A real-time SNR scalable transcoder for MPEG-2 video streams

by

C.J.J. Brouwers

Supervisors:

Dr. J.J. Lukkien
Eindhoven University of Technology
j.j.lukkien@tue.nl

Dr. P.D.V. van der Stok
Philips Research
peter.van.der.stok@philips.com

Eindhoven, June 2006
Abstract

Philips is one of the major companies with the aim of realizing the Connected Home vision. This vision embodies the idea of one’s home, where all kinds of devices are connected to each other via a heterogenous network to serve the purpose of watching and/or listening to digital content everywhere inside the home. As this heterogenous network may contain wireless links, problems may occur when transmitting high quality video using present wireless technologies since these suffer from both short- and long-term fluctuations in the effective bandwidth they provide. As a result, there exists a Quality of Service (QoS) trade-off between picture quality and required bandwidth.

One solution to tackle the QoS problem is by adapting the video to the currently available bandwidth before it is transmitted over the network. To achieve this, the video is converted to a scalable video format consisting of a base layer and up to two enhancement layers, each layer providing a different quality level. The conversion is performed by a real-time transcoder and the video is sent over the network by a network sender. The network sender is able to measure the currently available bandwidth by analyzing network traffic. Depending on these measurements, it sends a layering configuration for the scalable video to the transcoder. This layering configuration consists of the number of enhancement layers and the bit rates of each individual layer. This implies that the transcoder is capable of adjusting the number of layers and their bit rates at run-time.

The base layer can be created using existing transcoding architectures. Three different architectures are discussed; open-loop, closed-loop and pixel cascaded domain transcoder. From these, the closed-loop transcoder was chosen as a compromise between the picture quality provided by the cascaded pixel-domain transcoder and the performance of the open-loop transcoder. The closed-loop transcoder reuses the motion vectors obtained from the incoming video stream.

Since performance is a key aspect, a method to significantly increase the speed of the transcoder was investigated. This method performs motion compensation of the closed-loop transcoder directly in the frequency domain. This prevents the transformations between the pixel domain and the frequency domain. It is expected that by implementing this method, the performance of motion compensation for P-pictures can be increased with a factor of approximately two. For B-pictures, no significant increase in performance is expected.

Furthermore, methods were investigated to control the bit rates of the indi-
vidual layers. An important requirement of such a method was the ability to adapt the target bit rate at run-time. None of the investigated methods could satisfy this requirement. Therefore, a widely used bit rate control algorithm, TM-5, was adapted such that all requirements concerning the lowest layer were met. A bit rate control algorithm for the other layers could not be developed within the time budget of this project. It is expected however, that only small modifications to the bit rate control for the lowest layer are needed to make it suitable for the other layers as well.

The proposed architecture consists of the closed-loop transcoder to generate the lowest layer, with the addition of generating the remaining layers. It is expected that the performance of the proposed solution is insufficient to transcode video to scalable video in real-time on a modern PC. Therefore, optimizations are needed to achieve this. When this proves to be insufficient, the choice can be made to perform motion compensation on P-pictures only. It is expected that this increases the performance significantly at the cost of only minor quality decrease.
Preface

I am proud to present to you the major deliverable of the graduation project executed as part of the master course in Computer Science and Engineering provided by the Eindhoven University of Technology.

At first, I would like to thank my former employer, ICT Automatisering B.V. Eindhoven, for firing me back in November 2002. Due to the persistent recession in the IT industry, this company felt they had to fire its less experienced employees, unfortunately including me. Although I regretted this decision in the beginning, I changed my point of view and considered this as a perfect opportunity to continue my education.

In December 2002, I continued my education and enrolled at the university to follow the master course. At first, I was uncertain about whether I would be able to adapt to the higher level of education. These doubts were proven by the results achieved in the beginning. After a while, the results gradually improved, as I managed to adapt to the used notations and the higher level of abstraction. Eventually, due to hard work, dedication and persistence, I was allowed to start with this graduation project.

I had the profound wish to execute the graduation project externally at a large company with the intention of staying there if the collaboration satisfied both parties. Thanks to the effort of Johan Lukkien and Peter van der Stok, I was offered to execute a project at Philips Research. Since the assignment was very challenging and the company exceeded my expectations, I accepted this offer. I have enjoyed my stay at Philips Research and I am satisfied with the work I have produced. I hope my supervisors, my project leader and the judgement committee agree on this.

Finally, I would like to use this occasion to thank the people who supported me during this graduation project, in one way or another. In particular, I want to thank my supervisors Johan Lukkien and Peter van der Stok for their effort in guiding me through this project. Furthermore, I want to thank Jeffrey Kang, Dmitri Jarnikov and Matthias Krause for the pleasant collaboration with them concerning this project. On a personal basis, I would like to thank my girlfriend, family and friends who supported me during the past nine months.

Enjoy reading my master thesis!

Niels Brouwers.
# Contents

1 Introduction................................................. 13
   1.1 Purpose of this document ........................................ 13
   1.2 Context ................................................... 13
   1.3 Project goals .............................................. 14
   1.4 Structure of this document .................................... 16

2 MPEG-2 basics................................................. 17
   2.1 The MPEG-2 standard ......................................... 17
   2.2 Intra frame coding ........................................... 18
   2.3 Inter frame coding .......................................... 20
   2.4 Video stream semantics ..................................... 21
   2.5 Scalability ................................................ 23

3 Video transcoding............................................. 27
   3.1 Bit rate reduction techniques ................................ 27
   3.2 Transcoding architectures .................................. 29
      3.2.1 Cascaded pixel-domain transcoder ..................... 29
      3.2.2 Open-loop transcoder .................................. 29
      3.2.3 Closed-loop transcoder ................................ 32
   3.3 Evaluation ................................................ 33

4 Motion compensation in transcoding.......................... 35
   4.1 Motion compensation in the spatial domain .................. 36
   4.2 Motion compensation in the DCT domain ..................... 38
   4.3 Possible optimizations ..................................... 41
      4.3.1 Factorize shifting matrices ............................ 41
      4.3.2 Replacing fixed point operations with binary operations 43
      4.3.3 Exploit sparseness of blocks X ........................ 45
   4.4 Evaluation ................................................ 46

5 Bit rate control................................................. 49
   5.1 TM-5 bit rate control algorithm ............................ 49
   5.2 Adapting TM-5 to satisfy the required reaction time .......... 53
      5.2.1 TM-5 requires a GOP delay ............................ 53
      5.2.2 Calculation of the local spatial activity of a macroblock 53
      5.2.3 Changing the target bit rate at run-time ................ 54
   5.3 Bit rate control for the enhancement layers ................. 58
CONTENTS

6 Proposed architecture
  6.1 Real-time performance ........................................ 62
  6.2 Run-time adaptation of the target bit rate ................. 62
  6.3 Achievable picture quality .................................. 62
  6.4 Implementation ............................................. 63

7 Conclusions ............................................. 65

8 Recommendations and future work
  8.1 Enhancement layers ........................................ 67
  8.2 Motion compensation loop ................................. 67
  8.3 Improvement of picture quality .......................... 68
  8.4 Increasing performance ................................. 69

Abbreviations ........................................ 70

Appendix A ........................................ 72

Appendix B ........................................ 77

Bibliography ........................................ 80
List of Figures

1.1 The Connected Home vision . . . . . . . . . . . . . . . . . . . . . 14
1.2 Overview of the system . . . . . . . . . . . . . . . . . . . . . . 15
2.1 Discrete Cosine Transform applied on a 8x8 pixels block. . . . . 19
2.2 Quantization applied on a 8x8 block of DCT coefficients. . . . . 19
2.3 (a) Inter frame coding for P-pictures. (b) Inter frame coding for
B-pictures. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
2.4 Overview of a video sequence. . . . . . . . . . . . . . . . . . . . . 22
2.5 Generating base layer and enhancement layer in SNR scalability. 25
3.1 A basic video life cycle including transcoding in the context of the
Connected Home. . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.2 Different bit rate reduction techniques. . . . . . . . . . . . . . . 28
3.3 Cascaded pixel-domain transcoder. . . . . . . . . . . . . . . . . 29
3.4 Architecture of an open-loop transcoder. . . . . . . . . . . . . . . 30
3.5 Life cycle of intra and inter blocks when open-loop transcoding. . 31
3.6 Exaggerated picture to show the effect of drift. a) Video sequence
encoded at bit rate $R_1$ b) Video sequence at $R_1$ decoded and
re-encoded at bit rate $R_2$ c) Video sequence at bit rate $R_1$ open-
loop transcoded without drift correction at bit rate $R_2$. Note
that ($R_2 < R_1$). . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
3.7 The effect of drift on the quality of frames inside a GOP. . . . . 32
3.8 Architecture of a closed-loop transcoder. . . . . . . . . . . . . . . 33
4.1 Basic principle of motion compensation. . . . . . . . . . . . . . . 36
4.2 Example of a reference block $\hat{x}$ represented by the red square,
not aligned with 8x8 blocks of reference frame. . . . . . . . . . . 36
4.3 Architecture of a closed-loop transcoder with DCT-MCP. . . . . 38
5.1 An example of rate control for an I-picture. . . . . . . . . . . . . 51
5.2 Average value of complexity measures at four different bit rates. 56
5.3 Achieved reduction ratio per picture using the first method. . . . 57
5.4 Achieved reduction ratio per picture using the second method. . 58
5.5 Objective quality measurement of a video transcoded using both
methods of adapting TM-5. . . . . . . . . . . . . . . . . . . . . . 59
6.1 Proposed architecture of the scalable transcoder . . . . . . . . . 61
6.2 Temporarily transcoding to 2 MB/s demonstrating the reaction
time of one frame of the bit rate control algorithm. . . . . . . . . 63
8.1 Cascaded quantization (dashed arrows) compared with direct quantization with the coarser quantizer (solid arrow) (a) Cascaded quantization yields the same result as direct quantization. (b) Cascaded quantization yields a different result as direct quantization. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
8.2 Typical curve for bit rate reduction characteristics. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
## List of Tables

1.1 Important requirements of the transcoder .................................................. 16

2.1 Parts of the MPEG-2 standard ................................................................. 18

2.2 Video stream headers with corresponding start code ID ......................... 22

2.3 Entities inside a MPEG-2 video stream .................................................. 23

3.1 Comparison between the different architectures ...................................... 34

4.1 Number of calculations required for motion compensation in both the spatial and frequency domain ................................................................. 41

4.2 The result of matrix multiplication with matrices $c_{ij}$. The red square indicates the values subject to the shifting operation ...................... 42

4.3 Achieved optimization of the different methods of DCT-MCP compared with MCP in the spatial domain ......................................................... 46

8.1 Definitions, acronyms and abbreviations ................................................ 71
Chapter 1

Introduction

1.1 Purpose of this document

The final phase of the master course in Computer Science and Engineering as provided by Eindhoven University of Technology comprises a graduation project. This graduation project has been executed externally at Philips Research Eindhoven. It consisted of a research study and the development of software demonstrating the research results. This thesis is the most important deliverable of the project and describes the software system that was developed and the techniques investigated during the research phase.

1.2 Context

The graduation project, with the title "A real-time SNR scalable transcoder for MPEG-2 video streams", has been carried for the Information Processing Architectures (IPA) group, as part of Philips Research Eindhoven. The key capability of the IPA group is the architecture of computer-based consumer terminals. This capability consists of defining, architecting, and designing resource constrained systems. A large part of its work is dedicated to realizing Philips’ Connected Home vision. This vision embodies the idea of one’s home, where all kinds of devices are connected to each other via a heterogenous network \(^1\) to serve the purpose of watching and/or listening to digital content everywhere inside the home. By connecting this network to the internet, preferably through the use of a broadband connection, a user is able to obtain new digital content by streaming or downloading it from the internet. This vision is illustrated in figure 1.1.

As a result of the heterogenous network, the network may contain wireless links using present wireless network technology such as defined by the standards IEEE 802.11a, 802.11b and 802.11g \(\text{[1]}\). Streaming, for example, audio over these wireless links should not cause any problems since the required bandwidth for this can already be provided by present technologies. In theory, this holds for the

\(^1\) A network that consists of workstations, servers, network interface cards, operating systems, and applications from many vendors, all working together as a single unit. The network may also use different media and different protocols over different network links.
14

Chapter 1. Introduction

Figure 1.1: The Connected Home vision

Case of streaming high quality video over this network as well. In practice however, streaming video over a wireless link causes heavy degradation in perceived video quality by the perception of hick-ups, artifacts and blocking in the received video. This is a direct consequence of the following problems in present wireless network technology:

- The available bandwidth heavily fluctuates due to external interferences (e.g. microwaves, bluetooth devices).
- The wireless channel is not very reliable due to the frequent occurrence of transmission errors. This has a direct negative effect on the effective available bandwidth.
- The actual achieved bandwidth rapidly decreases when the distance between the nodes increases.

Apparently, there is a Quality of Service (QoS) trade-off between the available bandwidth provided by the wireless channel and the picture quality perceived by the viewer. It is the mission of the Keen Integration of Sub-Systems (KISS) project, which is executed in the IPA group, to provide solutions to manage this QoS problem. This graduation project is part of the KISS project.

1.3 Project goals

The global goal of this project is to tackle the QoS problem such that high quality MPEG-2 video can be streamed over a wireless channel. In this project, a solution for this problem is provided by converting high quality MPEG-2 video into scalable MPEG-2 video, before it is sent over the wireless channel. In the
1.3. PROJECT GOALS

Connected Home vision, this conversion is performed in real-time and takes place at a storage server[^2] where the video downloaded from the internet via the broadband connection is stored. This situation is further explained with figure 1.2.

![Figure 1.2: Overview of the system](image)

As shown in the figure, the storage server consists of a video transcoder and a network sender. By analyzing network traffic, the network sender continuously measures the available bandwidth that can be used to stream video over the wireless network. Depending on these measurements, it request the transcoder to deliver a scalable video stream that is adapted such that the network sender can correctly transmit the video over the wireless network. The scalable video stream can be adapted by changing the layering configuration at run-time. The layering configuration consists of the number of layers and the target bit rate of each individual layer.

The adaptation of the scalable video in terms of number of layers, provides a solution for the short-term fluctuation present in wireless networks. This is achieved by transmitting frames from a higher layer over the network only when sufficient bandwidth is available. The long-term fluctuations are being dealt with by the possibility to specify the bit rate of each layer. These forms of adaptation imply that the transcoder should be able to dynamically adjust the number of layers it produces and the bit rate of each individual layer, at run-time.

In this project, most attention is being drawn to research about and the development of the video transcoder. The network sender is out of the scope of this project. The most important requirements[^3] of the transcoder are stated in table 1.1. In this table, the layering configuration is expressed with requirement FR.1 and FR.2. Requirement FR.3 implies that the transcoder operates in real-time.

As such, performance of the transcoder is of high importance in this project. By means of requirement NFR.2, it is expressed that the transcoder should also achieve excellent picture quality.

The first goal of this project was to perform research on the subject of real-time video transcoding and to this specific video transcoding application in general.

[^2]: Ideally, the conversion takes place at the gateway, connecting the in-home network to the internet. However, due to the lack in performance of these gateways, this is not possible at present.

[^3]: The requirements for the transcoder are the result of previously executed research on this topic and were predetermined at the start of this project.
The number of video layers should be adjustable at runtime in the range [1..3].

The desired bit rate for each layer should be dynamically adjustable at runtime. The transcoder should respond within the time of a frame period to the adjusted bit rate.

The transcoder should transcode a frame of the input video to frames for the video layers within the time of $\frac{1}{f}$ second, where $f$ is the frame rate of the input video.

The transcoder should be able to communicate with a control program which defines the desired bit rate for each layer and the number of enhancement layers.

The transcoder should be able to connect to a network sender that will send the video layers over a network.

The transcoder should satisfy FR.3 on a standard PC (e.g. Intel Pentium 4 2.4 GHz CPU with 512 MB internal memory).

When the output stream consists of only one layer and has the same bit rate as the input stream, this stream should have a maximum decrease in quality of 1 dB in terms of Peak Signal to Noise Ratio.

<table>
<thead>
<tr>
<th>FR.1</th>
<th>The number of video layers should be adjustable at runtime in the range [1..3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR.2</td>
<td>The desired bit rate for each layer should be dynamically adjustable at runtime. The transcoder should respond within the time of a frame period to the adjusted bit rate.</td>
</tr>
<tr>
<td>FR.3</td>
<td>The transcoder should transcode a frame of the input video to frames for the video layers within the time of $\frac{1}{f}$ second, where $f$ is the frame rate of the input video.</td>
</tr>
<tr>
<td>FR.4</td>
<td>The transcoder should be able to communicate with a control program which defines the desired bit rate for each layer and the number of enhancement layers.</td>
</tr>
<tr>
<td>FR.5</td>
<td>The transcoder should be able to connect to a network sender that will send the video layers over a network.</td>
</tr>
<tr>
<td>NFR.1</td>
<td>The transcoder should satisfy FR.3 on a standard PC (e.g. Intel Pentium 4 2.4 GHz CPU with 512 MB internal memory).</td>
</tr>
<tr>
<td>NFR.2</td>
<td>When the output stream consists of only one layer and has the same bit rate as the input stream, this stream should have a maximum decrease in quality of 1 dB in terms of Peak Signal to Noise Ratio.</td>
</tr>
</tbody>
</table>

Table 1.1: Important requirements of the transcoder

Secondly, the gained knowledge was to be applied in practice by developing the transcoder such that all requirements are satisfied.

### 1.4 Structure of this document

The next chapter describes the basic techniques specified in the MPEG-2 standard, which are needed to understand the remaining chapters. Chapter 3 covers the video transcoding principles. In this chapter, the different transcoding architectures are discussed, from which one of them is selected to serve as the base for this project. In chapter 4, a method to increase performance of this architecture is described. A major challenge in the development of the transcoder is the problem of accurately control the bit rate of the video layers. Chapter 5 describes this problem and presents a solution in the case of real-time transcoding. The proposed architecture and its characteristics are described in chapter 6. This thesis is concluded with chapters 7 and 8, providing the conclusions and the recommendations, respectively.
Chapter 2

MPEG-2 basics

The first part of this chapter gives an overview about the MPEG-2 standard with a historical perspective. Secondly, the basic coding techniques present in the MPEG-2 video standard are described.

2.1 The MPEG-2 standard

Although video looks like continuous motion, it is actually a series of sequentially displayed still images changing with a sufficiently high frequency, providing the viewer with the perception of continuous motion. The frame rate of a video is defined as the number of displayed images per second. To achieve the illusion of continuous motion, the minimum frame rate is approximately ten frames per second.

Problems occur when analog video has to be digitally stored and/or transmitted, since the amount of digital information needed to represent an analog video is enormous. Therefore, several committees invented compression techniques to reduce the amount of digital information needed to represent the original video. The Motion Pictures Expert Group (MPEG), is the committee that developed both the MPEG standard, and its successor the MPEG-2 standard. The MPEG-1 standard is a standardization providing compression algorithms for both audio and video to reduce the size of a movie. The standard aims at storing a regular movie with a duration of two hours on a compact disc, such that the provided picture quality is approximately the same as can be achieved by popular analog video standards such as PAL, NTSC and SÉCAM. Although theoretically MPEG-1 can provide a significantly better picture quality, the standard was mainly developed for this purpose and was therefore limited in its use.

The development of MPEG-2 however was, due to the fact that the standard was defined by a lot of companies and stakeholders, aimed at a lot broader range of applications. These applications aimed not only at storing digital video, but also on transmitting digital video over networks. This resulted in a standard providing solutions for compressing video for a wide range of applications with various different requirements. Most of the compression techniques used in the MPEG-1 standard are also available in the MPEG-2 standard. This is shown
with the fact that typical MPEG-2 decoders are also capable of decoding MPEG-1 video streams.

The MPEG-2 standard consists of 9 specification documents, each covering a different aspect about video or audio compression. An overview of the specifications is given in table 2.1. As can be seen in table 2.1 part 8 of the MPEG-2 standard is missing. Work on this part was discontinued when it became apparent that there was insufficient interest from industry for it. In this project, only part 2: Video [5] was of interest. As a result, the most important aspects of this part needed for this project are described in the next sections. The other parts of the MPEG-2 standard are not needed for understanding the remainder of this thesis.

### 2.2 Intra frame coding

Intra frame coding exploits the spatial redundancy that exists between adjacent pixels of a picture, i.e. it is likely that neighboring pixels look similar. Pictures coded using intra frame coding only, are called I-pictures. In order to code an I-picture, a picture is first partitioned into blocks of 8x8 pixels. Then, each of these blocks are transformed to the frequency domain by applying a two-dimensional Discrete Cosine Transform (DCT). The two-dimensional DCT is defined according to the following equation:

\[
X(k, l) = c(k) c(l) \frac{1}{2} \sum_{n=0}^{7} \sum_{m=0}^{7} x(n, m) \cos(\frac{2n + 1}{16} \cdot k\pi) \cos(\frac{2m + 1}{16} \cdot l\pi) \tag{2.1}
\]

where \(c(0) = \frac{1}{\sqrt{2}}\), \(c(k) = 1\) for \(k > 0\), \(X(k, l)\) is a coefficient in the frequency domain and \(x(n, m)\) is a coefficient in the spatial domain.\(^2\) After this transformation, most of the energy of this block is concentrated in the low-frequency DCT coefficients. These low-frequency DCT coefficients store the global image

---

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO/IEC 13818-4:1998</td>
<td>Part 4: Conformance testing</td>
</tr>
<tr>
<td>ISO/IEC 13818-7:1997</td>
<td>Part 7: Advanced Audio Coding</td>
</tr>
<tr>
<td>ISO/IEC 13818-10:1999</td>
<td>Part 10: Conformance extensions for DSM-CC</td>
</tr>
</tbody>
</table>

Table 2.1: Parts of the MPEG-2 standard.

\(^1\)Part 8 of MPEG-2 was originally planned to be coding of video when input samples are 10 bits.

\(^2\)Domain in which an image is represented by intensities at given points in space. This is the most common representation for image data. The value of each pixel can be calculated according to the values of these intensities. Therefore, the spatial domain is commonly referred to as the pixel-domain.
of a block. In addition, the high frequency values enhance the level of detail of the image. The transformation is graphically depicted in figure 2.1.

![Discrete Cosine Transform](image1)

Figure 2.1: Discrete Cosine Transform applied on a 8x8 pixels block.

After transforming the 8x8 pixels blocks to the frequency domain, the DCT coefficients of each block are quantized to limit the number of possible values. This is achieved by dividing the coefficients with a quantizer step size. The quantizer step size is modified by two mechanisms; a quantization weighting matrix is used to modify the step size within a block and a quantizer scale factor is used such that the step size can be modified at the cost of only a few bits. This effect is visualized in figure 2.2. Another important characteristic of quantization is that the division of a low-value DCT coefficient by the quantizer step may result in the quantized DCT coefficient becoming zero.

![Quantization](image2)

Figure 2.2: Quantization applied on a 8x8 block of DCT coefficients.

The final step of coding an intra block is to apply a Variable Length Code (VLC) process to the quantized DCT coefficients to further reduce the number of bits. In the VLC process two different operations are applied on the DCT blocks; entropy coding and run-length coding. In entropy coding each DCT coefficient of a DCT block is replaced by a code with the length of only a few bits, representing the same value as the original quantized DCT coefficient. These replacement codes have different lengths and are assigned to a DCT coefficient similar as
in huffman coding; most-frequently occurring values are converted to shortest bit strings and least-frequently occurring values are converted to the longest. This can be compared with the assignment of letters to signals in Morse-code; the letter ‘A’ is represented with fewer signals than the letter ‘X’. The second operation applied in the VLC, run-length coding, prevents the coding of DCT coefficients having the value zero. This is achieved by counting the number of zero value DCT coefficients between two succeeding non-zero value DCT coefficients. As a result, only the number of succeeding zeros is coded which result in fewer bits than when the succeeding zeros would have been coded.

Note that the DCT and VLC operations in intra coding are loss-less. This means that applying the inverse DCT and inverse VLC operations on the coded video results on a perfect reconstruction of the original video. In contrast, the quantization process is a lossy operation, i.e. quantization prevents the possibility of a perfect reconstruction of original values. Therefore, the achieved picture quality of intra coding highly depends on the degree of quantization. In general, performing a coarser quantization leads to usually high frequency DCT coefficient values becoming zero, and thus more loss in detail.

2.3 Inter frame coding

Many of the requirements of MPEG-2 video can be achieved using only intra frame coding techniques. However, the image quality that can be achieved using these techniques at low bit rates is usually not sufficient. Therefore, MPEG-2 contains inter frame coding to further increase coding efficiency. Inter frame coding exploits the temporal redundancy that exists between adjacent frames in a video sequence. This is achieved by computing a difference signal called the prediction error. In MPEG-2, predictions are formed on a macroblock basis, where a macroblock is defined as a block of 16x16 pixels.

In unidirectional prediction, also called forward prediction, a target macroblock in the picture to be encoded is matched with a set of displaced macroblocks of the same size in a past picture called the reference picture. The macroblock in the reference frame that best matches the target macroblock is used as the prediction macroblock. The prediction error is then computed by subtracting the target macroblock with the prediction macroblock. A motion vector describing the horizontal and vertical displacement from the target macroblock to the prediction macroblock indicates the position of this best matching prediction macroblock. The prediction error signal is compressed using intra frame coding and stored in the video stream. Pictures coded using forward prediction are called Predicted Pictures, or P-pictures for short. This is depicted in figure 2.3a. For a P-picture, both I-pictures and P-pictures can serve as a reference picture.

Besides forward prediction, a key feature of MPEG is the presence of bi-directional temporal prediction, also called Motion-Compensated Interpolation. Pictures

\footnote{P- and B-pictures may also contain intra coded macroblocks. If so, it was not more efficient to predict these macroblocks.}
2.4. VIDEO STREAM SEMANTICS

coded with bi-directional prediction, called B-pictures, differ from forward predicted picture in the sense that they may use two reference pictures for making the prediction. A target macroblock in bi-directional coded pictures can be predicted by a prediction macroblock from the past reference picture (forward prediction), or one from the future reference picture (backward prediction), or by two prediction macroblocks, one from each reference picture (interpolation). In the case of interpolation, the best matching macroblocks out of both pictures are averaged to create a more accurate prediction.

B-pictures can use both I-pictures or P-pictures as one of the reference pictures. In every case, a prediction macroblock from a reference picture is associated with a motion vector, so that up to two motion vectors per macroblock may be used with bi-directional prediction. Inter frame coding for B-pictures is graphically depicted in figure 2.3b. In general, using bi-directional predication in a video stream usually results in a higher compression ratio than could be achieved with using I and P-pictures only.

2.4 Video stream semantics

The video stream is organized in a number of headers and the actual video data. The start of each header is indicated by a corresponding startcode. Table 2.2 displays the headers with their corresponding startcodes. Each start code ID is preceded by a fixed prefix code, which is a string of at least 23 zeros followed by a one.

A layer consists of a number of objects. The most important entities present in a MPEG-2 video stream are shown in table 2.3.

Figure 2.4 shows a graphical representation of a MPEG-2 video sequence. Although this figure is simplified, it hopefully provides a good understanding of how a MPEG-2 video is stored into a stream.
<table>
<thead>
<tr>
<th>Start Coding Type</th>
<th>Start code ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>picture_start_code</td>
<td>00</td>
</tr>
<tr>
<td>slice_start_code</td>
<td>01 to AF</td>
</tr>
<tr>
<td>user_data_start_code</td>
<td>B2</td>
</tr>
<tr>
<td>sequence_header_code</td>
<td>B3</td>
</tr>
<tr>
<td>sequence_error_code</td>
<td>B4</td>
</tr>
<tr>
<td>extension_start_code</td>
<td>B5</td>
</tr>
<tr>
<td>sequence_end_code</td>
<td>B7</td>
</tr>
<tr>
<td>group_start_code</td>
<td>B8</td>
</tr>
<tr>
<td>reserved</td>
<td>B0, B1, B6</td>
</tr>
</tbody>
</table>

Table 2.2: Video stream headers with corresponding start code ID.

In this figure the smallest entity is a block, which has a size of 8x8 pixels. The decomposition shown in figure 2.4 typically holds for an I-picture. I-pictures consist entirely of intra coded macroblocks. As described in section 2.3, P- and B-pictures consist out of mostly inter coded macroblocks, represented by one or two motion vectors and a number of residual blocks per macroblock. These inter coded macroblocks are stored in the video stream similar as intra coded macroblocks. The representation of macroblocks as being full color images is also simplified. In reality macroblocks consist of several luminance and chrominance blocks\(^4\) that together form a full color image.

\(^4\)The actual number of luminance and chrominance blocks depends on the chroma format used. Chroma format 4:2:0, defines that a macroblock consists of 4 luminance blocks and 2 chrominance blocks. Chroma format 4:2:2 and 4:4:4 defines that a macroblock consist of 4 and 8 chrominance blocks, respectively.
2.5. **SCALABILITY**

<table>
<thead>
<tr>
<th>Block</th>
<th>A block of 8x8 neighboring pixels. Intra frame coding techniques are applied on these blocks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macroblock</td>
<td>A macroblock contains a number of blocks depending on the chroma format that is used. For example, when the 4:2:0 chroma format is used, a macroblock consists of 4 luminance blocks and 2 chrominance blocks. Inter frame coding techniques are operations executed on a macroblock scale.</td>
</tr>
<tr>
<td>Slice</td>
<td>A slice consists of an arbitrary number of macroblocks. Slices offer a mechanism for synchronization and thus limit the propagation error.</td>
</tr>
<tr>
<td>Picture</td>
<td>A picture consists of an arbitrary number of slices.</td>
</tr>
<tr>
<td>Group of Pictures</td>
<td>Group of pictures is, as the name already suggests, a collection of pictures. A group of pictures generally starts with an I-picture. Between two succeeding I-pictures, an arbitrary number of P and B-pictures may occur in any order. The group of pictures mechanism provide a mechanism for achieving requirements like fast-forward, fast-reverse and instant play.</td>
</tr>
<tr>
<td>Sequence</td>
<td>A sequence consists of an arbitrary number of GOPs.</td>
</tr>
</tbody>
</table>

Table 2.3: Entities inside a MPEG-2 video stream.

### 2.5 Scalability

The presence of scalability techniques in MPEG-2 is a major difference between the MPEG-1 and MPEG-2 standards. Scalable video coding involves generating a coded representation in a manner that facilitates the derivation of video of more than one resolution or quality level from a single bitstream. Each coded representation is stored in a individual layer. The lowest layer is referred to as the **base layer**. Other layers are called **enhancement layers**. MPEG-2 specifies four different modes of scalability, namely:

- **SNR Scalability** involves coding in a manner to generate separate bitstreams representing individual layers such that an enhancement layer carries DCT refinement coefficients which are intended to enhance the DCT coefficients included in the lower layer. Using DCT coefficients from both layers obviously results in a better picture quality. Since the enhancement layer is said to enhance the signal-to-noise ratio (SNR) of the lower layer, this type is called SNR scalability.

- **Spatial Scalability** refers to layered coding in a spatial domain which means that in this type of coding scheme, a video can be provided with two different spatial resolutions in one single video stream. The lower layer provides video with a lower spatial resolution, which can be enhanced by also decoding the enhancement layer. This type of scalability can

---

5 Usually, the DCT refinement coefficients are the quantization error of the previous layer. This quantization error is calculated by the difference between the input signal before quantization, and output signal after quantization.
be used to provide backward compatibility with other standards. For example, spatial scalability can be used to transmit a SDTV and HDTV signal in one bitstream. If such a bitstream is received by a Standard Definition television (SDTV) it will only decode the standard resolution signal. However, when a High Definition television (HDTV) receives the bitstream it will use the high resolution signal as well.

- **Temporal Scalability** refers to layered coding that produces two or more layers, each with either the same or different temporal resolutions (i.e. frame rate), which when combined provide full temporal resolution as available in the input video. The spatial resolution of frames in each layer is identical to that of the input video.

- **Data Partitioning**, although not a true form of scalability, does provide a means of partitioned coded video data into two priority classes: essential data and additional data. In data partitioning the distinction between essential data and additional data is not very strict since both types of data are simply derived by partitioning a single stream of coded MPEG-2 video data, depending on the amount of data desired for each data type.

The type of scalability used in this project is an adapted version of SNR scalability [4]. In this adapted form of SNR scalability more than one enhancement layer is allowed. Experiments carried out that coding video by using more than two enhancement layers did not provide significant advantages. As such, the transcoder in this project should be able to produce a maximum of two enhancement layers. Furthermore, the enhancement layers coded by the adapted version have the following characteristics:

- Frames of an enhancement layer have no reference to other frames from the same enhancement layer.

- Frames of an enhancement layer have no reference to frames from previous layers.

- The first frame in an enhancement layer is an I-picture. The last frame is a P-picture. All the pictures in between are B-pictures.

These characteristics enable decoding an enhancement layer independently of other layers, using a legacy MPEG-2 decoder. Furthermore, it allows for the dropping of frames from an enhancement layer arbitrarily, without resulting in artefacts in the perceived video. This makes it possible to adapt to fluctuations in the available bandwidth of the wireless network. When received, the decoded frames from the enhancement layers are added to the base layer to enhance the quality.

The base layer of this scalable video stream is generated by requantizing the DCT coefficients such that the target bit rate for the base layer is satisfied. The requantization errors of a previous layer are stored in the enhancement layers. An example of this is shown in figure 2.5.
Figure 2.5: Generating base layer and enhancement layer in SNR scalability.
Chapter 3

Video transcoding

Video transcoding is the operation of converting a video from one format into another format. A format is defined by characteristics such as the bit rate, frame rate, spatial resolution and coding syntax. In this project, a MPEG-2 video stream is transcoded to a SNR scalable MPEG-2 video stream. In this chapter, it is explained which type of bit rate reduction is achieved to create the layers of the scalable video stream. Furthermore, it contains a section containing a discussion about the different video transcoding architectures to apply this bit rate reduction technique.

3.1 Bit rate reduction techniques

In this project, the goal of transcoding is to convert an MPEG-2 video stream to a SNR scalable MPEG-2 stream, such that the individual layers of the scalable video stream have a lower bit rate than the input stream. In this section, techniques are mentioned to achieve this for the base layer. Figure 3.1 shows the entire life cycle of a video stream to illustrate the problem of bit rate reduction. In this figure, the encoder compresses an input video at a bit rate $R_1$, then the transcoder converts this compressed video at a bit rate $R_2$, which is smaller than $R_1$. Next, the decoder decodes the transcoded video bit stream for display. Popular methods that can be applied to achieve bit rate reduction on an incoming MPEG-2 video stream are:

1. Dropping of frames
2. Reducing the spatial resolution
3. Discarding high-frequency DCT coefficients

Figure 3.1: A basic video life cycle including transcoder in the context of the Connected Home.
4. Requantizing the DCT coefficients

The first two methods need little explanation; dropping of frames or reducing the spatial resolution obviously result in fewer bits in the output stream and thus achieve bit rate reduction. The latter two methods are illustrated in figure 3.2. The block of 8x8 coefficients in the figure shows an example when the third

method is applied. The red circle in the figure indicates the DCT coefficients that are encoded into the outgoing video stream. The other DCT coefficients, most likely high-frequency coefficients, are discarded.

The figure also shows an example what happens when the fourth method is applied on a block of 8x8 coefficients. The left block in the figure now serves as an example block from the incoming video stream, the block on the right one is the result of the same block after requantization. As can be seen, all coefficients in the requantized block have decreased significantly in value. As a result, the variation between those coefficients has decreased with the same factor. Furthermore, small values of the original block have become zero in the requantized block. The VLC process can code the requantized block more efficient due to the lower DCT values, because low values are more likely to occur and thus receive less bits, and because of the increase of zero value coefficients.

In this project, the method of SNR scalability is used to generate the enhancement layers. As already mentioned these enhancement layers carry DCT refinement coefficients to enhance the picture quality provided by the previous layer. These DCT refinement coefficients actually are the requantization errors\(^1\) of the previous layer. This directly corresponds to the bit rate reduction technique of requantizing the DCT coefficients, and is therefore applied in this transcoder to generate the base layer.

\(^1\)A requantization error is the difference between the original value and the inverse quantized value after requantization.
3.2 Transcoding architectures

This section describes three transcoding architectures frequently occurring in the literature [3]. These architectures can all be used to generate the base layer of the scalable video stream in this project. This sections describes the behavior, the advantages and the disadvantages of each of them.

3.2.1 Cascaded pixel-domain transcoder

The most straightforward approach of transcoding a MPEG-2 video stream, is to fully decode and encode a video stream successively by simply cascading a MPEG-2 decoder and a MPEG-2 encoder. First, the MPEG-2 stream is decoded to the spatial domain by a standard MPEG-2 decoder. Then, a standard MPEG-2 encoder codes the spatial video into a MPEG-2 video stream with the desired coding parameters. This situation is depicted in figure 3.3. The main advantage of using this type of architecture is that it achieves a high picture quality since it fully decodes and re-encodes the stream. This approach can also be easily used to transcode a video stream into a different video standard.

The main disadvantage of this method is the high computational power that it requires, mainly due to the motion estimation process. To decrease the impact of the motion estimation process on the required computational power, a method called Motion Vector Refinement [6] can be used to prevent a full motion estimation in the encoding process. Motion Vector Refinement is a method where the incoming motion vectors are refined by finding a better match from the picture store by employing a search with a small search window in the region defined by the incoming motion vector. However, even when this method is used, the computational complexity required by this architecture is relatively large.

3.2.2 Open-loop transcoder

A different approach is the open-loop transcoder architecture, as shown in figure 3.4 An incoming video stream is partly decoded by applying Variable Length Decoder (VLD) and inverse quantization (IQ) operations such that the inverse quantized DCT coefficients are retrieved. Then these DCT coefficients are requantized with a coarser requantization parameter by the quantizer (Q). Finally, these requantized DCT coefficients are stored in the stream after applying the

\[\text{Motion estimation is the process of finding the best matching macroblock in inter frame coding.}\]
Variable Length Coder (VLC). In this architecture it is assumed that the motion vectors and the headers do not change during transcoding. Therefore, the incoming motion vectors and incoming headers are copied to the output stream. Since for this architecture only a few operations are required, it is much less computationally complex than the cascaded pixel-domain transcoder. However, the degree of picture quality that can be achieved is usually low. This is mainly caused by two reasons, namely:

1. The assumption that motion vectors do not change during the transcoding process is proven to be false [6].
2. The output video stream from the transcoder suffers from drift errors (continuous degradation of picture quality in a GOP).

The first reason only causes a small decrease in quality and can therefore be ignored. The second reason however causes a great decrease in picture quality of the outgoing video stream. In general, the drift errors are caused by the continuous degradation in quality of frames used as a reference in inter frame coding. This is explained as follows. Consider the situation when the back-end decoder starts to decode a GOP which begins with the I-frame. The life cycle of the blocks contained in this I-frame are shown in figure 3.5a. It shows an intra block (IB) generated by the front-end encoder. Then, after transcoding it is reduced in quality due to the requantization error $\varepsilon_1$. The decoder decodes the block and displays it. The decoder also stores this block such that it can be used as a reference by future inter blocks. Now a mismatch occurs when such a future inter coded block, represented by the prediction error (PE), uses the above mentioned block IB as a reference. This is shown in figure 3.5b. The prediction error PE is computed in the encoder by distracting the target block with the prediction block as described in section 2.3. Due to the open-loop transcoding, PE is degraded in quality due to the requantization error $\varepsilon_2$. Now, when this degraded PE is decoded it uses a block $IB - \varepsilon_1$ as a prediction block. The decoder then applies motion compensation and reconstructs block $(PE - \varepsilon_2) + (IB - \varepsilon_1)$. As a result, the requantization error $\varepsilon$ is visual in the output block twice. Even worse, the mismatch in prediction increases since this block also serves as a reference block for future inter blocks.

As time goes on, the mismatch shown above progressively increases until the end of the GOP, resulting in the reconstructed frames severely degraded in quality.
3.2. TRANSCODING ARCHITECTURES

The effect of drift when transcoding a video sequence encoded at a bit rate $R_1$ is visualized in figure 3.6. The figure shows the difference in quality between the original sequence being both re-encoded and transcoded at bit rate $R_2$. The red and blue arrows in the figure represent forward and backward motion prediction respectively, as described in section 2.3. The figure visually illustrates how the quality continuously degrades as time goes on.

Experiments were executed in this project to investigate the effect of drift throughout the GOP. Therefore a video was both re-encoded and open-loop transcoded at 3 Mbit/s. It must be noted that in the case of re-encoding, the input movie needs to be decoded first. Therefore, this is similar to when the input video would have been transcoded using the cascaded pixel domain transcoder, as described in section 3.2.1. An objective quality measurement was executed on both the transcoded and the re-encoded video. The results in this experiment are shown in figure 3.7. The figure shows the Peak Signal to Noise Ratio (PSNR) values of frames from both output movies. It is clearly shown that starting at an I-frame, which is indicated by the vertical grid lines,
the quality of the frames from the transcoded movie continuously degrades until the next I-frame. This is not perceived when looking at the graph of the re-encoded video. This observation is a direct consequence of drift. The next section discusses a transcoding architecture providing a solution for drift.

![Figure 3.7: The effect of drift on the quality of frames inside a GOP.](image)

### 3.2.3 Closed-loop transcoder

The third architecture described in this chapter, is the closed-loop transcoder architecture which provides a solution for preventing drift by compensating for the quantization error. This architecture is shown in figure 3.8. In this architecture, the requantization errors of blocks belonging to frames that are used as a reference frame are stored in the frame store (FS). Then, before a prediction error block from an inter frame is transcoded, the requantization error of the block used as a reference by the inter coded block, referred to as $\varepsilon_1$, is added to the prediction error block. This compensates for the requantization error of the reference macroblock, and therefore the difference between the reference block used in the encoder and the transcoder is minimized. A compensated prediction error block is then requantized by the quantizer (Q) using a coarser quantization parameter, Variable Length Coded (VLC) and put in the stream. When this compensated block in its turn is used as a reference block, it enters the motion compensation loops. First it is inverse quantized (IQ) to be able to calculate the requantization error. Then this requantization error is stored in the FS such that it can be used as a reference by future inter coded blocks. The Motion Compensated Prediction (MCP) process recovers the requantization error of the correct block needed as a reference by applying the incoming motion...
3.3. EVALUATION

Figure 3.8: Architecture of a closed-loop transcoder.

vector on one of the stored frames. As a result, in contrast with the situation when using an open-loop transcoder in figure 3.5, the back-end decoder would reconstruct a block $(P_E - \varepsilon_2) + IB$. This shows that the prediction mismatch is prevented since the requantization error $\varepsilon_1$ of IB was compensated for.

A major advantage of this transcoder is its moderate computational complexity. This is the result of the absence of motion estimation and the reduced number of required transformations between the spatial and the frequency domain. Another major advantage of this transcoder is that it can achieve a picture quality almost identical as can be achieved with the cascaded pixel-domain transcoder.

The main disadvantage is that it requires more effort to implement this transcoder due to the motion compensation loop present in the architecture. Also, as already mentioned in previous section, reusing motion vectors from the incoming video stream results in a slightly degradation of picture quality because the motion vectors become non-optimal due to the requantization. Therefore, the closed-loop transcoder will deliver a lower picture quality compared with the cascaded pixel-domain transcoder.

3.3 Evaluation

The transcoder in this project has two important requirements; the transcoder has to produce excellent picture quality and the transcoder has to be able to operate in real-time. The architectures described in this chapter vary a lot in both computational complexity and achievable picture quality. A comparison is shown in table 3.1. The comparison table is based on the judgements and simulation results as found in the literature.

For the transcoder in this project, a low computational complexity architecture is of high importance since it is required to operate in real time. As can be seen in table 3.1, the open-loop transcoder is the fastest architecture. Although the effect of drift can be reduced by a bit rate control algorithm [19], by assigning more bits to code macroblocks which serve as a reference for next macroblocks,
CHAPTER 3. VIDEO TRANSCODING

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Picture Quality</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascaded pixel-domain</td>
<td>Very high</td>
<td>High</td>
</tr>
<tr>
<td>Open-loop</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Closed-loop</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison between the different architectures.

the perceived quality is still worse than the other two architectures. Since the perceived picture quality is also important, this architecture has not been chosen. In contrast, the cascaded pixel-domain architecture is able to achieve the highest picture quality possible in transcoding. Unfortunately, this architecture is the most computationally complex. Therefore this architecture is not chosen as well.

A compromise in the perspective of both achievable quality and computational complexity is found in the closed-loop architecture. It provides a high picture quality while maintaining an medium degree of computational complexity. Therefore, the closed-loop architecture is chosen as the basis architecture of this project. In the next chapters, this architecture will be adapted such that it will meet all the specified requirements.
Chapter 4

Motion compensation in transcoding

In standards such as MPEG-2, motion compensation prediction (MCP) is used in inter frame coding to exploit the temporal redundancy between succeeding frames. These inter coded frames consist of residual data blocks and motion vectors. A residual data block contains the prediction error, which was originally calculated by the encoder by subtracting the block to be encoded by a best matching reference block. The encoder uses previous and/or future frames for finding the best matching block. A motion vector describes the position of the best matching macroblock in the reference frame. In MPEG-2 decoding, MCP is used to recover a best matching block from the Frame Store (FS).

In a closed-loop transcoder, MCP is used to recover the requantization error from a best matching block, further referred to as block $\hat{x}$, when it is used as a reference by an incoming inter coded block. Then, before transcoding the incoming block, block $\hat{x}$ is added to it, to prevent drift in the output stream, as described in section 3.2.3. Thus, MCP in a closed-loop transcoder is the operation of obtaining block $\hat{x}$ from the FS. This is graphically depicted in figure 4.1. As shown in the figure, the MCP process uses at most 4 blocks. The actual number of intersecting blocks depends on the misalignment of $\hat{x}$ with the blocks used to recover $\hat{x}$, stored in the FS. The position of $\hat{x}$, and as a result the actual blocks to be used from the FS, are specified by the incoming motion vector. MCP can either be performed in the spatial domain, where the reference frames stored in the FS are represented with spatial values, or in the frequency domain, where the reference frames are represented with DCT values.

The purpose of this chapter is to determine which type of MCP, either the spatial or frequency domain, can be best used to meet the requirements of the transcoder from both the achievable picture quality and real-time performance point of view. The first section of this chapter describes a special approach of performing MCP in the spatial domain [7]. In the second section, this approach is adapted such that it performs MCP in the DCT domain (DCT-MCP) [7]. The third section describes three different approaches to speed up the basic approach of DCT-MCP. Finally, this chapter is concluded with an evaluation.
about the type of MCP that is the best choice with respect to this project.

4.1 Motion compensation in the spatial domain

As described above, MCP in the spatial domain recovers blocks from the FS, which stores at most two frames in the spatial domain. These frames serve as reference frames for inter coded frames; one for forward predicted frames and one for backward predicted frames. These reference frames are partitioned into blocks with a size of 8x8 pixels. Now, when MCP is applied, the situation can occur that block $\hat{x}$ does not align with the 8x8 blocks partitioning of the reference frame. To be more specific, block $\hat{x}$ may intersect with at most four neighboring spatial domain blocks, denoted as $x_1$, $x_2$, $x_3$ and $x_4$. Therefore, block $\hat{x}$ can also be expressed as a superposition of appropriately shifted and restricted versions of $x_1$, $x_2$, $x_3$ and $x_4$, i.e.:

$$\hat{x} = \sum_{i=1}^{4} c_{i1} x_i c_{i2}$$  \hspace{1cm} (4.1)

where $c_{ij}, i = \{1,...,4\}, j = \{1,2\}$, are sparse 8x8 matrices of zeros and ones that perform shift operations accordingly. This situation is shown in figure 4.2.

Figure 4.1: Basic principle of motion compensation.

Figure 4.2: Example of a reference block $\hat{x}$, represented by the red square, not aligned with 8x8 blocks of reference frame.
To illustrate how the matrices can be shifted, consider the following matrix $X$:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Assume that only values $\{1, 2, 4, 5\}$ are of interest and they need to be shifted to the lower-right corner. This can be achieved by a pre-multiplication with a shifting matrix $K$, followed by a post-multiplication with a shifting matrix $L$, where matrices $K$ and $L$ are defined as:

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad L = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

The result of these multiplications are stored in matrix $C$:

$$C = K \cdot X \cdot L = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \{\text{Multiply } K \text{ with } X\}$$

$$C = (K \cdot X) \cdot L = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \{\text{Multiply } (K \cdot X) \text{ with } L\}$$

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 4 & 5 \end{bmatrix}$$

As shown, shifting and restriction can be achieved by multiplying a matrix with appropriate shifting matrices. Also, due to the definition of matrix multiplication, the remainder of a shifted matrix, i.e. values not subject to the shifting operations, is filled with zeros.

A shifting matrix has zeros everywhere, except for an identity sub-matrix $I_h$. A $h \times h$ identity matrix is the matrix $I$ that has ones down the main diagonal and zeros everywhere else. A shifting matrix can be used to shift values inside a matrix $n$ positions to the right, left, up or down. Shifting in the diagonal direction is not possible with one shifting operation. When a $m \times m$ matrix is post-multiplied with a shifting matrix containing identity matrix $I_h$, the number of positions $n$ that are shifted is: $n = m - h$.

Shifting operations, as illustrated in the example above, can be used on 8x8 matrices in a similar way. For each block $x_1$, $x_2$, $x_3$ and $x_4$, at most two shift operations are needed. Only one shift operation is needed when block $\hat{x}$ aligns with the 8x8 partitioning of the reference frame in one direction. Obviously, when block $\hat{x}$ aligns with the partitioning in two directions, no shift operations are needed since $\hat{x}$ can be extracted directly. The shifting matrices are referred to as $c_{ij}$, where $1 \leq i \leq 4$ and $1 \leq j \leq 2$. Pre-multiplying a matrix with one of the shifting matrices $c_{i1}$ performs a vertical shifting operation, while
post-multiplication with one of the matrices $c_{ij}$ performs a horizontal shifting operation. Thus, eight types of window and shifting matrices are defined. Furthermore, each type of shifting matrix has seven variants, depending on the variables $h$ and $w$ representing the height and width parameters, as defined as shown in figure 4.2. In turn, the appropriate values of variables $h$ and $w$ are determined by the incoming motion vector.

Concluding, it is described that MCP in the spatial domain can be achieved by shifting operations on blocks represented in the spatial domain. The next section describes a similar approach for blocks represented in the DCT domain.

### 4.2 Motion compensation in the DCT domain

The previous section presented an approach of applying MCP in the spatial domain. In this section, this approach is adapted such that MCP is executed entirely in the frequency domain. As a result, there is no need to transform a block to the spatial domain and therefore saving on the use of both the DCT and IDCT operations. This reduced closed-loop architecture is shown in figure 4.3. When this figure is compared with its spatial domain equivalent depicted in figure 3.8, two different components can be distinguished. One of them is the DCT Frame Store (DCT-FS), which is equivalent to its spatial domain counterpart FS in the sense that it stores quantization errors of frames serving as a reference frame, only now in the frequency domain. The other component, the DCT-MCP performs MCP in the DCT domain (DCT-MCP). It therefore takes DCT blocks of a reference frame and motion vectors as inputs, and delivers DCT blocks of the corresponding spatial domain blocks, without any reference to past and future frames. Since it is not possible to perform MCP directly on the raw DCT blocks, there must be some form of transformation to the spatial domain. Therefore, a method of performing DCT and IDCT operations with matrix multiplications is described first.
The formal definition of the two-dimensional DCT and inverse DCT (IDCT) operations to transform a block $x$ to a frequency domain block $X$ and back to the spatial domain respectively are as follows:

$$X(k, l) = \frac{c(k)}{2} \frac{c(l)}{2} \sum_{n=0}^{7} \sum_{m=0}^{7} x(n, m) \cos(\frac{2n + 1}{16} \cdot k\pi) \cos(\frac{2m + 1}{16} \cdot l\pi) \tag{4.2}$$

$$x(n, m) = \sum_{k=0}^{7} \sum_{l=0}^{7} c(k) c(l) \frac{x}{2} X(k, l) \cos(\frac{2n + 1}{16} \cdot k\pi) \cos(\frac{2m + 1}{16} \cdot l\pi) \tag{4.3}$$

where for both formulas $0 \leq k, l, m, n \leq 7$, $c(0) = \frac{1}{\sqrt{2}}$ and $c(k) = 1$ for $k > 0$ hold.

Derived from formula 4.2, a $8 \times 8$ matrix $S = \{s(k,n)\}_{k,n=0}^{7}$ can be defined performing the one-dimensional DCT operation as follows:

$$s(k, n) = \frac{c(k)}{2} \cos(\frac{2n + 1}{16} \cdot k\pi) \tag{4.4}$$

In [7] it is shown that the transformation of a spatial block $x$ to the frequency domain block $X$ can be defined as:

$$X = SxS^t \tag{4.5}$$

where the superscript $t$ denotes matrix transposition. Similarly, let the superscript $-t$ denote transposition of the inverse, then spatial block $x$ can be calculated from frequency block $X$ by:

$$x = S^{-1}XS^{-t} = S^tXS \tag{4.6}$$

Using the above formulas, the basic formula for DCT-MCP can now be derived from formula 4.1 as follows:

$$\hat{x} = \sum_{i=1}^{4} c_{i1}x_i c_{i2}$$

{ use formula 4.6 }

$$\hat{x} = \sum_{i=1}^{4} c_{i1}(S^t X_i S) c_{i2}$$

{ use formula 4.5 }

$$\hat{x} = \sum_{i=1}^{4} c_{i1}(S^t(Sx_iS^t)S)c_{i2}$$

{ Pre- multiply both sides by $S$, and post-multiply both sides with $S^t$ }

$$S\hat{x}S^t = \sum_{i=1}^{4} S(c_{i1}(S^t(Sx_iS^t)S)c_{i2})S^t$$

{ Simplify }

$$\hat{X} = \sum_{i=1}^{4} C_{i1}X_i C_{i2} \tag{4.7}$$
where $C_{ij}$ are the DCT transformed versions of matrices $c_{ij}$. Since matrices $c_{ij}$ are constant, matrices $C_{ij}$ are constant as well and can therefore be pre-computed.

Formula 4.7 describes the approach of applying motion compensation directly in the frequency domain. Now it is interesting to determine how many calculations are required by this new approach, compared with the traditional technique of MCP in the spatial domain, which should not be confused with the method described in section 4.1. The amount of calculations is divided into the amount of additions and multiplications. For formula 4.7 this is calculated as follows:

$$\sum_{i=1}^{4} C_{i1} X_i C_{i2}$$

= { Define $MM =$ amount of calculations needed for 8x8 matrix multiplication} 

$$\sum_{i=1}^{4} 2 \cdot MM$$

= { Simplify. Define $MA =$ amount of calculations needed for 8x8 matrix addition } 

$$8 \cdot MM + 3 \cdot MA$$

= { Use $MM = 64 \cdot (8 \cdot MULT + 7 \cdot ADD)$ } 

$$8 \cdot (64 \cdot (8 \cdot MULT + 7 \cdot ADD)) + 3 \cdot MA$$

= { Simplify. Use $MA = 64 \cdot ADD$ } 

$$4096 \cdot MULT + 3776 \cdot ADD$$

where $ADD$ and $MULT$ denote one add and one multiplication operation, respectively.

When performing MCP in the spatial domain, only the DCT and IDCT operations are considered because the rest of the required calculations can be neglected. The amount of calculations needed to perform a DCT has been determined by counting the numbers of additions and multiplications in a real implementation of a fast DCT. The shift operations occurring in this implementations were counted as multiplications. Since the IDCT operation is the inverse operation of a DCT, the number of calculations of an IDCT operation is assumed to be identical to the number of calculations of the DCT. To be able to perform MCP in the spatial domain, 4 blocks have to be transformed to the spatial domain and back to the frequency domain, thus 8 transformations are needed.

The number of calculations for both the frequency and spatial domain approach are shown in table 4.1. As can be seen in table 4.1 DCT-MCP requires more calculations than its spatial domain equivalent. As such, in this form DCT-MCP

---

1 In contrast with matrices $c_{ij}$, matrices $C_{ij}$ are dense, which is a result of the DCT transformations.

2 Extracting a block from four other blocks represented in the spatial domain does not require any extra computations, due to the fact that a value of a block directly corresponds to the spatial value of a pixel. As such, each value of the best matching block pointed by a motion vector can be read directly from the FS. In contrast, when a block is represented in the DCT domain, a value of a block corresponds to a frequency value.
4.3 Possible optimizations

The technique of MCP in the DCT domain described in the previous section requires more calculations than its spatial domain counterpart. This section presents some methods to decrease the amount of needed calculations.

4.3.1 Factorize shifting matrices

A first method of achieving optimization is found in the literature \[7\] by using two observations. One observation is that some of the shifting matrices \(C_{ij}\) used in section 4.2 are equal to each other for any given \(w\) and \(h\). This is shown in table 4.2. Table 4.2 shows that \(c_{11} = c_{21}, c_{31} = c_{41}, c_{12} = c_{32}\) and \(c_{22} = c_{42}\). When solving the basic equation for DCT-MCP, this observation can be used to save on two matrix multiplications as follows:

\[
\hat{X} = \sum_{i=1}^{4} C_{i1} X_i C_{i2}
\]

\[
= \{ \text{Use definition of } \sum_{i=1}^{4} \} \]

\[
\hat{X} = (C_{11} \cdot X_1 \cdot C_{12}) + (C_{21} \cdot X_2 \cdot C_{22}) + (C_{31} \cdot X_3 \cdot C_{32}) + (C_{41} \cdot X_4 \cdot C_{42})
\]

\[
= \{ \text{Use observation } \}
\]

\[
\hat{X} = C_{11} \cdot (X_1 \cdot C_{12} + X_2 \cdot C_{22}) + C_{31} \cdot (X_3 \cdot C_{32} + X_4 \cdot C_{42}) \quad (4.8)
\]

The other observation is that rather than fully pre-compute shifting matrices \(C_{ij}\) it might be more efficient to leave these matrices factorized into relatively sparse matrices. A matrix factorization (or matrix decomposition) is the right-hand-side product in \(A = F_1 F_2 \ldots F_k\) for matrix \(A\). When matrix \(A\) is sufficiently structured, matrices \(F_k\) describe only simple operations.

First, matrix factorization is applied on matrix \(S\), which is defined in formula 4.4. After matrix factorization, matrix \(S\) can be represented as:

\[
S = DPB_1B_2MA_1A_2A_3 \quad (4.9)
\]

where matrices \(D\), \(P\), \(B_1\), \(M\) and \(A_i\) are sparse matrices performing simple operations, compared with the complexity of a matrix multiplication. This is further

<table>
<thead>
<tr>
<th>MCP spatial domain</th>
<th># ADD</th>
<th># MULT</th>
<th># Calculations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCP frequency domain</td>
<td>3776</td>
<td>4096</td>
<td>7872</td>
</tr>
</tbody>
</table>

Table 4.1: Number of calculations required for motion compensation in both the spatial and frequency domain.

can not be used to increase the performance of the transcoder. Fortunately however, optimizations for the frequency domain approach are found in the literature, which result in a significant increase in performance. These optimizations are discussed in the next section.
CHAPTER 4. MOTION COMPENSATION IN TRANSCODING

Table 4.2: The result of matrix multiplication with matrices $c_{ij}$. The red square indicates the values subject to the shifting operation.

<table>
<thead>
<tr>
<th>Shifting operation</th>
<th>Shifting matrix</th>
<th>Result of operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-multiply with $c_{11}$</td>
<td>$c_{11} = \begin{pmatrix} 0 &amp; I_h \ 0 &amp; 0 \end{pmatrix}$</td>
<td><img src="image1" alt="Result" /></td>
</tr>
<tr>
<td>Pre-multiply with $c_{21}$</td>
<td>$c_{21} = \begin{pmatrix} 0 &amp; I_h \ 0 &amp; 0 \end{pmatrix}$</td>
<td><img src="image2" alt="Result" /></td>
</tr>
<tr>
<td>Pre-multiply with $c_{31}$</td>
<td>$c_{31} = \begin{pmatrix} 0 &amp; 0 \ I_{8-h} &amp; 0 \end{pmatrix}$</td>
<td><img src="image3" alt="Result" /></td>
</tr>
<tr>
<td>Pre-multiply with $c_{41}$</td>
<td>$c_{41} = \begin{pmatrix} 0 &amp; 0 \ I_{8-h} &amp; 0 \end{pmatrix}$</td>
<td><img src="image4" alt="Result" /></td>
</tr>
<tr>
<td>Post-multiply with $c_{12}$</td>
<td>$c_{12} = \begin{pmatrix} 0 &amp; 0 \ I_w &amp; 0 \end{pmatrix}$</td>
<td><img src="image5" alt="Result" /></td>
</tr>
<tr>
<td>Post-multiply with $c_{22}$</td>
<td>$c_{22} = \begin{pmatrix} 0 &amp; I_{8-w} \ 0 &amp; 0 \end{pmatrix}$</td>
<td><img src="image6" alt="Result" /></td>
</tr>
<tr>
<td>Post-multiply with $c_{32}$</td>
<td>$c_{32} = \begin{pmatrix} 0 &amp; 0 \ I_w &amp; 0 \end{pmatrix}$</td>
<td><img src="image7" alt="Result" /></td>
</tr>
<tr>
<td>Post-multiply with $c_{42}$</td>
<td>$c_{42} = \begin{pmatrix} 0 &amp; I_{8-w} \ 0 &amp; 0 \end{pmatrix}$</td>
<td><img src="image8" alt="Result" /></td>
</tr>
</tbody>
</table>

Now, when a matrix is multiplied with the matrix factorization of matrix $S$, defined as depicted in formula 4.9, it only needs $112$ MULT + $224$ ADD instructions. This is far more efficient than the brute force approach of multiplying a matrix with the standard matrix $S$, requiring $448$ ADD + $512$ MULT instructions.

Next, using both the observation of the equivalent shifting matrices and matrix factorization, it is shown in [7], that formula 4.7 can be rewritten as:

\[
\hat{X} = [J_h B_2^T B_1^T P^T D (X_1 D P B_1 B_2 J_1^T + X_2 D P B_1 B_2 K_{h-w}^T)] + [K_{h-h} B_2^T B_1^T P^T D (X_3 D P B_1 B_2 J_2^T + X_4 D P B_1 B_2 K_{h-w}^T)] S^T
\]

(4.10)

where matrices $D$, $P$, $B_i$, $M$ and $A_i$ are sparse factorization matrices, $X_i$ are the reference blocks, $J_i$, $J_i^T$, $K_i$ and $K_i^T$ are the factorized versions of shifting matrices $C_{ij}$ and $S$ and $S^T$ are the 1-dimensional DCT matrices. The factorization matrices of $S$ are already given in formula 4.9. The paper also presents an efficient way of calculating $J_i$ or $K_i$ with only 5 MULT and 22 ADD operations.

The total number of operations for calculating formula 4.10 according to the
paper is 638 $\text{MULT}$ and 967 $\text{ADD}$ operations, which gives a total amount of 1605 calculations. This achieves an optimization of almost 80% compared with the brute force method of solving formula 4.7. Compared with its spatial domain counterpart, it achieves an optimization of almost 59%. The latter result corresponds with the conclusions about the performance of DCT-MCP made by the authors of paper [7].

4.3.2 Replacing fixed point operations with binary operations

The second method to optimize the performance of DCT-MCP found in the literature [8] shows how formula 4.7 can be computed with reduced number and complexity of the operations.

In order to achieve reduced computational complexity the number of basic integer operations are increased. As such, the fixed point values of the shifting matrices $C_{ij}$ are approximated to binary numbers with a maximum deviation of $\frac{1}{32}$. This is shown in the next example:

$$C_{11} = \begin{pmatrix}
0.500000 & -0.453064 & -0.000000 & \cdots & 0.090120 \\
0.453064 & -0.326641 & -0.357867 & \cdots & 0.135299 \\
0.000000 & 0.207867 & -0.500000 & \cdots & 0.138893 \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
-0.090120 & 0.135299 & -0.138893 & \cdots & 0.326641
\end{pmatrix}$$

is approximated by matrix $C'_{11}$:

$$C'_{11} = \begin{pmatrix}
\frac{1}{2} - \frac{1}{16} & -\frac{1}{16} + \frac{1}{16} & 0 & \cdots & \frac{1}{8} - \frac{1}{16} \\
\frac{1}{2} - \frac{1}{16} & -\frac{1}{16} + \frac{1}{16} & -\frac{1}{16} + \frac{1}{8} & \cdots & \frac{1}{4} + \frac{1}{16} \\
0 & \frac{1}{16} - \frac{1}{16} & 0 & \cdots & \frac{1}{4} + \frac{1}{16} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
-\frac{1}{8} + \frac{1}{16} & \frac{1}{8} & \frac{1}{8} & \cdots & \frac{1}{4} + \frac{1}{16}
\end{pmatrix}$$

Unfortunately, shifting a quantization error block $X_i$ with a shifting matrix $C'_{ij}$ would result in a lot of values rounded to zero, which is caused by most of the DCT coefficients having a low value. To prevent this, the values of the DCT coefficients of a block $X_i$ are multiplied with a scaling factor $2^8$. This scaling factor can be included in the quantization and inverse quantization process, and therefore does not introduce significant additional complexity. Now, shifting the scaled versions of blocks $X_i$ with the new shifting matrices only requires binary add and shift operations on integer numbers, which on most platforms are less computationally complex than multiplication and add operations on fixed point numbers.

To further reduce the number of required computations of the shifting operations is achieved by employing simple data manipulation. The idea is that all DCT coefficients from block $X_i$ in the same column that are multiplied by $k$ are
added together before multiplication, then this would result in less required multiplication operations. This is shown in the following example using the second row of shifting matrix $C'_{11}$:

\[
\begin{align*}
\left(\frac{1}{2} - \frac{1}{16}\right)z_1 + \left(-\frac{1}{4} - \frac{1}{16}\right)z_2 + \left(-\frac{1}{4} - \frac{1}{8}\right)z_3 &= \quad (4.11) \\
\frac{1}{2}z_1 - \frac{1}{16}z_1 - \frac{1}{4}z_2 - \frac{1}{16}z_2 - \frac{1}{4}z_3 - \frac{1}{8}z_3 &= \quad (4.12) \\
\frac{1}{2}z_1 - \frac{1}{4}(z_2 + z_3) - \frac{1}{8}z_3 - \frac{1}{16}(z_1 - z_2) &= \quad (4.13)
\end{align*}
\]

where formula 4.11 is part of a matrix multiplication when shifting a block $X_i$ and $z_i$ are actual DCT coefficients of block $X_i$. Calculation of formula 4.11 needs $5$ ADD and $6$ MULT operations, whereas formula 4.13 can be calculated using $5$ ADD and only $4$ MULT operations. Although this is a fairly simple method the achieved reduction of MULT is enormous. This is determined by examining the number of equivalent MULT operations present in the shifting matrices where the coefficients of these matrices are rounded to integer values as described in the method above. The result was that for each row and column in a shifting matrix, on average $6$ MULT operations could be saved employing the simple data manipulation. For the actual multiplication of a matrix with a shifting matrix, either post-multiplication or pre-multiplications, the reduction in the amount of required multiplications is then $64 \cdot 6 = 384$. Thus for multiplying a matrix with a shifting matrix using the methods described in this section, now only requires $128$ MULT operations which is a reduction of $75\%$ as compared with brute-force matrix multiplications. The number of ADD operations remain unchanged.

Next the authors of paper [8] also use the same observation as described in the previous subsection that some shifting operations can be combined. Thus for solving equation 4.7 only six matrix multiplications and three matrix additions are needed. Using the methods as described in this subsection this results in a total number of required operations of:

\[
\begin{align*}
\sum_{i=1}^{4} C_{1i} X_i C_{i2} &= \{\text{Define } MM = \text{amount of calculations needed for 8x8 matrix multiplication}\} \\
\sum_{i=1}^{4} 2 \cdot MM &= \{\text{Define } MA = \text{amount of calculations needed for 8x8 matrix addition}\} \\
8 \cdot MM + 3 \cdot MA &= \{\text{Use observation of similar matrices}\} \\
6 \cdot MM + 3 \cdot MA &= \{\text{Use } MM=128 \text{ MULT operations and } 448 \text{ ADD operations}\} \\
6 \cdot (128 \cdot MULT + 448 \cdot ADD) + 3 \cdot (192 \cdot ADD) &= \{\text{Simplify}\} \\
768 \cdot MULT + 2880 \cdot ADD
\end{align*}
\]
4.3. POSSIBLE OPTIMIZATIONS

According to this calculation, the total amount of optimization achieved when solving formula 4.7 is almost 54%. However, the fact that the algorithm is based on integer based computations, whereas formula 4.7 is based on fixed point computations, would further increase the amount of achieved reduction by this optimization. Experiments show that on a the PC platform, matrix multiplications with integer matrices is roughly 20% faster than when the matrices are filled with fixed point values.

From the above it can be concluded that this method achieves an optimization of a maximum of 74 % in solving formula 4.7 concerning DCT-MCP. However, the authors claim to have achieved a far better level of optimization, namely 81 % compared to the spatial domain method. It is expected that they use the method of factorizing the shifting matrices, as described in the previous section, to achieve the remaining amount of optimization. Unfortunately, this was not documented in their paper.

4.3.3 Exploit sparseness of blocks $X_i$

The third type of optimization described in this chapter is a method presented in paper [9]. In this paper, optimization is achieved by exploiting the sparseness of the quantization error blocks $X_i$. It is based on the fact that when performing a shifting operation on a block $X_i$ a lot of multiplications with zeros are performed. Preventing the execution of these redundant multiplications results in a reduction of the total amount of multiplications needed to perform a shifting operation. This is shown in the next example where a block $M$ is post-multiplied with a matrix $N$:

$$M = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}, \quad N = \begin{pmatrix} r & s & t \\ u & v & w \\ x & y & z \end{pmatrix}$$

This results in the following matrix:

$$(M \cdot N)_{ij} = \sum_{k=1}^{3} M_{ik} \cdot N_{kj} = \begin{pmatrix} ar + bu + cx & as + bv + cy & at + bw + cz \\ dr + eu + fx & ds + ev + fy & dt + ew + fz \\ gr + hu + ix & gs + hv + iy & gt + hw + iz \end{pmatrix}$$

The above 3x3 matrices are chosen for the purpose of illustration. Assume matrix $M$ is part of an actual quantization error block $X_i$. As such, matrix $M$ contains DCT values where most DCT coefficients are expected to be zero. Now consider element $e$ of matrix $M$, which contributes to the second row of matrix $M \cdot N$. In this row element $e$ is involved in 3 add and 3 multiply operations. If, for example, element $e$ has the value of zero then the above implies that the 3 add and 3 multiply operations did not contribute to the final result. As such, when each value is inspected for having the value zero before the additions and multiplications were performed, the number of operations would be reduced.

The paper presents an algorithm for matrix post-multiplications requiring $nx$ multiplications and $nx$ additions, where $n$ is the square root of the dimension of matrices $M$ and $N$ and $x$ is the number of non-zero DCT coefficients in matrix $M$. So, when half of the DCT coefficients have the value zero, which is
not unimaginable, the amount of both additions and multiplications are $8 \cdot 32 = 256$. Compared with brute-force matrix multiplication, this is a reduction of approximately 50%. This is directly translated to the amount of calculations needed for solving formula (4.7). Again, assume 50% of the DCT coefficients of each block $X_i$ have a non-zero value, then using this optimization only 2048 add and 2048 multiply operations are needed for solving the equation.

4.4 Evaluation

In this chapter the different approaches for MCP have been discussed. Furthermore, three methods of optimizing the basic operation of performing DCT-MCP are described. Furthermore, the amount of operations they require are examined to make a judgement about the achieved level of optimization. These results are listed in table 4.3. Note that the negative values of achieved optimization denote that the corresponding method actually performs worse than the spatial domain MCP. Furthermore, it must be mentioned that the values corresponding the third method were based on the assumption that half of the DCT coefficients of a block have the value of zero. An increase of this number would yield a high level of optimization and visa versa.

<table>
<thead>
<tr>
<th>Method</th>
<th># Operations</th>
<th>Achieved optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT-MCP standard</td>
<td>7872</td>
<td>-105 %</td>
</tr>
<tr>
<td>DCT-MCP method 1</td>
<td>1605</td>
<td>59 %</td>
</tr>
<tr>
<td>DCT-MCP method 2</td>
<td>531</td>
<td>81 %</td>
</tr>
<tr>
<td>DCT-MCP method 3</td>
<td>3936</td>
<td>-2 %</td>
</tr>
</tbody>
</table>

Table 4.3: Achieved optimization of the different methods of DCT-MCP compared with MCP in the spatial domain.

The table shows that the second method of optimization outperforms the others in terms of performance. As already mentioned, the paper presenting this method does not include a detailed description of the algorithm. It is expected that a large amount of optimization is caused by applying the method of matrix factorization, similar as described in section 4.3.1 although with different factorized matrices. Since these matrices are not mentioned, it will be almost impossible to implement this method.

Optimization method 1 is the secondly best performing algorithm. An advantage of this method is the detailed description of the algorithm, making it straightforward to implement.

Concluding, the first method is the most suitable method to be applied in the transcoder for this project. According to the calculations, it is expected that this method of DCT-MCP achieves an optimization of almost 59 % compared with the spatial domain counterpart. Unfortunately, this only holds for forward-predicted blocks since they only need one block as a reference. A bi-directional predicted block uses two reference blocks which have to be recovered from both the past reference frame and the future reference frame, both stored in the DCT-
FS. A direct consequence of storing the DCT coefficient in memory is that the DCT-MCP operation has to be performed twice for each bi-directionally predicted block; one on each reference frame. This is caused by the fact that the DCT-FS stores the DCT coefficients, and for each DCT-MCP operation the frequency values are virtually transformed to the spatial domain to enable the extraction of a block. In contrast, applying MCP on a bi-directionally predicted block in the spatial domain only results in a small increase of required computations. This is because the values stored in the FS are already transformed to the frequency domain by the DCT operation.
Chapter 5

Bit rate control

The SNR scalable video stream generated by the transcoder consists of a base layer and up to two enhancement layers. The base layer is generated using the requantizing DCT coefficients bit rate reduction technique as described in section 3.1. According to the SNR scalability syntax, described in section 2.5, an enhancement layer carries the requantization errors of the previous layer. A bit rate control process is needed to generate the layers such that the target bit rate for each layer is met.

In general, bit rate control for the base layer is equivalent to bit rate control for a non-scalable MPEG-2 stream in both encoding and transcoding. In transcoding however, coding parameters can be obtained from the incoming video stream that can be used in the bit rate control process. One of the algorithms investigated was based on a Piecewise Linearly Decreasing (PLD) model [10], describing the relation between the requantization parameter and the ratio between the incoming bit rate and the target bit rate. This algorithm is described in appendix B. The model allows for an easy calculation of the requantization parameter to meet a certain target bit rate. Unfortunately, this model assumes that blocks in the incoming video are quantized using a fixed quantization parameter. This assumption cannot be guaranteed for all video streams. Thus, unfortunately this model can not be used in this project. Therefore, an existing bit rate control algorithm was adapted, such that it can be used as a basis for developing a bit rate control algorithm for the base layer. A widely used bit rate control algorithm is described in the first section of this chapter. This algorithm is adapted in section two such that all the requirements concerning bit rate control for the base layer can be met. In the third section, the approach for bit rate control for the enhancement layers is discussed.

5.1 TM-5 bit rate control algorithm

During the development of the MPEG-2 standard a Test Model was written, which served as a cook book for creating bit streams during the collaborative co-experimental phase of MPEG-2 video. The many documented experiments were an attempt to verify the usefulness of various proposed coding techniques. When the proposal met the criteria, e.g. coding gain, implementation complex-
CHAPTER 5. BIT RATE CONTROL

ity, robustness, it survived. The fifth version of the test model [11] was the last major update of the Test Model and contains a proposal of dealing with the problem of bit rate control. This bit rate control algorithm became widely known as TM-5 and is often used in relatively simple MPEG-2 encoders.

The algorithm is described in detail as follows:

- **Step 1: Bit allocation for next GOP**
  At the start of a new GOP, a target bit size for the next GOP is computed:
  \[ R = \frac{\text{target bit rate} \cdot N}{\text{frame rate}} \]  
  (5.1)
  where \( N \) is the number of pictures in the next GOP and \( R \) is the bit target for the GOP to be transcoded.

- **Step 2: Bit allocation for each Picture in the GOP**
  This step computes the target bit size for each picture by the following formulas according to the corresponding picture type:
  \[ T_i = \max \left( \frac{R}{1 + \frac{N_p R_p X_i}{X_i, K_p} + \frac{N_b R_b X_b}{X_b, K_b}}, \frac{\text{target bit rate}}{8 \cdot \text{frame rate}} \right) \]  
  (5.2)
  \[ T_p = \max \left( \frac{R}{N_p + \frac{N_b R_b X_b}{X_b, K_b}}, \frac{\text{target bit rate}}{8 \cdot \text{frame rate}} \right) \]  
  (5.3)
  \[ T_b = \max \left( \frac{R}{N_b + \frac{N_p R_p X_p}{X_p, K_p}}, \frac{\text{target bit rate}}{8 \cdot \text{frame rate}} \right) \]  
  (5.4)
  where \( N_p \) and \( N_b \) are the number of P- and B-pictures remaining in the current GOP, respectively. As can be seen, the global complexity measures \( X_i, X_p \) and \( X_b \) are used to compute the target bit size for a picture. This ensures that relatively more bits are allocated to relatively more complex pictures.

- **Step 3: Rate control**
  This step computes a reference requantization parameter \( Q_j \) for each macroblock \( j \) of a picture with the aim of meeting the target bit size of a picture as close as possible. Before transcoding each macroblock \( j \), the fullness of appropriate virtual buffers, one for each picture type, is computed:
  \[ d_i^j = d_i^0 + B_{j-1} - \left( \frac{T_i \cdot (j - 1)}{\text{macroblock cnt}} \right) \]  
  (5.5)
  \[ d_p^j = d_p^0 + B_{j-1} - \left( \frac{T_p \cdot (j - 1)}{\text{macroblock cnt}} \right) \]  
  (5.6)
  \[ d_b^j = d_b^0 + B_{j-1} - \left( \frac{T_b \cdot (j - 1)}{\text{macroblock cnt}} \right) \]  
  (5.7)
  where \( B_j \) is the number of bits used to generate all preceding macroblocks up to and including macroblock \( j \) and \( \text{macroblock cnt} \) is the total number
5.1. TM-5 BIT RATE CONTROL ALGORITHM

of macroblocks in the picture. The fullness of the appropriate virtual buffer $d^{i}_{j}$, $d^{p}_{j}$ or $d^{b}_{j}$ is now used to compute a reference requantization parameter:

$$Q_{j} = \left( \frac{d_{j} \cdot 31}{r} \right)$$ (5.8)

As shown with these formulas, the fullness of virtual buffers $d^{i}_{j}$, $d^{p}_{j}$ and $d^{b}_{j}$ have a large impact on the value of the reference requantization parameter $Q_{j}$. In turn, $Q_{j}$ indirectly influences the fullness of the same virtual buffers by means of the sizes of the generated macroblocks. Therefore, this mechanism can be seen as a incremental feedback loop\textsuperscript{1} Rate control of a picture is further explained with figure 5.1. The red line in the figure acts as a lower bound for the virtual buffer and is defined by the latter sub-formula in formulas 5.5 - 5.7. The black line denotes the value of $B_{j-1}$; the total amount of bits needed to code up to and including macroblock $j-1$.

- **Step 4: Adaptive quantization**
  This step modulates the reference requantization parameter according to

\textsuperscript{1}Incremental here means that the result of a change of requantization parameter $Q_{j}$ is notified when determining the fullness of the virtual buffers when transcoding the next macroblock

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{rate_control}
\caption{An example of rate control for an I-picture.}
\end{figure}
the spatial activity in the macroblock to derive the value of the final requantization parameter $mquant$, that is used to requantize the macroblock. The spatial activity measure for macroblock $j$ is computed from the four luminance frame-organised sub-blocks ($n=1..4$) and the four luminance field-organised sub-blocks ($n=5..8$) using the intra (i.e. original) pixel values:

$$act_j = 1 + \min(vblk_1, vblk_2, vblk_3, ..., vblk_8)$$ (5.9)

where

$$vblk_k = \frac{1}{64} \sum_{k=1}^{64} (P^n_k - P\text{mean}_n)^2$$ (5.10)

and

$$P\text{mean}_n = \frac{1}{64} \sum_{k=1}^{64} P^n_k$$ (5.11)

and $P_k$ are the sample values in the $n$-th original 8*8 block. Then $act_j$ is normalized:

$$N\cdot act_j = \frac{(2 \cdot act_j) + \text{avg}.act}{act_j + (2 \cdot \text{avg}.act)}$$ (5.12)

where $\text{avg}.act$ is the average value of $act_j$ of the last picture to be transcoded. Using the above computed values $mquant$ can now be computed:

$$mquant_j = Q_j \cdot N\cdot act_j$$ (5.13)

where $Q_j$ is the reference requantization parameter obtained in step 3. The final value of $mquant_j$ is clipped to the range [1..31] and is used to requantize macroblock $j$.

**Step 5: Updating**

After a picture is transcoded, the global complexity measure corresponding to the picture type is updated as follows:

$$X_i = S_i \cdot Q_i$$ (5.14)

$$X_p = S_p \cdot Q_p$$ (5.15)

$$X_b = S_b \cdot Q_b$$ (5.16)

In the above formulas, $S_i$, $S_p$ and $S_b$ are the sizes in bits of the transcoded pictures and $Q_i$, $Q_p$ and $Q_b$ are the average requantization parameter used for requantizing the picture.

The algorithm described above is usually used in MPEG-2 encoding, but can be used in transcoding as well. Therefore, and due to its low computational complexity, it is chosen to serve as a base for the bit rate control for the transcoder. Unfortunately, to use it in the case of real-time transcoding, it needs to be adapted to meet all requirements. This adaptation is described in the next section.
5.2 Adapting TM-5 to satisfy the required reaction time

As being said before, standard TM-5 is not suitable in case of real-time transcoding as required in this project. Therefore, it needs to be adapted such that the requirements concerning the bit rate control of the base layer can be met. The next subsections describe the problems of TM-5 when being applied in real-time transcoding, and the solutions for them.

5.2.1 TM-5 requires a GOP delay

TM-5 can only determine the target bit budget for the next GOP and the pictures contained in the GOP, when the total number of pictures ($N$), the number of P-pictures ($N_p$) and the number of B-pictures ($N_b$) are known. Therefore, TM-5 requires a delay of one GOP to determine these numbers. In this project, a GOP delay is unacceptable and should therefore be removed.

In general, the GOP structure in MPEG-2 encoded video is fixed when a relatively simple encoder was used. However, most video material is encoded using a more advanced encoder which can decide to start a new GOP when a scene change occurs to increase coding efficiency. Therefore, for most video streams subject to transcoding, it must be assumed that the GOP structure is not fixed [2].

In real-time transcoding, such changes of the GOP structure can not be foreseen. Therefore, the target number of bits to be assigned to a GOP that was shortened due to an upcoming scene change is too large. However, this is compensated since the bit budget is not entirely used for the GOP due to the shortening. Therefore, it is assumed that a scene change does not result in a large variation in bit rate or quality. Thus, in the transcoder for this project, the GOP structure is learned by examining picture headers in the previous GOP and store these characteristics in variables $N$, $N_p$ and $N_b$. However, these variables are not updated when a previously transcoded GOP was shortened due to a scene change.

5.2.2 Calculation of the local spatial activity of a macroblock

At step 4 in TM-5, the local spatial activity for each macroblock is calculated in order to apply the method of adaptive quantization. These calculations are based on spatial values and, as already mentioned in chapter [4] the transcoder in this project will operate entirely in the frequency domain. Therefore this step needs to be adapted or replaced to successfully apply TM-5 in this transcoder.

The solution for this is found in the literature [12], where the local spatial activity of a macroblock is estimated by the ratio between the original quantization parameter of the macroblock retrieved from the input stream and the average quantization parameter of the entire picture. This approach is based on the fact that the front-encoder, as depicted in the situation shown in figure [3.1]
also computes the original quantization parameter of a macroblock based on the local spatial activity of the macroblock to achieve adaptive quantization. According to the paper, step 4 of TM-5 can therefore be replaced by:

\[ mquant = Q_j \cdot \left( \frac{Q1}{Q_{1,avg}} \right) \]  

(5.17)

where \( mquant \) is the requantization parameter used to requantize the current macroblock, \( Q_j \) is the base requantization parameter computed in step 3 of TM-5, \( Q1 \) is the quantization parameter retrieved from the incoming bit stream which was originally computed by the front-end encoder and \( Q_{1,avg} \) is the average quantization parameter of the picture. Obviously, this solution can be applied in both spatial and frequency domain transcoders. Replacing the above formula with step 4 of TM-5 also reduces the computational complexity of TM-5 since step 4 of TM-5 requires the largest amount of computations.

5.2.3 Changing the target bit rate at run-time

An important requirement of the transcoder is that it should be possible to change the target bit rate at run-time with at most one frame delay, allowing for the real-time adaptation of a video to deal with both the long-term and short-term fluctuations of the available bandwidth in wireless networks. Standard TM-5 however, is developed to produce a video at a fixed pre-defined bit rate. Therefore, TM-5 needs to be adapted to allow for a bit rate change at arbitrary times.

The first method discussed to achieve this is to scale each picture according to the desired reduction ratio \( R_r \). Obviously, scaling each picture down with a fixed ratio \( R_r \) results in a video stream at a bit rate of \( R_r \cdot \text{input bit rate} \). Step 1 and 2 in TM-5 can therefore be replaced by:

\[ T_i = S \cdot R_r \]  

(5.18)

\[ T_p = S \cdot R_r \]  

(5.19)

\[ T_b = S \cdot R_r \]  

(5.20)

where \( S \) denotes the size in bits of the incoming picture and \( R_r = \frac{\text{target bit rate}}{\text{input bit rate}} \). The formulas show that the target bit budget assignment for each picture is only dependant on the value of \( R_r \). Now, when a new target bit rate is received by the transcoder, the current value of \( R_r \) changes accordingly. As such, the next picture to be transcoded receives a target bit size that corresponds to the new target bit rate.

The second method discussed to achieve adaptation of the target bit rate at run-time, is based on adapting the variables influencing the target bit size for each picture when a target bit rate change occurs. This adaptation is achieved by changing the value of a variable as if they were calculated based on the new target bit rate. The variables present in formulas 5.2 - 5.4 influencing the calculation of the target bit sizes when a bit rate change occurs are:

- \( R \); target bit size of the entire GOP to be transcoded.
5.2. ADAPTING TM-5 TO SATISFY THE REQUIRED REACTION TIME

- $X_i$, $X_p$ and $X_b$: relative complexity measure per picture type.
- target bit rate: target bit rate specifying the bit rate of the output video.

Obviously, it is trivial how to deal with the latter item. How to deal with variable $R$ depends on the moment when the bit rate change occurs. When the new target bit rate arrives during transcoding the last picture of a GOP, the bit rate change affects the next GOP entirely. Therefore, variable $R$ is correctly computed according to formula (5.21). At other times, variable $R$ needs to be changed since it is computed at the beginning of the GOP and is therefore based on the old target bit rate. Therefore, when the bit rate changes at these times, variable $R$ is changed as follows:

$$R = R \cdot \left( \frac{R_r}{\text{prev} \cdot R_r} \right) \quad (5.21)$$

where $\text{prev} \cdot R_r$ and $R_r$ denotes the target reduction ratio before and after the bit rate change, respectively. This computation has to be executed before the execution of step 2 in TM-5, only when a bit rate change has occurred. The second item in the list of variables influencing the computation of the bit allocation for a picture concerns the relatively complexity measures by means of variables $X_i$, $X_p$ and $X_b$. The relative complexity measures $X_i$, $X_p$ and $X_b$ are defined as the product of the generated number of bits and the average requantization parameters used to transcode a picture of the corresponding picture type. Now, when a new target bit rate change occurs, these variables are used to compute the bit allocation for the next picture. However, these variables are computed according to the old target bit rate before the target bit rate change was received. To compensate for this change, variables $X_i$, $X_p$, $X_b$ need to be changed as if they were computed at the new target bit rate. Therefore, an experiment was executed to determine how complexity measures change when computed at different bit rates. In this experiment, three movies were transcoded to four different bit rates. During transcoding, the values of variables $X_i$, $X_p$ and $X_b$ used when computing bit allocation for the pictures, were recorded. After transcoding, the values of the complexity measures were analyzed. It was concluded that the complexity measures change relatively little as a result of a change in bit rate. This is illustrated in figure 5.2, showing the average values of $X_i$, $X_p$ and $X_b$ at the four different bit rates. From the figure it is concluded that, on average there is a small difference between the complexity measures captured at different bit rates. This is further explained by the definition of a complexity measure: average quantization parameter to encode a picture multiplied with the size of the coded picture in bits. Obviously, increasing the average quantization parameter results in a smaller picture, and vice versa. Apparently, this is an almost linearly relation where the ratio of changing the average quantization parameter corresponds with the ratio of difference in picture size. Furthermore, the complexity measures are used in formulas (5.2 - 5.4) only relatively to each other when determining a target bit size for a picture. Due to this reason, and due to the small difference of the complexity at different bit rates, there is no need to change the complexity measures when a new target bit rate is received by the transcoder.

The two methods discussed above adapted TM-5 such that it allows for a target bit rate change at run-time. Next, the two methods are compared with each
other to judge which method is the best to apply in the bit rate control algorithm for the transcoder to be developed for this project. Therefore, a video encoded at a bit rate of 6 Mbit/s is transcoded to 4 Mbit/s. However, during transcoding a new target bit rate was received four times: at frame 50 a bit rate change to 3 Mbit/s, at frame 100 a bit rate change back to 4 Mbit/s, at frame 150 a bit rate change to 5 Mbit/s and finally at frame 200 a bit rate change back to 4 Mbit/s. The results of this experiment are shown in figures 5.3 and 5.4 for the first and second method, respectively. The figures show the reduction ratios achieved per frame. The graph in figure 5.3 shows that each picture is transcoded according to the current value of $R_r$. The small deviations, shown in the graph, are caused by the rate control step present in TM-5. The reduction ratios achieved by the second method, as depicted in figure 5.4, show a much larger deviation of the desired reduction ratio $R_r$. These large variations are caused by TM-5, which assigns relatively more bits to I-pictures than to P and B-pictures. Therefore, almost no reduction is achieved on I-pictures. As a result, almost all reduction is gained by reducing the size of the B-pictures. This is shown in the figure, where the pictures with a reduction ratio of more than average (as depicted by the red line) are mostly I- and P-pictures. The pictures below average are the B-pictures. From a bandwidth point of view, the first method would undoubtedly be the best choice. However, after comparing the perceived quality of the two transcoded videos it was immediately noticed that the video transcoded using the second method provided a much better picture quality. This was also confirmed by an objective image quality test by calculating the Peak Signal to Noise Ratio (PSNR) value for each frame. The results of this objective quality test is shown in figure 5.5. The red and blue lines in the figure show the PSNR values of pictures from the movies transcoded with using the first and second method of adapting TM-5, respectively. Note that the streams are transcoded using an open-loop transcoder, since a closed-loop transcoder was not available at that time. As described in section 3.2.2, videos
5.2. ADAPTING TM-5 TO SATISFY THE REQUIRED REACTION TIME

Figure 5.3: Achieved reduction ratio per picture using the first method.

Transcoded with an open-loop transcoder suffer from drift. This effect is clearly visible in the graph by the continuous quality degradation of the pictures during the GOP. However, the absence of drift correction does not justify the 2.6 dB in quality degradation, since drift correction only prevents the continuously degradation in quality. Hence, there must be an other reason for this quality degradation. This is found when further analyzing figure 5.3. As can be seen in the figure, the difference in objective image quality between the I-frames is large. This is a direct consequence of the bit size assignment of TM-5, where I-pictures are assigned a larger number of bits compared with B-pictures. Obviously, assigning relatively more bits to I-pictures results in a higher objective image quality. However, it is also shown that assigning more bits to I-pictures has a positive effect on the image quality of all the pictures in the GOP. This is explained by the fact that an I-picture is used as a reference frame by all inter frames in the GOP, either directly or indirectly. Hence, from the above it can be concluded that the target bit budget can not be computed by simply scaling the size of an input picture with ratio $R_r$, while preserving good picture quality.

Since it is important to achieve a high picture quality perceived by the viewer, the second method is the best choice to adapt TM-5 such that the target bit rate can be changed at run-time. The characteristic that the reduction on I-frames is only minimal, and most of the reduction is achieved by reducing the size of B-frames is expected not to endanger the total system. This is because the transcoded video stream is buffered by the network sender before it is transmitted over the wireless network. Therefore, only the bit rate averaged over a few frames is of importance, which is achieved by the second method as shown in figure 5.4.
CHAPTER 5. BIT RATE CONTROL

Figure 5.4: Achieved reduction ratio per picture using the second method.

5.3 Bit rate control for the enhancement layers

In the previous section it is discussed how to adapt TM-5, such that the real-time requirements concerning the base layer are met by the transcoder. Unfortunately, due to the lack of time, thorough research on how to adapt TM-5 to create a bit rate control algorithm for the enhancement layers could not be completed. However, it is expected that a bit rate control algorithm for the enhancement layers can be developed using the same approach of rate control as used in TM-5 by means of step three of the algorithm. Therefore, it is expected that most effort concerning the development of the bit rate control is required to invent a proper way of determining the bit allocation for the pictures in the enhancement layer.
Figure 5.5: Objective quality measurement of a video transcoded using both methods of adapting TM-5.
Chapter 6

Proposed architecture

This chapter proposes the architecture of the transcoder as a solution for the problem of transporting high quality MPEG-2 video through a wireless link, as described in chapter 1. The proposed architecture is depicted in figure 6.1 and is a result of the choices made in the previous chapters. As depicted in the figure, the base layer is generated with a closed-loop transcoder (section 3.2.3) using the technique of motion compensation in the DCT domain (chapter 4). As described in section 3.1, an enhancement layer carries the requantization errors, defined as the difference between the inverse quantized DCT coefficients before and after requantization, of the previous layer. The bit rate control algorithm, as described in chapter 5, assures that the target bit rate for each layer will be met.

The next sections discuss whether the most important requirements that should be satisfied by the transcoder are met.

Figure 6.1: Proposed architecture of the scalable transcoder.
6.1 Real-time performance

An important requirement of the transcoder is that it should transcode non-scalable MPEG-2 video to SNR scalable video, in real-time. Unfortunately, the prototype implementation of this architecture could not be completed before the end of this project. As such, no accurate statement can be made whether the architecture as proposed above can satisfy this requirement on the PC platform. However, an estimation can be made based on the available open-loop transcoder that was realized during this project.

This open-loop transcoder is implemented using the programming language C, and is able to transcode a video to a non-scalable video (i.e. base layer only) at an arbitrary bit rate by means of the bit rate control algorithm described in chapter 5. Experiments carried out on a standard PC (e.g. Intel Pentium 4 2.4 GHz CPU, 512 MB) running Windows XP showed that transcoding a video to a non-scalable video is three times faster than real-time. That is, the transcoder is able to transcode 1 minute of video encoded at 6 Mbit/s to a non-scalable video at any bit rate within approximately 20 seconds. The transcoder is able to produce enhancement layers as well, but in an uncontrolled way and is therefore not representative for the architecture above. Transcoding the same video as in the previous experiment to scalable video, consisting of one base layer and two enhancement layers, is achieved in approximately 56 seconds. Hence, based on the time figures above, it is expected that the open-loop transcoder should be able to satisfy the requirements concerning the real-time performance.

Based on the experiments performed with the open-loop transcoder, it is expected that there is enough time left to perform motion compensation in the DCT domain while still satisfying the real-time requirements when transcoding to a non-scalable video. For the scalable video, it is expected that real-time requirements can not be satisfied based on this implementation at this current state.

6.2 Run-time adaptation of the target bit rate

Another important requirement, expressed by requirement FR.2 as listed in table 1.1, is that it should be able to change the target bit rate for each layer at run-time. Furthermore, requirement FR.2 also states that the next frame after receiving a new target bit rate should already satisfy a newly received target bit rate. These requirements are satisfied with the bit rate control algorithm described in chapter 5. This is shown by figure 6.2. The figure shows the actual reduction ratios of the frames from a transcoded stream when temporarily transcoding a 5 Mbit/s video to 2 MBit/s. It is shown that the target bit rate can be changed during transcoding and that the reaction time of one frame is achieved.

6.3 Achievable picture quality

As already mentioned, implementing the proposed architecture could not be finished within the time budget of this project. Therefore, judgements whether
6.4. IMPLEMENTATION

Figure 6.2: Temporarily transcoding to 2 MB/s demonstrating the reaction time of one frame of the bit rate control algorithm.

NFR.2 as listed in table 1.1 can not be made yet. It is expected that this requirement can be satisfied by the proposed solution. This is based on the picture quality that can be achieved by the open-loop transcoder realized during this project. During the project, a 6 Mbit/s video was transcoded to 6 Mbit/s, as stated by NFR.2. An objective image quality test was performed on the output stream. The results of the test were that the transcoded stream had degraded in quality with approximately 1.6 dB due to the open-loop transcoding. Since a closed-loop transcoder should be able to achieve a higher picture quality, it is expected that the proposes solution can satisfy NFR.2.

Other judgements about the picture quality that can be achieved by this proposed architecture can not be made at this time.

6.4 Implementation

At the beginning of the project, the aim was to develop and implement the transcoder from scratch. Soon, it became apparent that this could not be achieved within the time budget of the project. Therefore, the choice was made to use existing software by means of both a reference MPEG-2 decoder and encoder, which were implemented in the C programming language. Using parts from this reference software, an open-loop transcoder as described in section 3.2.2 was realized. In general, parts of the decoder were used to parse the incoming MPEG-2 video stream and decode the video blocks from the stream. Parts of the encoder were used to requantize these blocks and to correctly generate the output MPEG-2 video stream.
Next, this implementation was enhanced with the bit rate control algorithm, as described in chapter 5. As a result, it was possible to transcode a MPEG-2 video file to another MPEG-2 video file, where the target bit rate could be changed at run-time at arbitrary times. Unfortunately, the motion compensation loop for realizing a closed-loop transcoder and the generation of the enhancement layers could not be implemented in time.
Chapter 7

Conclusions

In this thesis, the theoretical foundations for implementing a real-time SNR scalable transcoder for MPEG-2 video streams were discussed.

The proposed architecture consists of the closed-loop transcoder to generate the base layer, with the addition of generating the enhancement layers. The closed-loop transcoder was chosen as a compromise between the picture quality achievable by the cascaded-pixel domain transcoder and the performance of the open-loop transcoder. The enhancement layers are generated according to an adapted form of SNR scalability. As a result, frames of an enhancement layer can be easily dropped.

To increase the performance of the proposed architecture, research was executed on the subject of motion compensation in the frequency domain. It was concluded that performing motion compensation in the frequency domain can save up to 81% compared to its spatial domain equivalent. Unfortunately, this method can not be implemented directly due to incomplete documentation. Another method found in the literature was claimed to achieve a reduction of 59% in computational complexity. In contrast with the first method, this method was clearly documented and can be implemented directly. Therefore, this method was chosen as the best candidate for implementation in the transcoder. However, the mentioned reduction only holds when performing motion compensation on P-pictures. For B-pictures, the amount of reduction is considerably lower since the entire operation of motion compensation needs to be performed twice.

Next, several bit rate control algorithms for the base layer were investigated. Unfortunately, none of them could satisfy the requirements for this transcoder. Therefore, a widely used bit rate control algorithm, TM-5, was adapted to enable the change of the target bit rate at run-time and to satisfy the reaction time of one frame. It was shown that these requirements are satisfied. Bit rate control for the enhancement layers could not be developed due to the lack of time.

Besides the research topics mentioned above, a lot of effort was spend on specifying, designing and implementing the proposed architecture. Unfortunately, this architecture could not be completed within the time budget of the project.
However, an implementation of the open-loop transcoding architecture was realized. Furthermore, this implementation was enhanced with the bit rate control algorithm developed during the research phase. With this, it was shown that the requirements concerning the bit rate control for the base layer could be satisfied.
Chapter 8

Recommendations and future work

It may be apparent that a lot of work still needs to be done to realize a prototype of the proposed solutions. The next sections contain recommendations and mention the future work to be done.

8.1 Enhancement layers

As already mentioned, the current implementation already supports the generation of enhancement layers. However, this was not further investigated. I recommend to first examine the functions to generate the enhancement layers thoroughly. Next, bit rate control for the enhancement layers should be implemented. In section 5.3 it is described that it is very likely that the bit rate control for the base layer can be used for the enhancement layers as well. Maybe some minor adaptations are necessary. When application of this bit rate control is unsatisfactory, a research study about a new method of bit rate control should be performed.

8.2 Motion compensation loop

The next logical step is the addition of the motion compensation loop to realize the closed-loop transcoder, to prevent drift in the output video. Before this however, I suggest to take a look at the Selective Error bit rate control scheme. It is an improvement of TM-5 with the aim of reducing the effect of drift in the output video. Perhaps, this method prevents the need for the motion compensation loop. If the achieved picture quality is insufficient, the choice can be made to implement the motion compensation loop. Then, the choice needs to be made between implementing it in the frequency domain or in the spatial domain. For execution on the PC platform, I recommend to implement the frequency domain equivalent. In the case of execution on the TriMedia platform, I recommend to implement the motion compensation loop in the spatial domain. This is based on the assumption about the availability of optimized versions, either software or hardware, of the DCT and IDCT operations on the TriMedia
platform. If so, there is no need to apply motion compensation in the frequency domain.

8.3 Improvement of picture quality

Besides enhancing the implementation with a motion compensation loop, there is an additional method of increasing the picture quality that can be achieved by the transcoder. This method [14] aims to reduce the additional error that exists due to the cascading of two quantization operations as depicted in figure 3.1. In this figure, quantization operations are executed in the encoder and the transcoder successively. To achieve bit rate reduction, the quantization in the transcoder is coarser than in the encoder. Ideally, applying the cascaded quantization should be equivalent to applying the coarser quantization immediately. As shown in figure 8.1 situations may occur where the results of cascaded quantization and direct quantization differ. This obviously results in a decrease in picture quality. Preventing this mismatch would result in an increase in picture quality.

Figure 8.1: Cascaded quantization (dashed arrows) compared with direct quantization with the coarser quantizer (solid arrow) (a) Cascaded quantization yields the same result as direct quantization. (b) Cascaded quantization yields a different result as direct quantization.
8.4 Increasing performance

Based on the experiments performed with the open-loop transcoder, it is expected that the performance of the proposed solution is insufficient to transcode video to scalable video on the PC platform in real-time using a standard PC. Therefore, it is expected that the performance of the transcoder needs to be increased. This can be achieved by:

- Further optimization of the code and compiling it using different optimization flags.
- Introducing concurrency in the implementation such that computationally complex and independent operations are executed simultaneously. These operations include operations like quantization and variable length coding.
- Pipelining the dataflow in the transcoder. For example, parsing and decoding the next frame can be performed while requantizing the DCT coefficients and generating the output stream.
- Updating the PC platform to today’s standards (e.g. faster CPU and corresponding faster front-side bus).

The actions above hopefully result in a scalable transcoder fast enough to satisfy the corresponding real-time requirements. However, when the actions above prove not to be sufficient, the choice can be made to perform motion compensation only on the P-frames in the stream to achieve a more drastic increase in performance. Applying this method prevents the occurrence of drift in the P-frames and therefore also the continuous degradation during the GOP. It is expected that this will greatly improve the performance of the transcoder while preserving good picture quality.

\[1\] 2.4 GHz Intel Pentium 4 CPU, 512 MB internal memory
Abbreviations

Acronyms and abbreviations used in this document are listed in table below.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DCT-FS</td>
<td>DCT domain Frame Store</td>
</tr>
<tr>
<td>DCT-MCP</td>
<td>DCT domain Motion Compensation Prediction</td>
</tr>
<tr>
<td>FR</td>
<td>Functional Requirement</td>
</tr>
<tr>
<td>FS</td>
<td>Frame Store</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Pictures</td>
</tr>
<tr>
<td>HDTV</td>
<td>High Definition Television</td>
</tr>
<tr>
<td>IDCT</td>
<td>Inverse Discrete Cosine Transform</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institution of Electronic and Electric Engineers</td>
</tr>
<tr>
<td>IPA</td>
<td>Information Processing Architectures</td>
</tr>
<tr>
<td>IQ</td>
<td>Inverse Quantizer</td>
</tr>
<tr>
<td>KISS</td>
<td>Keen Integration of Sub-Systems</td>
</tr>
<tr>
<td>MCP</td>
<td>Motion Compensation Prediction</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>NFR</td>
<td>Non-functional requirement</td>
</tr>
<tr>
<td>NTSC</td>
<td>National Television Systems Committee</td>
</tr>
<tr>
<td>PAL</td>
<td>Phase Alternating Line</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PLD</td>
<td>Piecewise Linearly Decreasing</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>Q</td>
<td>Quantizer</td>
</tr>
<tr>
<td>Qos</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SDTV</td>
<td>Standard Definition Television</td>
</tr>
<tr>
<td>SECAM</td>
<td>Séquentiel couleur à mémoire</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>VLC</td>
<td>Variable Length Coder</td>
</tr>
<tr>
<td>VLD</td>
<td>Variable Length Decoder</td>
</tr>
</tbody>
</table>

Table 8.1: Definitions, acronyms and abbreviations
Appendix A

Factorizing shifting matrices

In this appendix, the factorized matrices are displayed together with the operations they perform. These matrices are meant to change the data stored by an input matrix, further referred to as X. When applying normal matrix multiplication X is post-multiplied with one of the factorized matrices. The result is stored in another matrix Y, which is then used in future calculations. In stead, the data stored in X can be mutated by simple operations to gain the same result as Y. These simple operations are defined by each factorized matrix and are displayed below each matrix. Note that this prevents the use of temporary matrix Y, since X is mutated and can be directly used in future calculations.

The notation is as follows: Matrix X has 8x8 entries referred to as $x_{ij}$, where $i$ and $j$ denote the rows and columns of the matrix, respectively.

Matrix $D$

$$D = \begin{pmatrix}
0.3536 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.2549 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.2706 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3007 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.3536 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.4500 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.6533 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2814
\end{pmatrix}$$

Alternative operations:

$x_{11} = x_{11} \cdot 0.3536$
$x_{12} = x_{12} \cdot 0.2549$
$x_{13} = x_{13} \cdot 0.2706$
$x_{14} = x_{14} \cdot 0.3007$
$x_{15} = x_{15} \cdot 0.3536$
$x_{16} = x_{16} \cdot 0.4500$
$x_{17} = x_{17} \cdot 0.6533$
$x_{18} = x_{18} \cdot 1.2814$
Matrix $P$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Alternative operations:

\begin{align*}
    x_{i2} &= x_{i5} \\
    x_{i4} &= x_{i7} \\
    x_{i5} &= x_{i6} \\
    x_{i6} &= x_{i2} \\
    x_{i7} &= x_{i8} \\
    x_{i8} &= x_{i4}
\end{align*}

Matrix $B_1$

$$B_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Alternative operations:

\begin{align*}
    x_{i5} &= x_{i5} - x_{i8} \\
    x_{i6} &= x_{i6} + x_{i7} \\
    x_{i7} &= x_{i6} - x_{i7} \\
    x_{i8} &= x_{i5} + x_{i8}
\end{align*}

Matrix $B_2$

$$B_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$
Alternative operations:
\[ x_{i3} = x_{i3} - x_{i4} \]
\[ x_{i4} = x_{i3} + x_{i4} \]
\[ x_{i6} = x_{i6} - x_{i8} \]
\[ x_{i8} = x_{i6} + x_{i8} \]

Matrix \( M \)
\[
M = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.7071 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -0.9239 & 0 & -0.3827 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.7071 & 0 & 0 \\
0 & 0 & 0 & -0.3827 & 0 & 0.9239 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

Alternative operations:
\[ x_{i2} = x_{i2} \cdot 0.7071 \]
\[ x_{i5} = (x_{i5} \cdot -0.9239) + (x_{i7} \cdot -0.3827) \]
\[ x_{i6} = x_{i6} \cdot 0.7071 \]
\[ x_{i7} = (x_{i5} \cdot -0.3827) + (x_{i7} \cdot 0.9239) \]

Matrix \( A_1 \)
\[
A_1 = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

Alternative operations:
\[ x_{i1} = x_{i1} + x_{i2} \]
\[ x_{i2} = x_{i1} - x_{i2} \]
\[ x_{i4} = x_{i3} + x_{i4} \]
Matrix $A_2$

$$A_2 = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}$$

Alternative operations:

\begin{align*}
    x_{i1} &= x_{i1} + x_{i4} \\
    x_{i2} &= x_{i2} + x_{i3} \\
    x_{i3} &= x_{i2} - x_{i3} \\
    x_{i4} &= x_{i1} - x_{i4} \\
    x_{i5} &= -x_{i5} \\
    x_{i6} &= x_{i6} - x_{i5} \\
    x_{i7} &= x_{i6} + x_{i7} \\
    x_{i8} &= x_{i8} + x_{i7}
\end{align*}

Matrix $A_3$

$$A_3 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & -1
\end{pmatrix}$$

Alternative operations:

\begin{align*}
    x_{i1} &= x_{i1} + x_{i8} \\
    x_{i2} &= x_{i2} + x_{i7} \\
    x_{i3} &= x_{i3} + x_{i6} \\
    x_{i4} &= x_{i4} + x_{i5} \\
    x_{i5} &= x_{i4} - x_{i5} \\
    x_{i6} &= x_{i3} - x_{i6} \\
    x_{i7} &= x_{i2} - x_{i7} \\
    x_{i8} &= x_{i1} - x_{i8}
\end{align*}

(8.1)
Appendix B

Bit rate control algorithm based on the Piecewise Linearly Decreasing model

The front-end encoder quantized the video blocks with quantization parameter $Q_1$. In the transcoder this coded video is inverse quantized with quantization parameter $Q_1$ acquired from the coded video stream. Next, bit rate reduction is applied by requantizing the video blocks with a quantization parameter $Q_2$ which is larger than $Q_1$. It is shown that with a given quantizer parameter $Q_1$, the bit rate of the outgoing stream rapidly decreases at a given value of $Q_2$ regardless of the content of the video. The reason for this behavior is that the input bit stream, quantized by using the quantization parameter $Q_1$ and then coupled to the transcoder, is dequantized before being requantized and there exists a sufficiently large value of $Q_2$ capable of setting the dequantized value to zero. DCT coefficients having an identical weighting factor in the quantization matrix can all become zero after quantizing with a sufficiently large $Q_2$. This causes the rapidly decrease of the resulting bit rate.

These bit rate reduction characteristics of the requantization can be modeled as the Piecewise Linearly Decreasing model (PLD) as shown in figure 8.2. In this figure, the vertical axis shows the normalized bit rate value ratio $R_r = \frac{R_2}{R_1}$ where $R_1$ is the incoming bit rate and $R_2$ the outgoing bit rate. The horizontal axis represents the value of $Q_2$ which is the requantization parameter. The figure shows there are two points at which the bit rate rapidly decreases. However, the second point may not always exist. This depends on the value of $Q_1$.

The algorithm is described with the following six steps:

1. **Initialization**: calculate the normalized bit rate $R_r = \frac{R_2}{R_1}$ and the target bit rate $B_T = \text{current slice bit} \cdot R_r$. The value of `current slice bit` is obtained by counting the bit number of the slice fed to the transcoder.

2. **Calculate $Q_2$**: Depending on the value of $R_r$, $Q_2$ is calculated according
Figure 8.2: Typical curve for bit rate reduction characteristics.

to one of the following equations:

\[
R_r = \frac{1 - \alpha}{Q_1 - Q_{d1}}(Q_2 - Q_1) + 1 \quad \text{for } \alpha \leq R_r < 1
\]

\[
R_r = \frac{\alpha - \beta}{Q_{d1} - Q_{f1}}(Q_2 - Q_{d1}) + \alpha \quad \text{for } \beta \leq R_r < \alpha
\]

\[
R_r = \frac{\beta - \gamma}{Q_{f1} - Q_{d2}}(Q_2 - Q_{f1}) + \beta \quad \text{for } \gamma \leq R_r < \beta
\]

\[
R_r = \frac{\gamma - \delta}{Q_{d2} - Q_{f2}}(Q_2 - Q_{d2}) + \gamma \quad \text{for } \delta \leq R_r < \gamma
\]

\[
R_r = \frac{\delta - 0.2}{Q_{f2} - 31}(Q_2 - Q_{f2}) + \delta \quad \text{for } R_r < \delta
\]

If \( \beta \leq R_r < \alpha \) then proceed to step 3 otherwise proceed to step 5.

3. **Calculate number of**, \( Q_{d1} \) **and number of**, \( Q_{f1} \): These values are calculated according to the following formulas in order to provide a appropriate distribution.

\[
\text{number of} \ Q_{d1} = \text{nint} \left( \frac{M}{\alpha - \beta} \cdot (R_r - \delta) \right)
\]

\[
\text{number of} \ Q_{f1} = M - \text{number of} \ Q_{d1}
\]

Proceed to step 4.

4. **Determine mixing pattern**: Determine the mixing pattern of \( Q_{d1} \) and \( Q_{f1} \) for consistent picture quality according to the values calculated in step 3.

5. **Requantize macroblocks**: Requantize each macroblock from this slice with the quantization parameter \( Q_{d1} \) determined in the previous steps. As a result, a bit amount \( B_G \) is obtained. If this step is finished the bit rate
control for one slice is practically completed. A bit amount $B_D$ deviated from the bit amount $B_T$ is computed by $B_D = B_T - B_G$. During the above processes being performed, a next slice is input to the transcoder and the bit number of the next slice $\text{next\_slice\_bit}$ is obtained at this time. Proceed to step 6.

6. **Update parameters:** The parameters for transcoding the next slice are updated by the following equations:

$$R_r = \frac{R_r \cdot \text{next\_slice\_bit} + B_D}{\text{next\_slice\_bit}}$$

$$B_T = \text{next\_slice\_bit} \cdot R_r$$

Then in order to transcode the next slice, the process is fed back to step 2.
Bibliography


