

[15] M. Kunt *et al.*, *Traitement numérique des images*, vol. 2. Presses polytechniques et universitairs romandes, première ed., 1991.

[16] P. M.Frank, *Advances in Control.* Springer, 1999.

[17] B.P.Lathi, *Signal Processing & Linear Systems.* Oxford University Press, third ed., 1998.

[18] J. G.Proakis and D. G.Manolakis, *Digital signal processing.* Prentice Hall, third ed., 1996.

Appendix A

AC/DC intra prediction in slicewise qp update

MPEG-4 Visual Simple Profile explicity says that the QP value should not be changed inbetween one frame. Anyway, it is a required feature for the encoder of this project, as it has been seen in 5.4.1. Some modifications are required in the encoder to avoid decoding errors. The exact problem at the solution found is explained in the following sections.

A.1 Problem

It is worth recalling the prediction coding introduced in section 2.2.2 as MPEG-4 Visual standard, when it comes to the macroblocks which are to be intra-coded, it makes use of the prediction coding in order to reduce the amount of bits required to code a macroblock in intra mode. The prediction is called DC/AC intra prediction and is done in the entropy encoder, that means that the macroblocks are already DCT transformed and quantified. To illustrate what DC/AC intra prediction does an example is used. Let's suppose that the block[1] X_q of figure A.1 is about to be predicted. The prediction may come from its upper block C_q or its previous block A_q. The DC value of blocks A_q,B_q and C_q determine the method of DC prediction. If A_q,B_q or C_q are outside the VOP boundary, or if they are not intra-coded, their DC coefficient value is assumed to be equal to 1024 (which represents a mid-grey value). The direction of prediction is determined by:

```
if    $|DC_A - DC_B| < |DC_B - DC_C|$
      predict from block C
else
      predict from block A
```

The direction of the smallest DC gradient is chosen as the prediction direction for block X_q. The prediction P_{DC}, in the current implementation is formed by dividing the DC coefficient of the chosen neighbouring block by a scaling factor and P_{DC} is then substracted from the actual quantised DC coefficient Q_{DC_X} and the residual $P_{Q_{DC_X}}$ is coded and transmitted.

[1]Block refers to any group of 8x8 pixels

AC coefficients prediction is also done but will not be explained in this chapter as the proof of concept of the problem is the same for DC and AC coefficients. Further information can be found in [2].

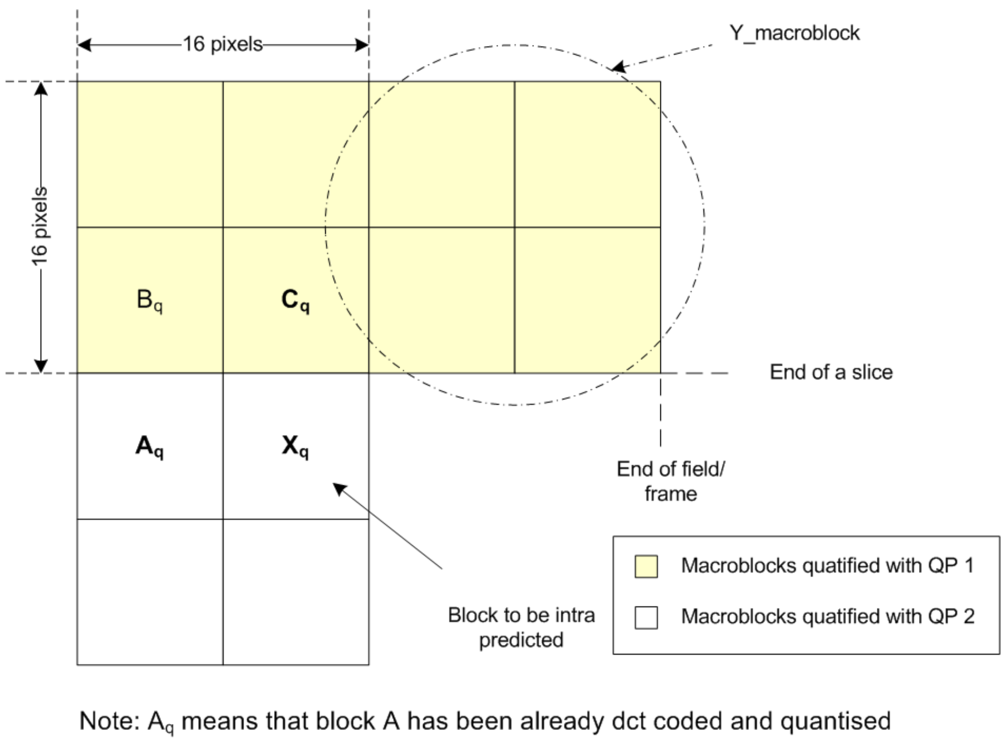

16 pixels
16 pixels
Y_macroblock
B$_q$
C$_q$
A$_q$
X$_q$
End of a slice
End of field/ frame
Block to be intra predicted
Macroblocks quatified with QP 1
Macroblocks quatified with QP 2
Note: A$_q$ means that block A has been already dct coded and quantised

Figure A.1: Intra prediction

At the decoder side, one should inverse the process by reconstruct the prediction P_{DC} as it knows the direction of prediction (it is transmitted within the bitstream), then decode $P_{Q_{DC_X}}$ and add it to P_{DC} in order to recover the original DC value Q_{DC_X}.

At this point all the relevant information for the real problem explanation is given. Let's point out what the actual problem was.

As to perform the predictions, DC coefficients values from one row of previous blocks must be stored. There is a difference between encoder and decoder on the way they store those coefficients. Encoder stores them once quantised whereas decoder stores them prior quantisation and it quantises them at the moment of the prediction.

Let's suppose now that the prediction of block X_q is from block C_q which is in a previous macroblock row, and that the previous macroblock row was quantified using the 'QP_C' QP parameter and the current row with 'QP_X' QP parameter, where $QP_C \neq QP_X$. In this scenario, the prediction will be done with a DC parameter from block C_q quantified with the QP parameter 'QP_C'. On the other hand, when block X_q will have to be decoded, DC coefficient will be copied from block C_q , it will be quantised with 'QP_X' and used for the prediction. It can be noted that this procedure ends with an *error* as the decoder is not using the same DC value for its prediction than encoder did.

The previous explanation it is also represented in figure A.2.

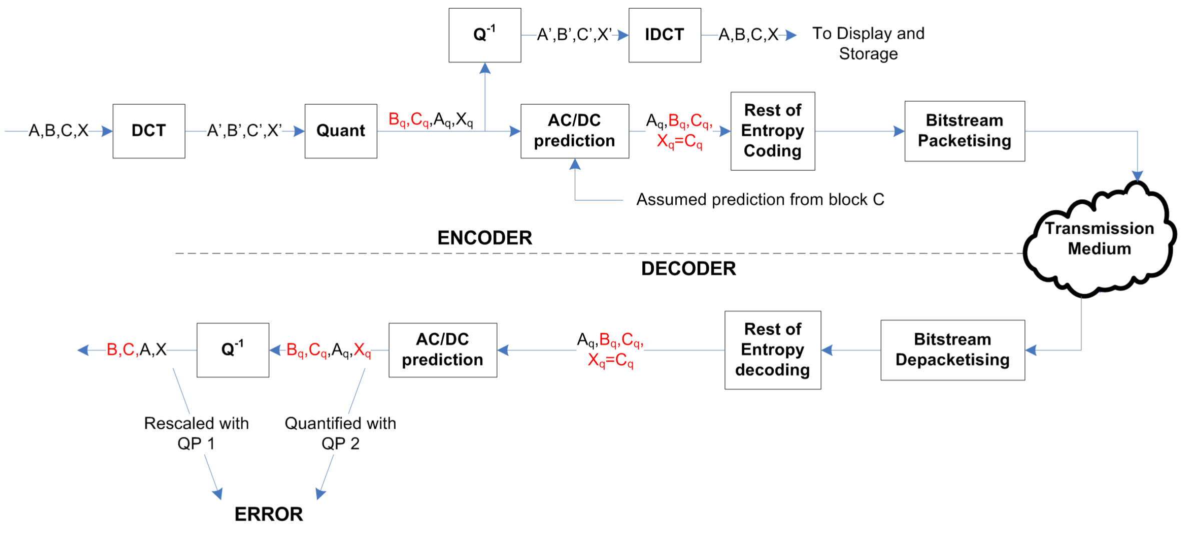


Figure A.2: Intra prediction error propagation flowchart

A.2 Solution

There are two solutions for the present problem:

1. Forcing the prediction from block A_q whenever there is a QP parameter change from previous slice. As block A_q is in the same slice, the QP parameter is the same.
2. Adapt both encoder and decoder to use the same QP parameter for quantisation of the coefficients used for prediction.

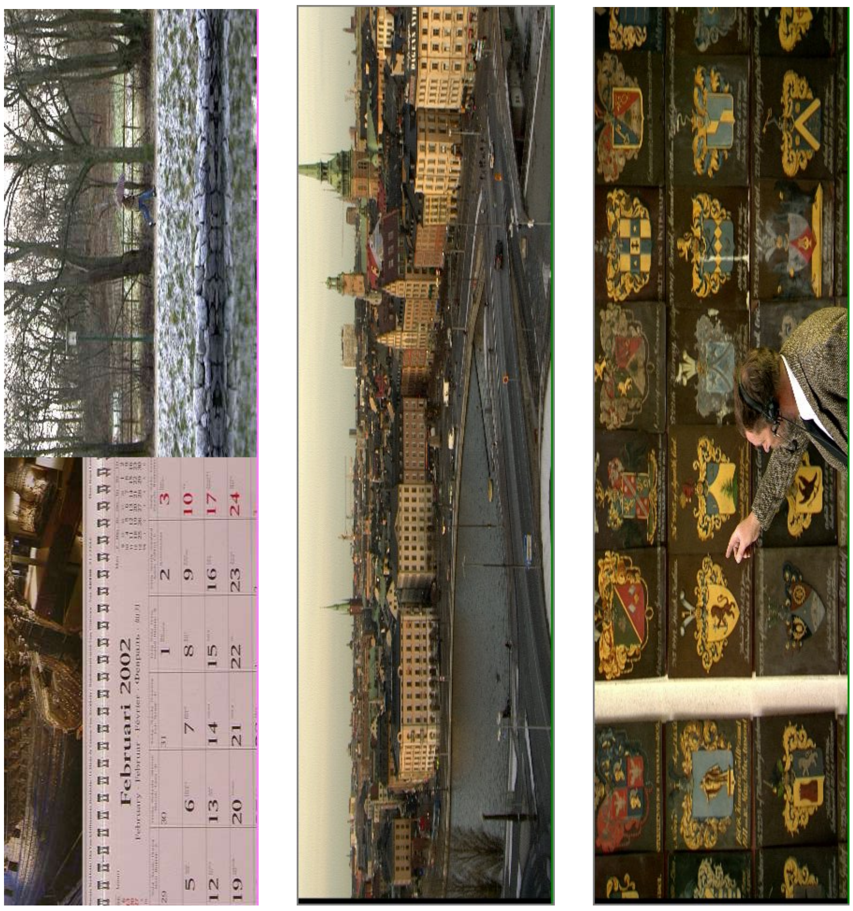
The solution implemented in the current project is the first one.

It requires little changes on the source even if this is not an optimal solution as it prevents optimal AC/DC predition. Anyway, this performance penalty is very limited since, only one slice per frame is susceptible of incurring on forced prediction, that is because of the intra-refresh is done slicewise. That becomes even less important if half slice intra-refresh is used as only half slice per frame is potentially affected.

Appendix B

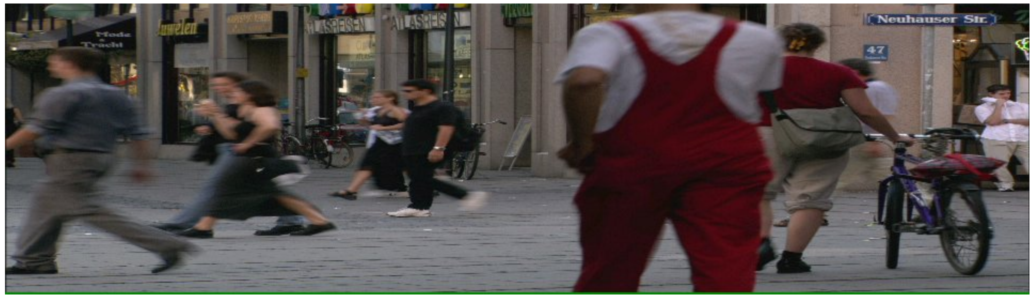
Video Sequences characteristics

This chapter describes the source, not compressed, video sequences used for testing the CODEC. It starts with a snapshot of each sequence and it ends with table B.1 which gives some details about the format and a short description of the sequence.

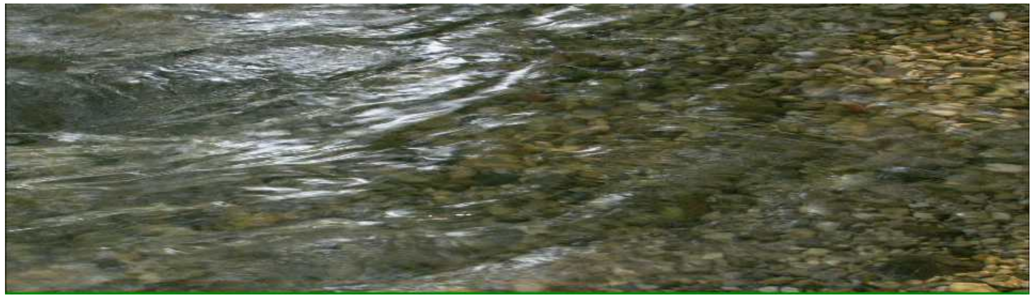


(a) Mixed (b) Stockholm (c) Shields

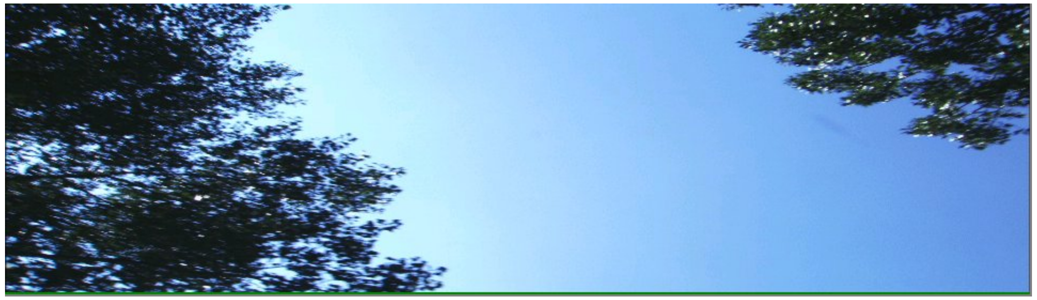
Figure B.1: Video sequences



(a) Pedestrian



(b) Riverbed



(c) Bluesky

Figure B.2: Video sequences

Sequence	**Resolution (pixels)**	**Scan mode**	**Frame rate (Hz)**	**Color space/sampling**[a]
Mixed Stockholm Shields Pedestrian Riverbed Bluesky	1920x1080	interlaced	25	YUV/420

[a]please see more info on digital YUV formats and standards on [2].

Sequence	**Num.fields**	**Short description**
Mixed	991	Created by appending parkrun to mobcal sequence. The sequences has been joined in a time basis, one after the other. Has a scene transition. It is *very versatile for tests.*
Mobcal	495	Moving calendar with text and a detailed photo of the Vasa ship. Moving train with colorful toys. Background with two types of wallpaper, one brown with details and one yellow with drawn figures. Very detailed but not very demanding.
Parkrun	496	Long shot. Man running in a park with an umbrella in his hand. Trees, snow and water in the background. Very detailed and demanding. It is the *most demanding* sequence of all.
Stockholm	503	Panning view over the Old Town of Stockholm. Detailed houses, water and moving cars.Panning view over the Old Town of Stockholm. Detailed houses, water and moving cars. Not very demanding
Shields	503	Man with beard and a speckled jacket walking in front of a wall of detailed knight shields. Zoom at the end of the sequence, preventiing motion compensation.
Pedestrian	374	Shot of a pedestrian area. Low camera position, people pass by very close to the camera. High depth of field. Static camera. Not very demanding
Riverbed	249	Riverbed seen through the water. hard to code but less critical than parkrun.
Bluesky	216	Top of two trees against blue sky. High contrast, small color differences in the sky, many details. Camera rotation.

Table B.1: Source video sequences. Characteristics

Appendix C

Testbench A. Results

C.1 Brief description

Table C.1 shows the main characteristics of the testbench A. Note that ‘former’ rate control algorithm refers to the algorithm that was implemented before the start of the project. Detailed information on this algorithm can be found in section 8.3. Equation (C.1) recalls the ‘former’ rate control algorithm.

$$qp[n+1] = qp[n] + 4y^3[n] + x[n] \tag{C.1}$$

Where $qp[n+1]$ is the next slice QP value, $qp[n]$ is the current QP value, $y[n]$ is the normalized buffer ocuppancy and $x[n]$ is the normalized difference between previous and current slice buffer ocuppancy.

Input sequences	Mixed, stockholm
Bit rates	20,30,40 (Mbps)
Intra-refresh	1 slice per frame. Top to botom. Starts on the first P-frame
Rate control algorithm	former

Table C.1: Testbench A. Description

C.2 Results

Figure C.1 shows that the performance of the multistream approach depends on the source video sequence, but one might say that it performes almost exactly due to its little differents for all the bitrates tested. However, this results come from an unstable rate control algorithm, and so, for the final conclusion it is required to perform other test, which have been presented in section 7.3.

	Singlestream			Multistream		
	20 Mbps	*30 Mbps*	*40 Mbps*	*20 Mbps*	*30 Mbps*	*40 Mbps*
Stockholm	35.34541	36.80622	37.68960	35.33475	36.81796	37.71483
Mixed	30.66213	32.68533	34.09775	30.69410	32.68612	34.05209

Table C.2: Average PSNR (dB)

	Difference		
	20 Mbps	*30 Mbps*	*40 Mbps*
Stockholm	0,01066	-0,01174	-0,02523
Mixed	-0.03197	-0,00079	0,04566

Table C.3: PSNR difference (dB). Difference=Single-Multi

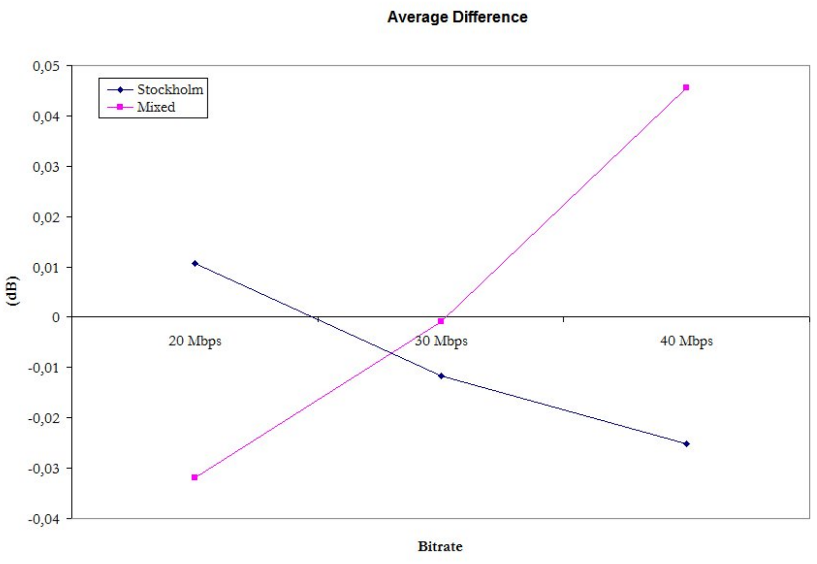


Figure C.1: PSNR average difference

C.2.1 Stockholm sequence

This sequence is easy to code because it has not many details or big motion, as it can be seen in figures from pag.63 to pag.65 with the highly flat QP response. It can also be noted that no instabilities were detected.

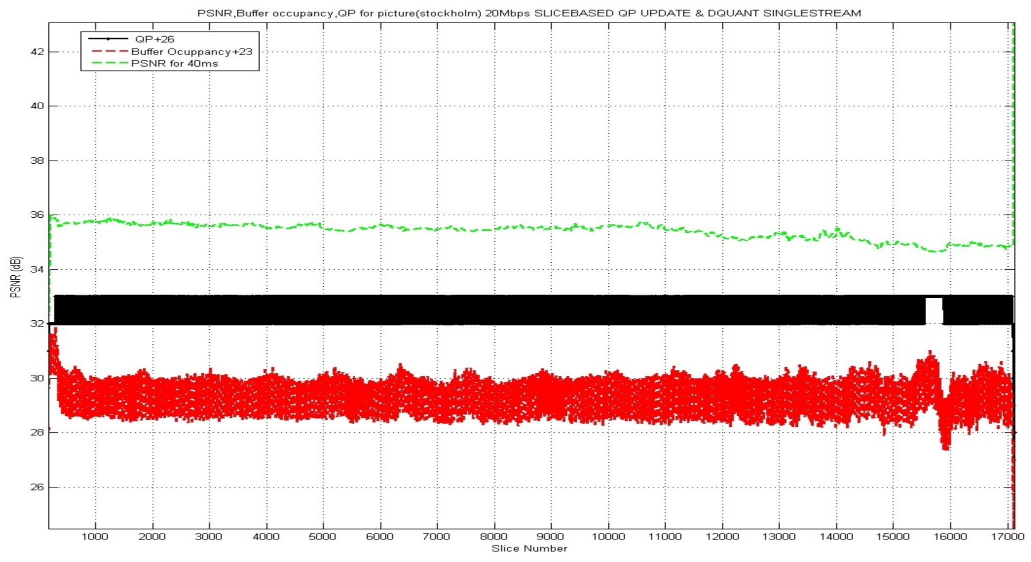


Figure C.2: Stockholm @ 20Mbps. SINGLESTREAM

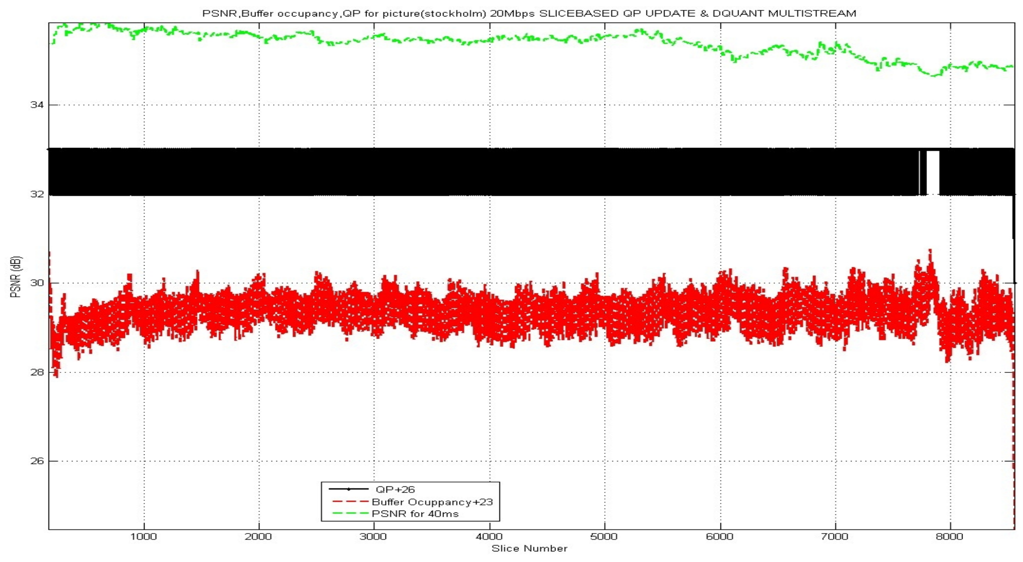


Figure C.3: Stockholm @ 20Mbps. MULTISTREAM

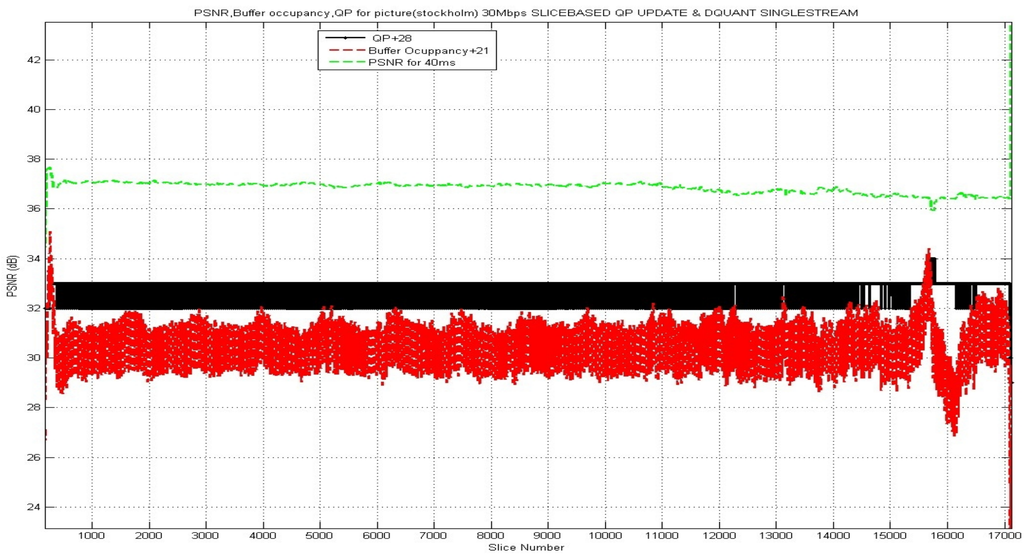


Figure C.4: Stockholm @ 30Mbps. SINGLESTREAM

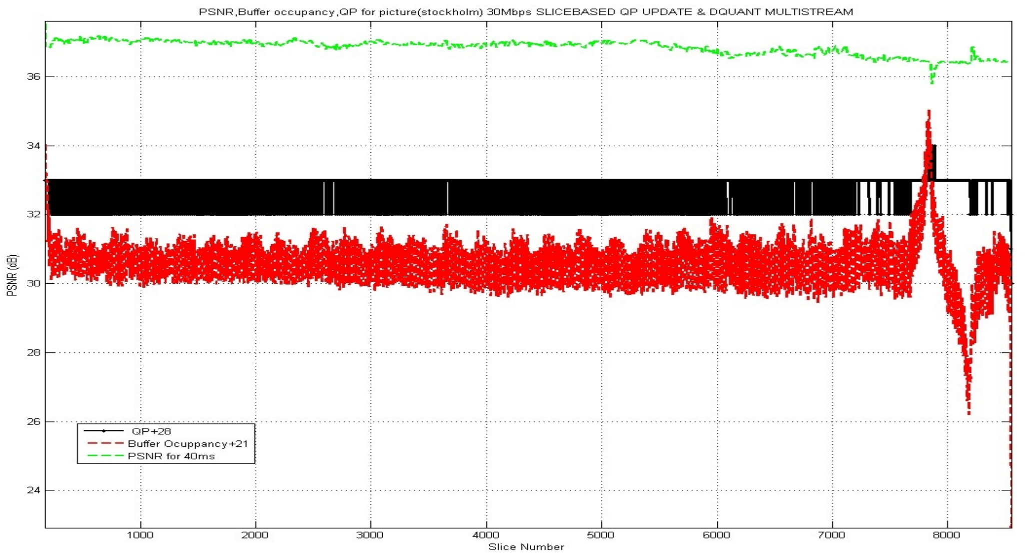


Figure C.5: Stockholm @ 30Mbps. MULTISTREAM

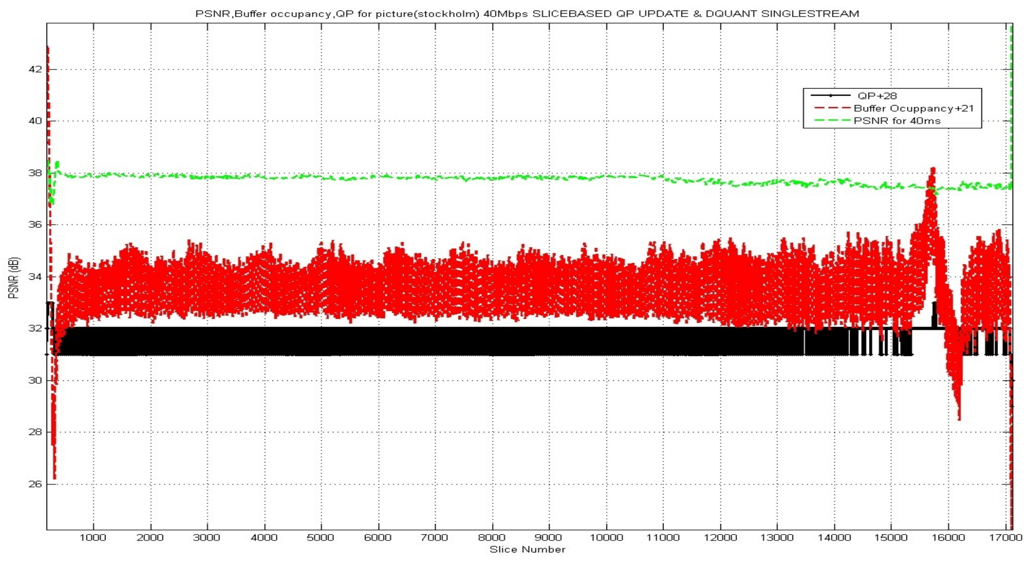


Figure C.6: Stockholm @ 40Mbps. SINGLESTREAM

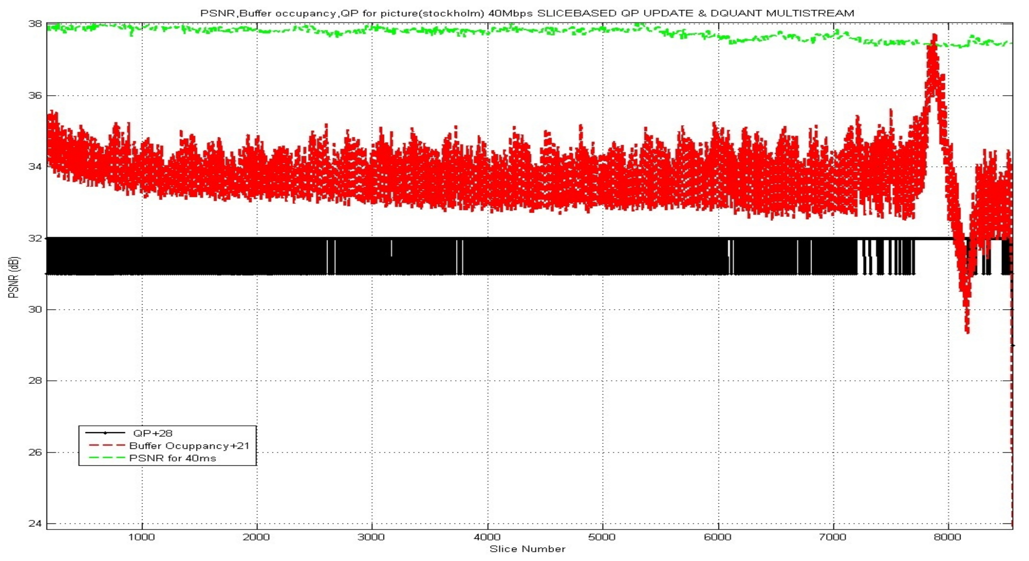


Figure C.7: Stockholm @ 40Mbps. MULTISTREAM

C.2.2 Mixed sequence

This video sequence has many issues that make it really difficult to code, e.g. scene change,parkrun sequence, and that is reflected with a high QP variance. In figureC.8, one can see the unstable oscillations of the QP value and the buffer occupancy.

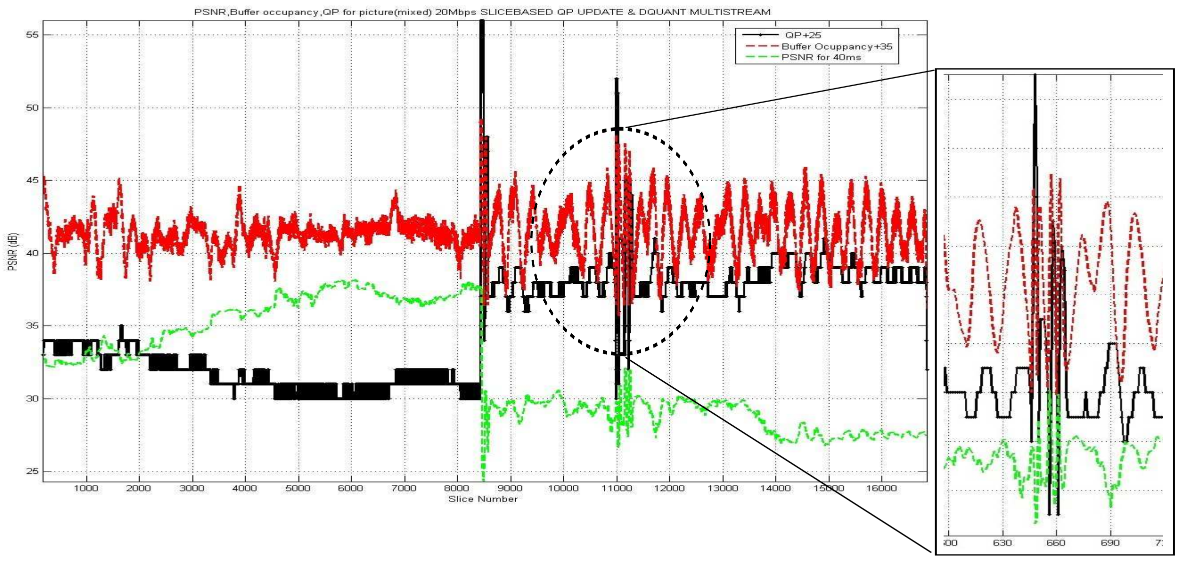


Figure C.8: Mixed @ 20Mbps. SINGLESTREAM

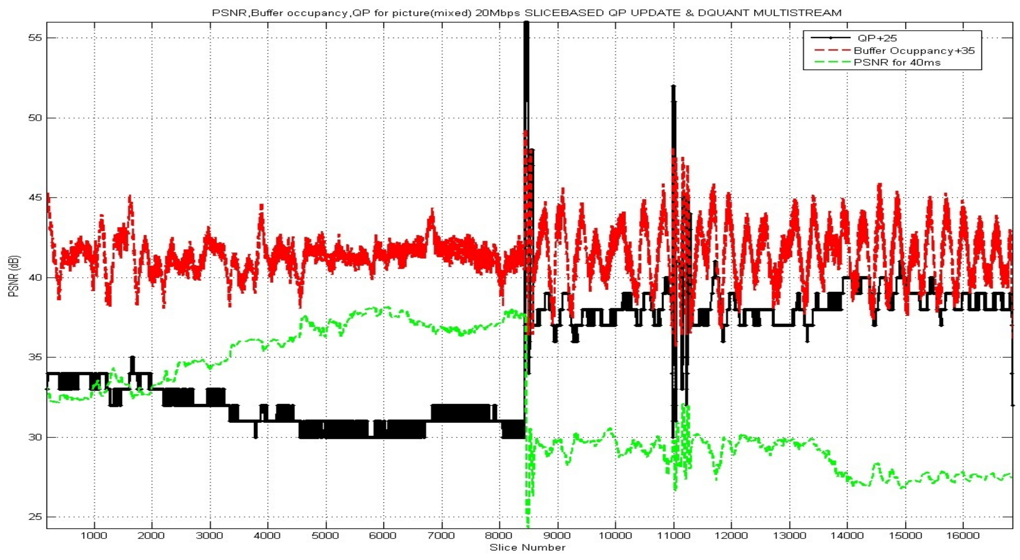


Figure C.9: Mixed @ 20Mbps. MULTISTREAM

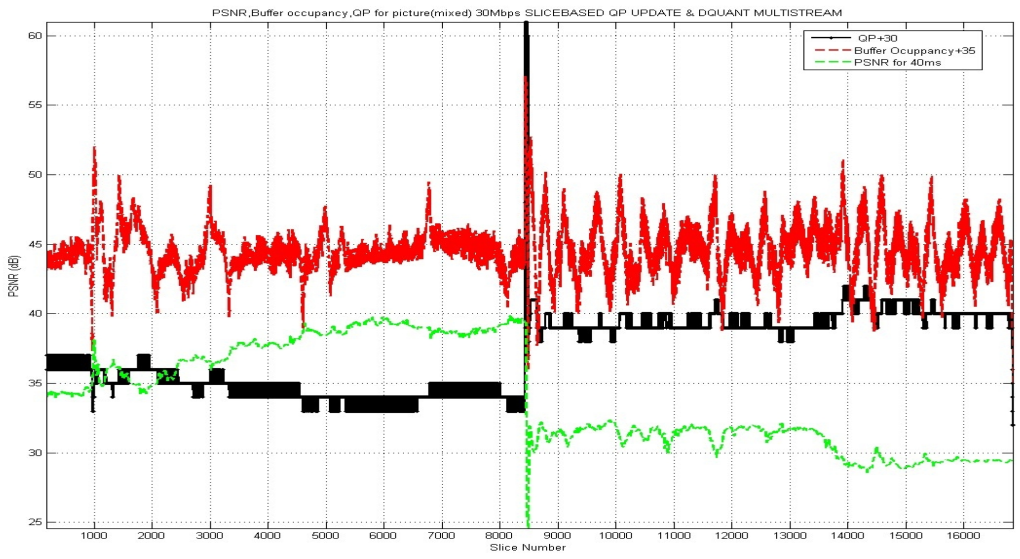


Figure C.10: Mixed @ 30Mbps. MULTISTREAM

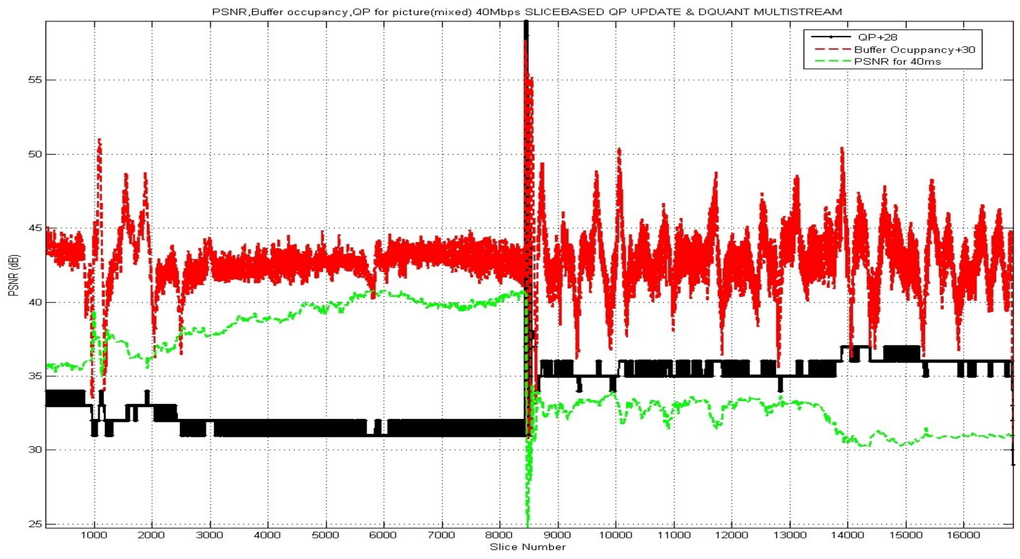


Figure C.11: Mixed @ 40Mbps. MULTISTREAM