Scalable Transcoding of H.264 Video

Hussain Al-Muscati

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

May 2010

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Masters of Engineering.

© 2010 Hussain Al-Muscati
Abstract

Digital video transcoding provides a low complexity mechanism to convert a coded video stream from one compression standard to another. This conversion should be achieved while maintaining a high visual quality. The recent emergence and standardization of the scalable extension of the H.264 standard, together with the large availability of encoded H.264 single-layer content places great importance in developing a transcoding mechanism that converts from the single layer to the scalable form.

In this thesis, transcoding of a single layer H.264/AVC stream to H.264/SVC stream with combined spatial-temporal scalability is achieved through the use of a heterogeneous video transcoder in the pixel domain. This architecture is chosen as a compromise between complexity and reconstruction quality.

In this transcoder, the input H.264/AVC stream is fully decoded. The macroblock coding modes and partitioning decisions are reused to encode the output H.264/SVC stream. A set of new motion vectors is computed from the input stream coded motion vectors. This extracted and modified information is collectively downsampled, together with the decoded frames, in order to provide multiple scalable layers. The newly computed motion vectors are further subjected to a 3 pixel refinement. The output stream is coded with either a hierarchical B-frame or a zero-delay referencing structure.

The performance of the proposed transcoder is validated through simulation results. These simulations compare both the compression efficiency (PSNR/bit-rate) and computational complexity (computation time) of the implemented transcoding scheme to a setup that preforms a full decoding followed by a full encoding of the incoming video stream. It is shown that a significant decrease in computational complexity is achieved with a reduction of over 60% in some cases, while maintaining a small loss in compression efficiency.
Sommaire

Le transcodage vidéo numérique fournit un mécanisme de faible complexité pour convertir un flux vidéo d’un format de compression à un autre. Cette conversion devrait être atteinte tout en maintenant une haute qualité visuelle. La récente émergence et la normalisation de l’extension “scalable” (en couches) de la norme H.264, ainsi que la grande disponibilité de contenu codé au format H.264 à couche unique donnent une grande importance au développement d’un mécanisme de transcodage qui convertit du format à couche unique à la forme “scalable”.

Dans cette thèse, le transcodage d’un flux simple couche H.264/AVC vers un flux H.264/SVC combinant des couches spatiales et temporelles est obtenue par l’utilisation d’un transcodeur vidéo hétérogène dans le domaine des pixels. Cette architecture est choisie comme un compromis entre la complexité et la qualité de reconstruction.

Dans ce transcodeur, le flux d’entrée H.264/AVC est entièrement décodé. Le mode de codage et les décisions de partitionnement pour les macro-blocs sont réutilisés pour encoder le flux de sortie H.264/SVC. Un ensemble de nouveaux vecteurs de mouvement est calculé à partir des vecteurs de mouvement du flux d’entrée codé. Cette information modifiée est sous-échantillonnée, en même temps que les images décodées, afin de fournir de multiples couches spatiales. Les vecteurs de mouvement nouvellement calculé sont en outre soumis à un raffinement de 3 pixels. Le flux de sortie est codé soit avec soit un système de dimages B hiérarchique soit avec une structure à délai zéro.

La performance du transcodeur proposé est validée par les résultats de simulation. Ces simulations comparent à la fois l’efficacité de compression (PSNR/débit), et la complexité des calculs (temps de calcul) du système de transcodage à un système qui met en uvre un décodage complet suivi d’un ré-encoder complet du flux vidéo entrait. Il est démontré qu’une diminution significative de la complexité algorithmique est atteinte avec une réduction de plus de 60% dans certains cas, tout en maintenant une faible perte en efficacité de compression.
Acknowledgments

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Fabrice Labeau, who has supported me throughout the course of my graduate studies with his patience, knowledge, guidance, and financial support. I specially thank him for his constructive feedback in reviewing this thesis.

I would also like to extend my gratitude to all my fellow graduate students in the TSP lab for their support and assistance.

Finally and most importantly, I would like give special thanks to my parents for their continuous love, support, and encouragement. I would also like to thank my sister for her continuous love, and support.
# Contents

1 Introduction .............................................. 1
   1.1 Overview ........................................... 1
   1.2 Problem Formulation .................................. 2
   1.3 Our Contribution .................................... 3
   1.4 Thesis Description and Organization .................. 4

2 Transcoding Background ................................. 6
   2.1 Introduction ......................................... 6
   2.2 Generic Video CODEC .................................. 7
   2.3 Transcoding Requirements ............................. 8
   2.4 Transcoding Architectures ............................ 9
      2.4.1 Open-Loop Transcoding ......................... 10
      2.4.2 Closed-Loop Transcoding ....................... 13
      2.4.3 Pixel-Domain Transcoding ..................... 14
      2.4.4 Transform-Domain Transcoding ................. 16
   2.5 Homogeneous Transcoding ............................ 19
      2.5.1 Bit-Rate Reduction ............................. 20
      2.5.2 Spatial Resolution Reduction ................. 21
      2.5.3 Temporal Resolution Reduction ................ 25
   2.6 Heterogeneous Transcoding .......................... 26
   2.7 Summary ............................................ 28

3 The H.264 Video Compression Standard ................. 29
   3.1 The H.264/Advanced Video Coding Standard .......... 29
      3.1.1 H.264 Codec Overview .......................... 30
3.1.2 H.264 Standard Details ........................................... 31
3.1.3 H.264 High Level Design ........................................... 39
3.2 The Scalable Video Coding Extension of the H.264/AVC Standard ... 39
  3.2.1 Scalability Overview ........................................... 40
  3.2.2 SVC High Level Design ........................................... 44
3.3 Transcoding from H.264/AVC to H.264/SVC .......................... 45
3.4 Summary ................................................................. 47

4 Temporal Transcoding of H.264/AVC to H.264/SVC ........................ 48
  4.1 Time Profiling of Reference Software ................................ 49
    4.1.1 Profiling Method ........................................... 49
    4.1.2 Profiling Results ........................................... 50
  4.2 Proposed Temporal Transcoder ..................................... 52
    4.2.1 Transcoder Architecture ..................................... 52
    4.2.2 Frame Referencing Structure ................................ 53
    4.2.3 Motion Vector Re-estimation ................................ 59
  4.3 Simulation Setup and Results ..................................... 61
    4.3.1 Simulation Setup ........................................... 61
    4.3.2 Simulation Results ........................................... 63
  4.4 Summary ................................................................. 74

5 Combined Spatial-Temporal Transcoding of H.264/AVC to H.264/SVC ..................... 78
  5.1 Time Profiling of Reference Software ................................ 79
    5.1.1 Profiling Method ........................................... 79
    5.1.2 Profiling Results ........................................... 80
  5.2 Proposed Spatial-Temporal Transcoder ................................ 81
    5.2.1 Transcoder Architecture ..................................... 81
    5.2.2 Downsampler .................................................. 86
  5.3 Simulation Setup and Results ..................................... 92
    5.3.1 Simulation Setup ........................................... 92
    5.3.2 Simulation Results ........................................... 93
  5.4 Summary ................................................................. 99
Contents vi

6 Conclusion 101
   6.1 Summary ........................................ 101
   6.2 Future Work .................................... 102

References 103
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A cascaded decoder-encoder setup.</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Generic open-loop transcoder.</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Generic closed-loop transcoder.</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>Cascaded decoder-encoder.</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>Pixel-Domain transcoder.</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td>Transform-Domain transcoder.</td>
<td>18</td>
</tr>
<tr>
<td>2.7</td>
<td>Motion Compensation (MC).</td>
<td>19</td>
</tr>
<tr>
<td>2.8</td>
<td>Four macroblock modes downsampled to one.</td>
<td>24</td>
</tr>
<tr>
<td>2.9</td>
<td>Four motion vectors downsampled to one.</td>
<td>24</td>
</tr>
<tr>
<td>2.10</td>
<td>Motion vector adjustment.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>H.264/AVC Encoder.</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>H.264/AVC Decoder.</td>
<td>32</td>
</tr>
<tr>
<td>3.3</td>
<td>Macroblock partitions.</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Sub-Macroblock partitions.</td>
<td>36</td>
</tr>
<tr>
<td>3.5</td>
<td>Coding with a hierarchical B-frame structure.</td>
<td>41</td>
</tr>
<tr>
<td>3.6</td>
<td>Coding with a zero-delay structure.</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Breakdown of computation time for different encoding process.</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Breakdown of computation time for different encoding prediction process.</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Detailed schematic of H.264/AVC to H.264/SVC temporal transcoder.</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>SVC hierarchical B-frame structure.</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>SVC hierarchical zero-delay structure.</td>
<td>57</td>
</tr>
<tr>
<td>4.6</td>
<td>AVC IPPP referencing structure.</td>
<td>57</td>
</tr>
</tbody>
</table>
4.7 An 8×16 block area overlapping with 4 other macroblocks of different partitioning sizes. ........................................ 59
4.8 Example of the forwards dominant selection (FDVS) method. ............ 60
4.9 Macroblock and motion vector decomposition, evaluation, and combining. 61
4.10 Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical B-frames structure with a GOP size of 8. ................................. 66
4.11 Plot of PSNR/Bitrate of CIF Foreman for a hierarchical B-frames structure with a GOP size of 8. ................................. 66
4.12 Plot of PSNR/Bitrate of QCIF NBA for a hierarchical B-frames structure with a GOP size of 8. ................................. 67
4.13 Plot of PSNR/Bitrate of CIF NBA for a hierarchical B-frames structure with a GOP size of 8. ................................. 67
4.14 Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical B-frames structure with a GOP size of 16. ................................. 69
4.15 Plot of PSNR/Bitrate of QCIF NBA for a hierarchical B-frames structure with a GOP size of 16. ................................. 69
4.16 Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical zero-delay structure with a GOP size of 8. ................................. 70
4.17 Plot of PSNR/Bitrate of CIF Foreman for a hierarchical zero-delay structure with a GOP size of 8. ................................. 70
4.18 Plot of PSNR/Bitrate of QCIF NBA for a hierarchical zero-delay structure with a GOP size of 8. ................................. 71
4.19 Plot of PSNR/Bitrate of CIF NBA for a hierarchical zero-delay structure with a GOP size of 8. ................................. 71
4.20 Average computation time for QCIF streams encoded with a hierarchical B-frames structure. ........................................ 76
4.21 Average computation time for QCIF streams encoded with a hierarchical zero-delay structure. ........................................ 76
4.22 Average computation time for CIF streams encoded with a hierarchical B-frames structure. ........................................ 77
4.23 Average computation time for CIF streams encoded with a hierarchical zero-delay structure. ........................................ 77
List of Figures

5.1 Breakdown of computation time for different encoding process, with no inter-layer prediction (Mode 0) ........................................... 82
5.2 Breakdown of computation time for different encoding process, with inter-layer prediction (Mode 1) ........................................... 82
5.3 Comparing computation total computation time of base and enhancement layer, with no inter-layer prediction (Mode 0) ................ 83
5.4 Comparing computation total computation time of base and enhancement layer, with inter-layer prediction (Mode 1) ..................... 83
5.5 Breakdown of computation time for different encoding prediction processes, with no inter-layer prediction (Mode 0) ................. 84
5.6 Breakdown of computation time for different encoding prediction processes, with inter-layer prediction (Mode 1) ......................... 84
5.7 Detailed schematic of the H.264/AVC to H.264/SVC combined spatial-temporal transcoder ................................................................. 87
5.8 Four macroblock modes downsampled to one. .............................. 88
5.9 Plot of PSNR/Bitrate of Foreman for a hierarchical B-frames structure. . 96
5.10 Plot of PSNR/Bitrate of NBA for a hierarchical B-frames structure. .... 96
5.11 Plot of PSNR/Bitrate of Foreman for a hierarchical zero-delay structure. 97
5.12 Plot of PSNR/Bitrate of NBA for a hierarchical zero-delay structure. ... 97
5.13 Average computation time for streams encoded with a hierarchical B-frames structure .......................................................... 100
5.14 Average computation time for streams encoded with a hierarchical zero-delay structure ......................................................... 100
List of Tables

4.1 List 0 Motion Vector Mapping from AVC to SVC Stream. ............... 58
4.2 Different transcoding schemes and their properties. .................. 63
4.3 Decoder-Encoder percentage of motion vectors smaller than or equal to 3 pixels for QCIF. .................................................. 68
4.4 Decoder-Encoder percentage of motion vectors smaller than or equal to 3 pixels for CIF. ................................................. 68
4.5 Average coding time taken by different schemes for QCIF sequences encoded with a B-frame structure. .............................. 74
4.6 Average coding time taken by different schemes for CIF sequences encoded with a B-frame structure. .............................. 75
4.7 Average coding time taken by different schemes for QCIF sequences encoded with a zero-delay structure. ............................ 75
4.8 Average coding time taken by different schemes for CIF sequences encoded with a zero-delay structure. ............................ 75
5.1 Percentage of total MBs coded in each coding mode for an IPPP H.264/AVC sequence. .......................................................... 89
5.2 Downsampling of each macroblock partitioning. .......................... 90
List of Acronyms

CABAC  Context-based Adaptive Binary Arithmetic Coding
CAVLC  Context Adaptive Variable Length Coding
CIF    Common Intermediate Format
GOP    Group of Pictures
MB     Macroblock
MC     Motion Compensation
ME     Motion Estimation
MV     Motion Vector
NAL    Network Abstraction Layer
PSNR   Peak Signal-to-Noise Ratio
QCLF   Quarter Common Intermediate Format
VCL    Video Coding Layer
VLC    Variable Length Coding
VLD    Variable Length Decoding
Chapter 1

Introduction

1.1 Overview

In the last two decades, digital video has become an integral part of our daily lives. Today, digital video can be found on several different platforms, from digitally recorded video on compact disks to online video streaming. The requirement of providing high quality video with low bit-rate and delay has become an increasingly important demand. The vast movement of digital video from one platform to another requires that these video streams be stored in a compact format, such that to enable fast movement between platforms and to minimize the required storage space. Video compression (or coding) plays an important role in achieving this target. It reduces what can be described as bandwidth-intensive raw video to a manageable size, such that it is suitable for transmission or storage [1]. This reduction is achieved by removing any redundancy between different parts of a raw video sequence. The coding of a video sequence requires that this raw video is compressed before storage or transmission, and subsequently decompressed when viewed.

Generally, video coding standards exploit both temporal and spatial redundancy in a raw video sequence to achieve compression [1]. They make use of the high correlation between subsequently captured frame, and between different parts of the same frame. Different standards use similar methods to model this redundancy, such as block-based motion compensation [1].

The large variety of digital video applications, together with the different storage and playback requirements, has led to the development of several compression standards and
configurations. This broad range of video formats requires robust and computationally inexpensive mechanisms to convert between them. The most simple of methods is to completely decode a compressed video sequence to raw video, and then re-encode it to the compressed domain with the desired standard/configuration, using a cascaded decoder-encoder setup. Unfortunately, it has proven to be computationally expensive and time consuming due to the high complexity of the video encoding process [2].

Digital video transcoding provides a solution for this problem by converting a compressed video stream from one format to another. It makes use of coding parameters and statistics that are extracted from the existing compressed video stream to reduce the complexity of this conversion [2]. Several applications of video transcoding exist. A video transcoder can be used to alter the bit-rate and/or the resolution of a compressed video stream. It can also be used to convert a coded video stream from one compression standard to another.

1.2 Problem Formulation

Although different video compression standards rely on the same techniques to encode a raw video stream, there is still a considerable difference between them in terms of both syntax and structure. To improve compression efficiency, video standards have considerably evolved over the years, resulting in a vast range from simple MPEG-1 and H.261, to complex MPEG-4 and H.264 [3].

The development of video compression has taken a major step forward with the standardization of the H.264 standard. The standard has managed to provide a compression efficiency that is significantly higher than any other standard. It also provides a high transmission efficiency that supports reliable and robust transport over a range of networks [1]. The high compression efficiency achieved by H.264 has established it as the prevailing standard used for video compression today, such that a huge amount of video content is encoded in the H.264 format.

This recent standardization of the scalable extension of H.264 video compression standard, has placed a great importance on developing a mechanism in which an existing single layer H.264 video stream could be converted to the scalable form while incurring minimum delay. This scalable extension was established to provide partial transmission of fully encoded video streams. It was designed such that a scalable stream can be partitioned
into lower temporal and/or spatial and/or quality sub-partitions. This scalable extension achieves a significant improvement in compression efficiency over older scalable standards, with an increased degree of scalability as well. It is found to be most useful when used in networks and platforms that impose strict bit-rate limits. High quality scalable video streams could be easily partitioned to adhere to temporal and/or spatial and/or quality restrictions that might be imposed by such networks or platforms.

There is extensive work in the literature on transcoding between older video standards such as MPEG-1, MPEG-2, MPEG-4, and H.263. Nevertheless, the continuous evolution of video compression standards, together with the development of new techniques to increase their compression efficiency means that these transcoding mechanisms cannot be applied directly to the H.264 standard. They would require adaptation and modification to adhere to the H.264 coding properties.

Several contributions in the area of converting between H.264 video and other video standards can be found in the literature, such as transcoding from MPEG-4 to H.264 [3], and from H.264 to AVS [4]. Nevertheless, the existing literature on transcoding an H.264 compressed video stream from its single layer to its scalable form is at best minimal. Only a handful of papers deal with this problem, and only a few specific narrow approaches have been considered [5], [6], [7]. Hence, considerable work has yet to be done in this area, and much can be accomplished by developing new transcoding techniques and approaches to facilitate the transcoding of H.264 compressed video from its single layer to its scalable format.

1.3 Our Contribution

In this thesis, we present a novel implementation of a transcoder that provides scalability to an existing single layer H.264 video coded stream. Our contribution to this problem can be categorized into two parts.

The first part discusses the problem of providing temporal scalability to an existing single layer H.264 stream. Temporal scalability can be provided by adjusting the incoming H.264 coded video frame referencing structure, such that the frame rate of a coded stream can be reduced on the fly by simply dropping packets that correspond to some specific frames. This proposed mechanism makes use of some of the similarities between the single and scalable versions of the H.264 standard, together with the modification of certain
parameters from the input stream. Our proposed method has the flexibility to adhere to different structures of H.264 temporal scalability, and different levels of temporal scalability as well.

The second part discusses the problem of providing combined spatial-temporal scalability to an existing single layer H.264 stream. This combined spatial-temporal scalability is provided by extending the temporal transcoder discussed in the first part. Spatial scalability can be provided through the re-encoding of an incoming H.264 coded bitstream with multiple scalable resolutions, such that the spatial resolution of a coded bit stream can be modified on the fly through the dropping of packets that are associated with the high spatial resolution. This proposed mechanism makes use of some of the similarities between the single and scalable versions of the H.264 standard, together with the adjustment of certain information associated with the input stream, such that it can be used to encode multiple layers. Our proposed method has the flexibility to adhere to different combinations of H.264 combined spatial-temporal scalability, and different levels of both spatial and temporal scalability as well.

The results of this work show that our implemented techniques and methodologies provide both temporal and combined spatial-temporal scalability to an existing single layer H.264 video stream. It is also shown that the transcoding of an H.264 single layer stream to its scalable format can be achieved while maintaining a reasonable compression efficiency and considerably reducing the computational complexity of the process.

1.4 Thesis Description and Organization

In this thesis, we present a detailed discussion of our study and our results as follows: In Chapter 2 we present a general overview of digital video transcoding, together with a brief discussion on several transcoding architectures and functionalities. This general overview will help lay the foundation for a better understanding of the more particular details described in subsequent chapters.

In Chapter 3 we introduce the H.264 video coding standard. We present an overview of both its single layer and scalable extension. We also discuss some of the main issues involved in transcoding from single layer H.264 to the scalable form. This chapter provides the necessary technical background on the different aspects of the H.264 standard, and will help to provide a better understanding once we discuss some of the technical details that
1 Introduction

involve the transcoding of video streams that are coded with the H.264 standard.

Chapters 4 and 5 describe our proposed temporal transcoder and combined spatial-temporal transcoder, respectively. A detailed discussion is presented in each chapter, together with the necessary experimental setup and results to validate our study.

Finally, Chapter 6 concludes this thesis and presents some recommendations for future work that could be extended from this study.
Chapter 2

Transcoding Background

In this chapter, an overview of several transcoding architectures and functionalities is presented. We first present an overview of a generic video compression process. The basic set of requirement for transcoding are discussed. Several transcoding architectures are presented, with emphasis on comparing open-loop with closed-loop transcoding, and comparing pixel-domain to transform-domain transcoding. We categorize two different transcoding types in terms of functionality, homogenous and heterogenous transcoding. We present an overview of homogenous transcoding, as one of the most common transcoding approaches. Several methodologies for bit-rate, spatial-resolution, and temporal-resolution reduction are subsequently described. We also discuss some uses of heterogenous transcoding. This general overview of different transcoding architectures and functionalities will help lay the foundation for a better understanding of the more particular details described in subsequent chapters.

2.1 Introduction

There are several functionalities for video transcoding that exist. From a simple modification of a configuration within a given standard, to changing the coded video to a different format. A homogenous video transcoder performs a conversion between video bit streams of the same standard. It can be used to reduce the bit-rate of a high quality encoded video, to adhere to low bandwidth transmission requirements. Any dynamic change to the frame-rate or spatial-resolution of the encoded video stream could also be accomplished
with the use of a homogenous transcoder [2]. A heterogenous video transcoder converts between video bit streams of different standards, thereby changing them from one format to another, to better suit the playback platform. Additional information such as logos or watermarks could be inserted into the encoded video stream via the use of a transcoder.

In order to provide a better understanding of video compression, we present an overview of a generic video CODEC (enCOder/DECoder pair) in the next section.

### 2.2 Generic Video CODEC

A generic video CODEC makes use of common video coding methods such as motion estimation and motion compensation prediction, transform, quantization, and variable length coding. We present an overview of both the encoding and decoding processes.

#### Encoding

In order to compress a raw video stream, video compression mechanisms rely on the correlation between different parts of the video sequence. A generic encoder consists of two distinct data flow paths. A forward encoding path and a backward reconstruction path.

In the forward path, a prediction $P$ is constructed for each current processed frame $F$ by either inter or intra prediction. Inter prediction relies on the correlation between the current processed frame and previously encoded frames through the use of motion estimation and motion compensation [1]. Motion estimation (ME) estimates the relative motion between two or more video frames in order to find the best match, while motion compensation (MC) predicts a video frame with the modelling of motion [1]. Intra prediction on the other hand, relies on the correlation between different parts of the current processed frame. This prediction $P$ is subtracted from the current frame to form a prediction residue $R^1$.

The prediction residue $R$ is further subjected to a forward transform process, such that it is converted to the transform (frequency) domain. This forward transform results in forming a compact and de-correlated (separated into components with minimal inter-dependence) data form [1]. To further increase the compression efficiency, the transformed prediction residue is further processed by subjecting it to a quantization process that forms a set of quantized coefficients $QC$, such that it is represented by a smaller number of

---

$^1$To simplify the process, each frame is partitioned and processed in small macroblock units.
Finally, the quantized coefficients $QC$ are variable length coded to produce the output video stream. Variable length coding maps the quantized coefficients $QC$ to a series of codewords. Frequent coefficients are represented with short variable length codes, while less frequent coefficients are represented with long variable length codes. Over a sufficiently large number of encoded coefficients this leads to the compression of data [1].

In the backward reconstruction path, the quantized coefficients $QC$ are decoded in order to reconstruct each coded frame, such that the frame may be used for inter prediction for future frames [1]. The quantized coefficients $QC$ are inverse quantized and inverse transformed to reproduce the prediction residue $R'$\textsuperscript{2}. This prediction residue $R'$ is added to the prediction $P$ of the current frame, such that the current frame is reconstructed. Finally this reconstructed frame is stored in a reference picture buffer [1].

**Decoding**

The input compressed video stream is first processed by a variable length decoder to reproduce a set of quantized coefficients $QC$. These coefficients are further inverse quantized and inverse transformed to produce a prediction residue $R'$. The prediction $P$ is either constructed via inter prediction through MC of previously decoded reference frames, or through intra prediction of previously decoded parts from the current frame [1]. The prediction $P$ is added to the prediction residue $R'$, to reconstruct the current frame. The resulting frame is used to form a reference frame which is stored in a reference picture buffer [1].

In the next section, the basic set of requirement for transcoding are discussed.

**2.3 Transcoding Requirements**

The large variety of different video transcoding architectures and designs, together with the broad range of associated functionalities, give rise to the need to define a general set of requirements for a video transcoder. These requirements define design goals that need to be achieved by any video transcoder. The main goal of any transcoder is to convert from one video format to another while achieving maximum performance. There are two

\textsuperscript{2}Due to the quantization process, the reconstructed prediction residue $R'$ is not the same as the original prediction residue $R$. 
basic measures of performance in the context of video coding: the output visual quality\(^3\) of the video sequence and the computational complexity of the process. When designing any video transcoder, there exists a set of requirements that must be satisfied in order to optimize the transcoding process [8]:

1. Information embedded within the existing coded video stream should be exploited as much as possible.

2. The output of the transcoder should maintain a visual quality that is as high as possible. The output should also be as close as possible to a stream that is completely decoded and encoded again to the desired format (cascaded setup).

3. The computational complexity and delay incurred by a transcoding process should be reduced to a minimum.

These requirements provide a set of fundamental guidelines that are followed in the design of different transcoding architectures.

### 2.4 Transcoding Architectures

Transcoding architectures can be categorized along two different axes. One axis differentiates between open-loop and closed-loop systems, while the other differentiates between transcoding in the transform domain and the pixel domain. A cascaded decoder-encoder setup can be used as a basic reference to what a transcoder sets to achieve. It fully decodes an incoming video stream, followed by a full re-encoding to produce the desired output. A simple illustration of a cascaded setup can be seen in Fig. 2.1.

Since the decoded video stream is re-encoded with a lossy encoder, it introduces a degradation to the quality. This degradation compounds over the degradation associated with the original lossy encoding of the incoming compressed video. The probability that the degradation associated with the second lossy process occurs on the exact same parts as the first lossy encoding process is very small. This is due to the fact that errors occur in an encoded video sequence because of elements that are part of the encoding process, such

---

\(^3\)In the video coding context, the peak signal-to-noise ratio (PSNR) is used as measure of visual quality, while the PSNR/bit-rate ratio is used as a measure of the compression efficiency. In our discussion, we compare different video coding schemes by measuring the PSNR at a fixed bit-rate. Hence the terms visual quality and compression efficiency become interchangeable.
as block sizes, search, and quantization. When an encoded sequence undergoes a decoding followed by a re-encoding, the second encoding process introduces additional degradations. If the second encoding process is similar to the first, no significant additional degradation will be introduced, simply because the introduced error would be identical. If the second encoding is different, the additional degradation would unlikely occur at the same parts of the first degradation, simply because the second encoding would be using different encoding elements and methods. Hence, this setup is considered to be the optimal method in terms of minimizing any degradation of quality [8]. The main problem with such a setup is the high computational complexity and large delay associated with the full re-encoding process, making it an unrealistic and unpractical solution for most applications.

2.4.1 Open-Loop Transcoding

An open-loop transcoder performs a straightforward operation on an input video sequence. It transcodes an incoming video stream by directly modifying the coded video data, such that it is mapped from one format to another. It contains no feedback loop, thus requiring no memory for frame storage, which minimizes the processing delay [9].

A generic structure of an open-loop transcoder can be seen in Fig. 2.2. An incoming video stream is first variable length decoded to convert the compressed bitstream into a set of quantized coefficients. These quantized coefficients are subjected to a direct modification, followed by a variable length coding process to produce the output video stream. The structure of the modifier component highly depends on the desired transcoding operation. However, it is restricted in the sense that it can only perform direct operations that do not contain any feedback, such as re-quantization or the discarding of transform coefficients. A simple example of an open-loop transcoder is one that performs a homogenous functionality,
such as bit-rate modification through the adjustment of the quantization parameter. The modifier performs an inverse quantization on the coefficients followed by a re-quantization with the desired quantization parameter [9]. An open-loop transcoder could also be used for heterogeneous functionalities such as converting between different standards, specifically in the case when there is a large degree of similarity between the input and target video standards. An incoming stream could simply undergo a variable length decoding (VLD) operation followed by a variable length coding (VLC) operation. All the coding and motion information contained within the input stream is directly mapped to the output stream.

A major drawback of an open-loop structure is its large susceptibility to drift, which causes a degradation in picture quality [9]. Drift arises because video standards rely on predictive coding to compress a video stream. A prediction $P$ of the current video picture frame $F$ can be formed through the use of previously encoded and reconstructed frames. This prediction $P$ can be formed through inter prediction which relies on the correlation between the current frame $F$ and previously encoded frames. It could also be formed through intra prediction which relies on the correlation between different parts of the current frame $F$. Once the prediction $P$ is constructed, it is subtracted from the current frame $F$ and only the difference $R$ (prediction residue) is encoded as the output video stream, such that we have $R = F - P$.

A decoder reconstructs each video picture frame with the use of the incoming stream prediction residue $R'$. It also makes use of previously decoded and reconstructed frames to construct a prediction $P'$. The current frame is reconstructed by adding the decoded residue $R'$ to the prediction $P'$ \(^4\), such that we have $F' = R' + P'$. Because of the encoding and decoding quantization process, the decoded prediction residue $R'$ is not equal to the original prediction residue $R$. Hence the reconstructed frame $F'$ is not equal to the original frame $F$, and we have a resultant error between $F'$ and $F$, such that $e = F' - F$.

The open-loop transcoder modifies the encoded stream causing an added error in the transcoded frames such that the difference between $R$ and $R'$ is increased. That in turn causes a change in the prediction, such that $P$ is not equal to $P'$. This further implies that there is a larger difference between decoded frame $F'$ and the original frame $F$. This change results in introducing a non zero distortion $d$ to the current decoded frame, such that we have $d + e = F' - F$. This distortion $d$ accumulates in the reconstructed frames.

\(^4\) $P'$ typically equals $P$, unless there are transmission errors.
as they are progressively decoded, causing a cumulative picture quality degradation [2]. Thus, $d$ gets larger as more and more frames are decoded.

However, it should be noted that this reduction in quality should not be confused with the regular degradation of quality $e$ that is introduced through the encoding-decoding process [10]. The quantization of transform coefficients in the encoder is a lossy process that results in a degradation of quality in a single frame. This degradation $e$ does not accumulate as the number of decoded frames increases. This is due to the fact that there is no mismatch between the decoded and reconstructed frames that are used to form the prediction $P$, and the decoder reconstructed frames, that are used to form the prediction $P'$, which implies that $P' = P$.

The occurrence of drift is highly correlated with the presence of frames that are encoded with inter prediction. When there is a presence of a long sequence of inter predicted frames (all the frames are coded with inter prediction and depend on one another for reconstruction), the quality of each picture frame drops as more and more frames are decoded. P. A. A. Assuncao and M. Ghanbari [9] show that the cumulative effect of drift on the quality of a sequence increases as the length of the inter predicted sequence increases. This effect would cumulatively occur until the introduction of an intra predicted frame to refresh the video sequence [10]. Thus, one solution to reduce drift in an open-loop transcoder is to regularly introduce intra predicted frames into the sequence. The consequence of such an approach is the huge increase in bit-rate that is introduced by intra predicted frames.

Drift occurs in open-loop transcoding, mainly due to the lack of a feedback loop that would compensate for the introduced distortion. This feedback loop would provide a correction that would eliminate the cumulative effect of drift. The use of a feedback loop in a transcoder architecture introduces us to what is known as closed-loop transcoding [10].

![Fig. 2.2 Generic open-loop transcoder.](image-url)
2.4.2 Closed-Loop Transcoding

A closed-loop transcoder consists of an architecture that is similar to that of an open-loop transcoder with the addition of a feedback loop. This feedback compensates for any mismatch that might occur due to a change that is introduced to the coded video sequence, thereby eliminating the possibility of drift [11]. A generic structure of a closed-loop transcoder can be seen in Fig. 2.3. Similarly to the open-loop case, an incoming video stream is first variable length decoded to convert the compressed bitstream into a set of quantized coefficients. These quantized coefficients are subjected to a direct modification, followed by a variable length coding process to produce the output video stream. The closed-loop transcoder includes an additional drift compensation component which functions as a feedback that compensates for any drift incurred by the open-loop setup.

As was discussed in section 2.4.1, drift occurs due to a mismatch between the predictions formed by the encoder and decoder. This mismatch is introduced by the modifier component and results in a distortion that builds up in the sequence as more and more inter predicted frames are transcoded. For every transcoded frame, the drift compensation component calculates the mismatch in prediction and buffers it. It further uses it to compensate subsequently transcoded frames, thereby eliminating drift. More specifically, if the mismatch introduces a distortion $d$, the current frame is compensated by adding $-d$ to it, such that the distortion is eliminated when the sequence undergoes full decoding [11].

Closed-loop transcoders achieve greater flexibility over their open-loop counterparts. They do so by providing a high quality transcoding process for inter predicted frames, without any significant degradation in picture quality. However, one of the main drawbacks of using a closed loop structure is the associated increase in computational complexity and memory requirement.

Open and closed loop transcoders represent two of the most commonly used architectures. Open-loop transcoders are preferred when the application priority is simplicity and low complexity, while closed-loop transcoders are used for more complex operations that require a high quality output.

In the following sections, we present a discussion of two different kinds of transcoding architectures. We differentiate between transcoding in the pixel domain and transcoding in the transform domain.
2.4.3 Pixel-Domain Transcoding

Pixel-domain transcoders have great flexibility in providing a broad range of both homogeneous and heterogenous functionalities. A pixel-domain transcoder architecture performs all transcoding operations in the pixel domain. It performs a full decoding of the incoming video followed by a partial re-encoding to the desired format. The structure of pixel-domain transcoder is directly inspired from a cascaded decoder-encoder setup. A generic cascaded decoder-encoder setup is illustrated in Fig. 2.4.

As was discussed earlier, in a cascaded decoder-encoder setup, the input stream would be fully decoded and re-encoded to the desired format. The full re-encoding process provides great flexibility in choosing the output stream format independently from the input. The decoder and encoder loops are completely independent, such that they can operate at different bit-rates, frame-rates, resolutions, and standards [8]. The use of two independent loops also results in a drift free bitstream.

The main problems with the cascaded setup are the associated high complexity, memory, and delay [12]. This setup could be modified to reduce these high demands by exploiting useful information embedded within the incoming video stream. This information would be used to simplify some of the complex processes performed in the re-encoding process.

When considering the cascaded decoder-encoder setup, the majority of the computational
complexity and delay is a result of the encoding process. T. Shanableh and M. Ghanbari [13] show that for some video standards such as MPEG-1, up to 60-70% of the encoding process comprises of ME search and prediction. As a result, the encoding process could be made significantly faster and less complex if the ME process was to be simplified or avoided.

Most recent video standards use what is called as block based prediction, where each frame is divided into a set of equally sized macroblocks (MB). Each MB can be either coded via intra prediction mode or inter prediction mode. Intra coded MBs rely on the correlation between MBs within the same frame to form a prediction. Inter prediction is based on the fact that there exists a correlation between temporally succeeding picture frames. Even though such frames are considered to be highly correlated, there is still significant motion between them, such as the movement of an object in the video scene from one position to another. This movement can be quantified via ME, by deriving a motion vector (MV) for each MB. These MVs help describe the temporal correlation between different frames, and are used to form the best possible prediction for the current frame [1].

ME could be avoided by making use of MB coding and motion information that is embedded within the incoming stream [14]. Information such as MB coding modes and MVs are extracted from the decoder. This information is post-processed according to the desired transcoding application. Finally it is passed to the encoder, thus avoiding any complex motion estimation search and prediction. Other information such as quantization step-size, bit allocation statistics, and many other could also be extracted from the decoder and re-used to further simplify the encoding process [8].

A generic illustration of a simplified cascaded model, which we will refer to as a pixel-domain transcoder, can be seen in Fig. 2.5. An input video stream is fully decoded to the pixel domain. It undergoes variable length decoding, inverse quantization, inverse transform, and MC. The decoded stream is passed through a pixel-domain spatial/temporal adjustment component. This component provides the transcoder with the flexibility to adjust any of the video sequence temporal and/or spatial properties. Coding and motion information are extracted from the decoded sequence and is passed through a coding and motion information adjustment component. This component performs the necessary adjustments such that the extracted information could be re-used to encode the output stream. The adjustments are highly dependant on the nature and the desired functionality of the transcoding process. This adjusted coding and motion information is subsequently variable length coded with the output stream [8].
2.4.4 Transform-Domain Transcoding

The pixel-domain architecture described in section 2.4.3 could be further simplified by performing all transcoding operations in the transform domain. This can be achieved with the use of a transform-domain transcoding architecture. Fig. 2.6 illustrates a generic transform-domain transcoder architecture. The incoming video stream is only partially decoded through variable length decoding followed by inverse quantization [8]. The resulting transform coefficient are passed through a transform-domain spatial/temporal adjustment component. This component provides the transcoder with the flexibility to adjust any of the video sequence temporal and/or spatial properties, by modifying the transform-domain coefficients. In a similar manner to the pixel-domain transcoder, coding and motion information is extracted from the decoded sequence and are passed through a coding and motion information adjustment component. This component performs the necessary adjustments such that the extracted information could be re-used to encode the output stream. The adjustments are highly dependant on the nature and the desired functionality of the transcoding process. This adjusted coding and motion information is subsequently variable
length coded with the output stream [8].

Transform-domain transcoders have the capacity to provide functionality that is parallel to their pixel-domain counterparts. The elimination of forward and inverse transform results in a low complexity and low memory transcoding process [15]. The fact that no forward or inverse transform is performed gives rise to the need to perform MC in the transform domain.

Generally, when motion compensation (MC) is performed in the pixel domain, motion vectors (MVs) are used to extract a prediction macroblock (MB) from a reference frame. This prediction is used to encode the current MB. MC performs this extraction by adjusting the horizontal and vertical pixel positions of the current MB in par with the horizontal and vertical components of the MV. The MV components are not scaled in multiples of a MB size, hence they could point to a MB sized prediction region that overlaps with four other MBs in the reference frame. This results in four overlap prediction regions. Fig. 2.7 illustrates such a scenario, where the MV of a MB in the current frame is pointing to a MB sized prediction region in the previous frame. The MB sized region overlaps with four
existing MBs in the previous frame, thus creating four overlap prediction regions (overlap 1-4).

Transform-domain MC is performed by extracting the four prediction regions. Each of the four MBs in the previous frame (MB1-MB4) is multiplied with appropriate windowing and shifting matrices to reconstruct the MB prediction in the transform domain [16]. The extracted region is then used as a prediction to encode the MB in the current frame.

The downside of using a transform-domain transcoder is that it still potentially suffers from drift. Transform-domain MC is based on the assumption that the forward transform, inverse transform, and MC processes are linear operations, when in fact they are not. This results in potential rounding operations in the interpolation of sub-pixels [17] and clipping of transform coefficients during MC [2]. A transform-domain transcoder is also considered to be less flexible than pixel-domain transcoder, and is mostly suitable for homogenous applications such as bit-rate reduction. A performance consequence of these drawbacks is that transform-domain transcoders tend to have lower peak signal-to-noise ratio when comparing them to their pixel-domain counterparts [2].

Fig. 2.6 Transform-Domain transcoder.
2.5 Homogeneous Transcoding

A homogenous transcoder performs a conversion between video bit streams of the same standard. It provides the functionality to change properties of the incoming video stream. The importance of homogenous transcoding is related to the emergence of applications such as video-on-demand. In video-on-demand, high quality encoded videos are stored on servers from which users may request to receive. The outgoing channel may have an upper bandwidth limit that cannot be exceeded, hence the video delivered must satisfy real-time constraints by reducing the bit-rate [18]. The transport of high-quality video to wireless and wireline networks with user constrains may also require modification to the bit stream [8]. Due to these high network demands and constraints, there has been intensive research in the area of modifying encoded video bit streams while maintaining a reasonable visual quality [10].

Bit-rate reduction is considered to be one of the most important and commonly implemented homogenous transcoding schemes. There are several methods to reduce the bit-rate of an encoded video stream. The bit-rate could be reduced by directly modifying the quality of each video frame. This can be accomplished by reducing the amount of information that represents each frame. It can be done through methods such as re-quantization [2], or by reducing the amount of coding and motion information. Spatial and temporal resolutions could also be modified to reduce the bit-rate [8].

In this section, we will present an overview of some of the most important functionalities of homogenous transcoding. We start by discussing a few generic methods that are used to reduce bit-rates, then move on to methods involving spatial and temporal resolution.
2 Transcoding Background

2.5.1 Bit-Rate Reduction

Bit-rate reduction transcoding functions as a rate controller that changes the incoming bit stream rate from $R_1$ to $R_2$ where $R_1 > R_2$. Its main purpose is to satisfy certain bandwidth constrains that were not known prior to the encoding of the video stream [14]. The reduced bit stream must satisfy the general transcoding functionalities described in section 2.3, by exploiting as much information as possible from the input stream.

There are several generic methods to reduce the bit-rate of an encoded video stream, one of which is by the re-quantization of the incoming bit stream. The input video stream is variable length decoded and inverse quantized to form the corresponding transform coefficients, which are then re-quantized with a higher quantization step [18]. This reduction in the number of coefficients results in decreasing the number of bits in the output stream.

Another generic method is data partitioning [9]. The input video bit stream is variable length decoded, and the coefficients are modified by discarding the high frequency coefficient that are above a certain threshold. This is possible since the majority of information in a video frame is contained within the low frequency parts [1].

The re-quantization method provides a much better peak signal-to-noise (PSNR) ratio with respect to the data partitioning method [18]. In the original bit-stream, the transform coefficients are optimally quantized accordingly with the original high rate. This optimal quantization is performed again when the re-quantization method is performed. In the data partitioning method, when the high frequency components are discarded, the re-quantization process is not performed. Hence the new coefficients are no longer optimally quantized [18].

Coding and motion information is usually extracted from the incoming video stream and re-used to encode the output stream. There are several issues that need to be addressed when reusing coding and motion information from the input stream [14]:

- A large re-quantization step might result in a MB with only zero quantized coefficients. The MB mode would still indicate that it is coded with inter or intra prediction when it should be skipped. The coding mode of a skipped MB indicates to the decoder that the MB is not coded and that no further information regarding the MB is sent. The decoder reconstructs the MB using MC from the previous reference frame [1]. All
MBs should be checked to see if all the coefficients have been transformed to zeros. The MB mode should then be re-encoded as skip mode.

- MVs that are extracted from the input stream and are re-encoded with the output stream might require a small optimizing modification due to the change in the quantization step. This could be implemented through the use of a MV refinement process that consists of a small search window, such that a mini ME is performed, starting from the current MV position.

There are several architecture models that could be used to construct a bit-rate reduction transcoder. It can be designed with either an open-loop or a closed-loop form, or even a combination of both [19]. In addition, any modification could be performed in either the transform domain or the pixel domain. More specific methods for reducing the bit-rate involve changing of other parameters of the encoded video, such as the reduction of spatial and/or temporal resolutions. These methods ultimately reduce the number of bits in the encoded video stream.

### 2.5.2 Spatial Resolution Reduction

In recent years, there has been a huge growth in the popularity of high definition video. This increase is coupled with a large number of users with access to broadband networks. As a result, there has been a large increase in the availability of videos captured at very high spatial resolutions and quality. The large growth in low-end user devices such as smart phones and hand-held devices requires that these high resolution videos are downsized to adhere to the capabilities of such devices. An added advantage of downsizing is the associated reduction in bit-rate. This would help to better satisfy network and processing constraints. The strict limits associated with video streaming require that the downsizing of these high resolution videos be done with minimal delay. Spatial-resolution transcoding provides a comprehensive solution to this problem. It performs spatial transcoding of encoded videos, while maintaining a high quality output and low processing delay.

There are two major components associated with spatial-resolution transcoding: the downsizing of the pixel dimensions of each frame in the stream, and the downsampling of the associated coding and motion information [13]. We limit our discussion to the downsizing by a factor of two for each dimension (2:1). The same discussed principle could
be applied to other downsizing factors. We also assume that a MB consists of $16 \times 16$ pixels in dimension. A parameter size that is common among most coding standards.

The pixel dimensions of a video frame should be downsized while maintaining a high visual quality. The main objective of pixel downsizing is to reduce the frame pixel dimension, while retaining as much visual information as possible in the downsized frame. There are several methods that are commonly used for pixel downsizing. Some are performed with a pixel-domain transcoding architecture, while others are performed with a transform-domain transcoding architecture [15].

A common pixel-domain downsizing method is pixel averaging. Every $2 \times 2 (M \times N$ in the general case) pixels are averaged to a single value [13]. Other pixel-domain methods consist of filtering with the use of a digital filter in both the horizontal and vertical directions, followed by a downsampler that drops every alternate pixel [13]. A common transform-domain method downsizes the spatial dimensions of a video sequence through reducing an $8 \times 8$ transform block by keeping the top (low frequency) $4 \times 4$ transform coefficients. This procedure is performed on all four $8 \times 8$ transform blocks of every $16 \times 16$ MB [13], [15].

T. Shanableh and M. Ghanbari compare several pixel-domain methods with a transform-domain downsizing method [13]. A video sequence is downsized with each of the methods, followed by an upsampling to the original resolution. The upsampled sequences are compared with the original full-size sequence for each of the discussed methods. It is shown by T. Shanableh and M. Ghanbari that the transform-domain method outperforms the pixel-domain methods in terms of output PSNR [13]. This could be accounted for by the fact that the transform-domain approach keeps the most important information (low frequency coefficients), thus provides the highest PSNR. It is also shown that the filtering and pixel dropping method outperforms the pixel averaging method, since pixel averaging may cause the reconstructed pictures to become blurred [8].

The downsizing of pixel dimensions requires that any associated coding and motion information is downsampled as well. This may be accounted for by the fact that the original full resolution sequence is downsized by a factor of two such that every four MBs are reduced to one MB. In such a scenario, the associated MB coding modes and MVs need to be downsampled in order to be re-used by the transcoder to encode the output video stream [14]. The problem of downsampling MB modes is illustrated in Fig. 2.8, where four MBs of which two are coded with inter prediction (INTER), one coded with intra prediction (INTRA), and one skipped (SKIP) are combined to form a single MB. The
problem of downsampling MVs is illustrated in Fig. 2.9, where four MVs ($MV_1$ to $MV_4$) are reduced to a single MV ($MV_?$).

New coding mode decisions need to be assigned to each resulting MB in the downsized frame. This requires that the coding mode decisions of every four MBs in the original (source) frame are used to pick a coding mode for the target MB. A common method [14], [10] is to code the target MB with an intra coding mode if at least one of the four source MBs was originally intra coded. If none of the four MBs is intra coded and at least one of the MBs is coded with inter prediction, then the target MB should be inter coded. Otherwise if none of the four MBs are either intra or inter predicted, then the target MB should be skipped. This method gives priority to intra coded MBs since they do not have any associated MVs. When no intra coded MBs are present, priority is given to inter predicted MBs of skipped MBs. This is due to the fact that inter coded MBs result in forming a smaller prediction residue when motion is present in the sequence.

There are several other methods that are suggested by P. Yin, et al [20]. These methods handle MBs with mixed coding modes in a different manner:

1. All the target MBs are set to be inter coded, while the MVs of the intra coded MBs in the source frame are set to zero.

2. All the target MBs are set to be inter coded, while the MVs of the intra coded MBs in the source frame are predicted from neighboring MBs.

3. All the target MBs are set to be intra coded.

These methods give different priorities to intra and inter prediction. Several other strategies that use different priority schemes could also be devised. The compression efficiency resulting from these methods would vary accordingly to the nature of the encoded sequence and the type of video coding standard used.

Similarly, a new MV needs to be assigned for each resulting MB in the downsized frame. This requires that the MVs of every four MB in the original (source) frame are used to compute a MV for the target MB. Generally, a combination of the four MVs is used. Such combinations may include [14] randomly selecting one of the four MVs, calculating the mean of the four MVs, or calculating the median. The mean could also be calculated by moving with the majority through calculating the average of the MVs that have same
Fig. 2.8  Four macroblock modes downsampled to one.

Fig. 2.9  Four motion vectors downsampled to one.
direction, and ignoring the ones that do not [13]. The weighted average/median of the incoming MVs could also be used, where the weights are dependant on the spatial activity of the encoded prediction error [8].

A spatial-resolution transcoder would be constructed with the use of either a pixel-domain or a transform-domain transcoder architecture. The spatial/temporal adjustment components in Fig. 2.5 and Fig. 2.6 would downsize the pixel frames, while the coding and motion information adjustment component would take care of downsampling the MB coding modes and MVs.

### 2.5.3 Temporal Resolution Reduction

Temporal-resolution reduction could be used to reduce the bit-rate of an incoming stream, while maintaining a high PSNR per frame. Certain networks and platforms might impose restrictions on the maximum bit or frame rate of an incoming video. This gives rise to the requirement of being able to reduce the temporal resolution of a pre-encoded video, while incurring minimum complexity and delay. This minimization is achieved through the reuse of coding and motion information after subjecting it to an optimizing adjustment [15].

Temporal-resolution transcoding is accomplished through the dropping of frames from an encoded video stream. The extracted MVs cannot be used directly to encode the output stream, since the dropping of frames results in MVs that point to frames that do not exist in the output stream. Therefore, the MVs need to be adjusted such that they can be used for MC prediction of the transcoded video stream. Consider the scenario illustrated in Fig. 2.10, where the frame $F_{n-1}$ is dropped, and a new motion vector $MV_{1+2}$ needs to be estimated so that it can be used as a reference for the frame $F_n$ [21]. The main problem is that one cannot simply take the vector sum of $MV_1 + MV_2$. $MV_1$ points to an area in the frame $F_{n-1}$ with the size of a MB, not to an actual MB. The area pointed to by $MV_1$ has no associated MV, and thus $MV_2$ does not exist. The area pointed to by $MV_1$ can overlap with a maximum of 4 other MBs, each with an associated MV. These associated MVs could be used to estimate $MV_2$.

Several MV estimation methods exits:

- Bilinear interpolation takes a weighted average of the four motion vectors to estimate $MV_2$, where the weights are based on the degree of overlap that the referenced area has with each of the four MBs [15].
• Forward dominant vector selection (FDVS) relies on using the MV from the MB with the greatest area of overlap [22].

• Activity dominant vector selection (ADVS) relies on choosing the MB with greatest transform-domain activity, that is based on the number of none-zero transform coefficients [21].

• Telescopic vector composition (TVC) accumulates all the MVs of corresponding MBs in the dropped frames and uses the resultant accumulated MV [13].

Each of the discussed methodologies would subject the resultant MV to a refinement process that consists of a ME search with a small search window. The compression efficiency resulting from these methods would vary accordingly to the nature of the encoded sequence and the type of video coding standard used.

A temporal resolution transcoder would be constructed with the use of either a pixel-domain or a transform-domain transcoding architecture. The spatial/temporal reduction component in Fig. 2.5 and Fig. 2.6 would drop the required frames, while the motion information adjustment component would re-estimate the MVs.

![Fig. 2.10 Motion vector adjustment.](image)

2.6 Heterogeneous Transcoding

Heterogenous video transcoding converts between video bit streams of different standards. Similarly to homogeneous transcoding, it relies on exploiting information encoded in the
input video stream to reduce the computational complexity of re-encoding the output video stream. Although different video compression standards rely on the same techniques to encode a raw video stream, such as block based motion compensation and prediction, quantization, and transform, there is still a considerable difference between different video standards in terms of both syntax and structure. To improve compression efficiency video standards have considerably evolved over the years, resulting in a vast range from simple MPEG-1 and H.261, to complex MPEG-4 and H.264 [3]. This variety of different video standards added to the large existing quantity of coded video streams, places a huge importance in providing mechanisms to efficiently convert between these videos with minimum complexity, while maintaining a considerably high visual quality.

When transcoding between different video compression standards, there are several issues that need to be taken into consideration [10]. Different standards could use different header syntax and structure, in terms of the signaled flags and bit representation. The prediction residue itself might have a different encoding syntax from one format to another. Video standards often significantly differ in parameters related to the encoding structure. Different transforms might be used, as well as different prediction schemes. Different inter prediction schemes, frame referencing structures, MV range and accuracy, and different intra prediction algorithms could be used as well. Different standards might also have different MB properties, such as partitioning sizes, picture type,... [3].

There is an increased complexity associated with heterogeneous video transcoding, due to the different encoding formats of the input and output sequences, and the possible requirement of spatial-temporal adjustment. This increased complexity places restrictions on the transcoding architectures. It makes it necessary to use two independent decoder and encoder loops [13]. The pixel-domain and transform-domain architectures that were presented in sections 2.4.3 and 2.4.4, and illustrated in Fig. 2.5 and Fig. 2.6, could be used to construct a heterogeneous transcoder. Coding and motion information that is directly fed from the input to the output sequence needs to undergo a syntax adjustment to suit the output video standard [8]. Different homogenous transcoding functionalities could be provided in a similar manner to the homogenous case, where MVs, quality, spatial/temporal resolutions, among other properties of the input video stream could undergo various adjustments to suit the properties of the output stream [13].

In this thesis, we propose a heterogeneous transcoding scheme that implements a closed-loop and pixel domain architecture.
2.7 Summary

In this chapter, a general overview of different transcoding types and architectures was presented. A set of requirements that should be met by a transcoder were established. Open-loop and closed-loop transcoding architectures were discussed with special emphasis on the drift problem. Pixel-domain and transfer-domain architectures were also presented. Different transcoding types such as homogenous and heterogenous transcoding were discussed. Special attention was paid to bit-rate, spatial-resolution, and temporal-resolution reduction mechanisms.

The need for such mechanisms motivates the use of scalable video coding standards, such as the scalable extension of the H.264 standard. Different types of resolution reduction for a compressed video stream, such as quality, spatial, and temporal reduction could be achieved by simply discarding parts of the coded bit stream. The different schemes discussed in this chapter provide some of the basic methodologies used to design our proposed scalable transcoder. Some of the key difficulties that will be encountered when transcoding from a single layer H.264 stream to its scalable form were highlighted, such as MV interpolation and information downsampling. In the next chapter, an overview of the H.264 video compression standard is presented, together with its scalable extension. We also discuss various issues that are related to the transcoding between these two formats.
Chapter 3

The H.264 Video Compression Standard

In this chapter, we introduce the H.264 video coding standard. We present an overview of both its single layer and scalable formats. We also discuss some of the main issues that involve transcoding from the single layer to the scalable form.

3.1 The H.264/Advanced Video Coding Standard

Throughout the mid-nineties MPEG-2 (H.262) had emerged to be the prime video coding standard. The added advantages it had over the older MPEG-1 standard, such as support for interlaced video, established it as the dominant video coding standard for TV systems on a global scale. MPEG-2 is widely used for transmission of both standard and high definition TV signals via satellites and cable. It is also the most common method of storing high quality standard definition video on DVDs [23]. The gradual increase in demand for high definition video, coupled with the strict bandwidth limits that are imposed by transmission platforms such as cable modem, xDSL, and wireless networks, require an increase in the coding efficiency of transmitted data. In order to meet such high demands, video coding standards have thus evolved from MPEG-2 to more complex video formats such as H.263 and MPEG-4 [23].

Video compression standards have continued to evolve over the years, with the objective of maximizing coding efficiency and adapting to new network functionalities. MPEG-4
visual emerged to provide video shape coding capabilities, with many additional functionalities. In late 2001, the Video Coding Expert Group (VCEG) and the Moving Picture Experts Group (MPEG) formed a Joint Video Team, in order to finalize the draft for a video coding standard that can achieve double the coding efficiency with respect to any other current video coding scheme [23]. This newly standardized video standard was named H.264 Advanced Video Coding (H.264/AVC).

In the following section, we present a technical overview of the single layer form of the H.264 standard, H.264/AVC. We first present the encoder and decoder structures used by the H.264 standard. We then discuss some of the main technical details of the H.264 standard. Finally we discuss the main aspects of the high level design of the H.264 bit stream.

3.1.1 H.264 Codec Overview

The H.264 CODEC (enCODer/DECoder pair) consists of a similar structure to older CODEC designs. It makes use of common video coding methods such as motion estimation (ME), motion compensation (MC) prediction, transform, quantization, and variable length coding with the added exception of using a deblocking filter. Generic forms of the encoder and decoder are illustrated in Fig. 3.1 and Fig. 3.2, respectively.

The Encoder

The encoder consists of two distinct data flow paths. A forward encoding path and a backward reconstruction path. In the forward path, each frame is processed in macroblock (MB) units with each MB encoded with the use of either inter or intra prediction (a prediction \( P \) is formed). MBs can be partition to sub-MBs (blocks\(^1\)). The forward path of the encoder subtracts the prediction \( P \) from the current block, to produce a prediction residue \( R \). The prediction residue \( R \) is transformed and quantized to produce a set of quantized coefficients \( QC \). These quantized coefficients \( QC \), together with any coding and motion information are reordered and variable length coded to form an output packet. The output stream consists of entropy coded coefficients, together with side information that is used for decoding proposes [1].

\(^1\)The word block may refer to a MB or a sub-MB.
The prediction residue $R$ that is formed for each block is based on reconstructed picture samples. For the inter prediction case, the prediction is computed through motion-compensated prediction from one or two previously encoded reference pictures stored in the reference picture buffer component. Intra predicted blocks rely on previously encoded blocks of the current frame to form the prediction residue $R$. These blocks have undergone the encoding process, and have been decoded and reconstructed [1].

The reconstruction path decodes each block such that it may be used as a reference for inter or intra prediction. The quantized coefficients $QC$ are inverse-quantized, and inverse-transformed to reproduce the prediction residue $R'$, which is also added to the corresponding current prediction $P$ to form a reconstructed block, such that it can be used for future predictions. The deblocking filter component is applied to minimize any distortion that might be caused by the block edges that arise from the use of block based coding. The resulting blocks are used to form reference pictures and are stored in the reference picture buffer component [1].

**The Decoder**

The input compressed H.264 stream is processed by the entropy decoder and reordered to reproduce a set of quantized coefficients $QC$, together with associated coding and motion information that is used for frame reconstruction. These coefficients are further inverse quantized and inverse transformed to produce a prediction residue $R'$. A prediction of the current block $P$ is either constructed through MC of previously decoded reference frame taken from the reference frame buffer component, or through intra prediction of previously decoded MBs of the current frame. The prediction $P$ is added to the prediction residue $R'$ to reconstruct the current block. A deblocking filter component is applied to minimize any distortion. The resulting blocks are used to form reference frames and are stored in the reference picture buffer component [1].

**3.1.2 H.264 Standard Details**

**Profiles**

The H.264 standard consists of three different profiles [1], each supporting a particular set of features. The *Baseline Profile*, supports both intra and inter prediction with context-adaptive variable-length codes (CAVLC) [1]. The *Main Profile* supports interlaced video,
Fig. 3.1 H.264/AVC Encoder.

Fig. 3.2 H.264/AVC Decoder.
inter-coded bi-predicted frames with weighted prediction, and entropy coding using context-adaptive binary arithmetic coding (CABAC) [1]. Finally, the Extended Profile supports efficient switching between bitstreams and improved error resilience, but lacks the support for interlaced video or CABAC [1].

Video Formats

The H.264 standard supports a two field interlaced format, and supports a progressive format as well [1]. There are two main components that are used to represent a pixel, a luminance (luma) component which represents the brightness and chrominance (chroma) components that represent color. The default sampling format is a progressive 4:2:0 in which chroma components have half the horizontal and vertical resolution of luma component. The chroma samples are horizontally aligned with every other luma sample and are vertically aligned between every two luma samples [1]. In a 4:2:0 format, each MB consists of 16x16 luma region, with two 8x8 corresponding chroma regions [24].

Slices and Macroblocks

Each frame is coded as one or more slices [1]. Each slice contains a slice header and a payload consisting of a series of encoded MBs. The number of MBs per slice is not necessarily a constant. Inter predicted slices use lists that contain indices that refer to reference frames. These lists are known to as list 0 and list 1. The H.264 syntax consists of three different slice coding types [1]:

- **I-slices**: Contain only I-coded MBs, where each MB is intra predicted from previously coded MBs in the same slice.
- **P-slices**: Contain P-coded MBs, which are inter predicted from a list 0 reference frame and/or I-coded MBs.
- **B-slices**: Contain B-coded MBs, which are inter predicted from a list 0 and/or list 1 reference frame and/or I-coded MBs.

Each coded MB is predicted from previously encoded data. MBs coded with inter prediction make use of samples in frames that have been encoded and reconstructed, while intra prediction depends on previously encoded MBs that are within the same slice.
MB (or sub-MB) prediction is formed by subtracting previously encoded samples from the current block samples. The prediction is further compressed and transmitted with the corresponding MB header as the output bitstream [1]. Each MB header contains the quantization parameter, MB coding modes, MB reference lists (list 0 and list 1), and motion vectors (MVs) (if inter predicted).

**Referencing Structure**

Each block may use one or two previously encoded pictures as references for inter prediction. For each frame, the encoder maintains two lists of previously encoded reference pictures (independent of the display order), list 0 and list 1. P-coded inter predicted frames use list 0 for inter prediction references, while B-coded inter predicted frames use both list 0 and list 1 for inter prediction references. Each list may contain frames that precede or succeed the current frame in display order [1].

**Intra Prediction**

Intra prediction makes use of the correlation between the current block and previously encoded and reconstructed blocks of the same slice. It constructs a prediction from these previously encoded and reconstructed blocks, and subtracts this prediction from the current block to form the prediction residue. In the H.264 video standard, there are a number of supported intra prediction schemes, namely intra prediction for 4×4 luma samples, for 16×16 luma samples, and for 8×8 chroma samples. Each block relies on the encoded and reconstructed prediction samples that are directly above and to the left of the current block [23].

The intra 4×4 scheme splits each MB to 4×4 luma blocks, and predicts each partition independently. There are a total of nine different prediction modes for 4×4 blocks, each relying on a particular combination of the prediction samples. Mode 0 uses the samples that are directly above the block (vertical prediction), while Mode 1 uses the samples that are directly to the left of the block (horizontal prediction). Mode 2 averages the top and left samples (DC prediction), while modes 3-8 make use of different forms of diagonal predictions [23].

In the intra 16×16 scheme, an entire 16×16 luma component of a MB is predicted with a single prediction operation [1]. There are a total of four different prediction modes for
16×16 blocks, each relying on a particular combination of the prediction samples. Mode 0 uses the samples directly above the block (vertical prediction), while Mode 1 uses the samples that are directly on the left of the block (horizontal prediction). Mode 2 averages the top and left samples (DC prediction). The final mode uses a combination of the upper and left prediction samples [1].

For chroma samples, an intra 8×8 scheme is employed. Both chroma components always use the same prediction mode [1]. There are a total of four different prediction modes for 8×8 chroma blocks. These modes are identical to the ones used for predicting 16×16 luma blocks [23].

The intra 4×4 scheme is particularly suitable for coding parts of a video frame that contain a significant amount of detail, while the 16×16 scheme is more suited for coding smooth parts of a video frame. Chroma samples are always intra predicted with the 8×8 scheme since chroma samples are usually smooth over large areas of a frame [23].

**Inter Prediction**

Inter prediction makes use of the correlation between the current block and previously encoded and reconstructed frames. It creates a prediction with the use of block based ME and prediction. This prediction is subtracted from the current block to compute the prediction residue.

**Tree structured motion compensation**

The H.264 standard employs a tree structured motion compensation scheme. Each 16×16 luma MB can be partitioned in four different ways. It can be represented by one 16×16, or two 8×16, or two 16×8, or four 8×8 partitions. Fig. 3.3 illustrates the four different MB partitioning setups. If a MB is split to 8×8 partitions, each 8×8 partition could be further represented by one 8×8, or two 4×8, or two 8×4, or four 4×4 partitions [1]. Fig. 3.4 illustrates the four different sub-MB partitioning setups. A separate MV is required for each MB partition or sub-MB partition.

Chroma MBs are 8×8 pixels in dimension, and they also employ a tree structured motion compensation scheme. They have the same partitioning structure as luma MBs do, except that the partitioning sizes have exactly half the vertical and horizontal resolutions. Each 8×8 chroma MB can be partitioned in four different ways. It can be represented by
one 8×8, or two 4×8, or two 8×4, or four 4×4 partitions. If a chroma MB is split to four 4×4 partitions, each 4×4 partition could be further represented by one 4×4, or two 2×4, or two 4×2, or four 2×2 partitions [1].

Smaller partitions transmit a larger number of bits for specifying the motion vectors and partitioning sizes, but result in a smaller prediction residue for areas that contain significant detail. Larger partitions transmit a smaller number of bits for specifying the motion vectors and partitioning sizes, but might result in a larger prediction residue for areas that contain significant detail [1]. This implies that small partitions should be used for areas in the frame with significant detail, while larger partitions should be used for flat areas in the frame.

![Macroblock partitions](image)

**Fig. 3.3** Macroblock partitions: 16×16, 8×16, 16×8, 8×8.

![Sub-Macroblock partitions](image)

**Fig. 3.4** Sub-Macroblock partitions: 8×8, 4×8, 8×4, 4×4.

**Motion vectors**

In the H.264 standard, MVs have a quarter-sample (quarter-pixel) resolution for luma components, and a one-eight-sample (one-eight-pixel) resolution for chroma components [1]. A MV represents an offset between the position of the current block, and the position of the reference prediction area (with the same size as the current block) in the reference frame. Each constructed MV has a quarter pixel accuracy, such that it points to a position that
lies between the frame pixels. Quarter pixels are predicted through interpolation of regular pixels [1].

Motion vector prediction is a feature of the H.264 standard that reduces the number of bits required to transmit a MV. It makes use of the high correlation between MVs of neighboring MBs. Instead of encoding a full MV, a prediction $MV_p$ is formed from previously computed MVs, and is subtracted from the current MV. The difference between the current MV and the prediction $MV_p$ is encoded [1].

**P-coded MBs**

In the H.264 standard, each P-coded MB can use a single frame from list 0 as a reference [1]. The frame can precede or succeed the current frame in temporal (or display) order, but must precede the current frame in coding order. The standard limits one reference frame per MB, but each partition in the current MB can use a different part of the reference frame for prediction. Each frame is limited to the reference frames that are present in list 0.

**B-coded MBs**

In the H.264 standard, each B-coded MB can use one reference frame from list 0 and one reference frame from list 1 for prediction [1]. The frames in each list can precede or succeed the current frame in temporal order, but must precede the current frame in coding order. The standard limits one reference frame from each list for each MB. Although each partition in the current MB can use a different part of the reference frame for prediction. Each frame is limited to the reference frames that are present in list 0 and list 1.

The presence of two references per MB, one from list 0 and the other from list 1, provides a flexibility for having several prediction methods for B-coded MBs. Similarly to P-coded MBs, B-coded MBs can be predicted using motion compensated prediction from only list 0. They can also be predicted using motion compensated prediction from only list 1. Another possible scheme is motion compensated bi-predictive prediction from both list 0 and list 1. In the bi-predictive scheme [1], a prediction block is constructed from both list 0 and list 1, where two block sized motion compensated areas are obtained, and the resultant prediction is calculated as a weighted average of the two areas.

A special kind of prediction scheme for B-coded MBs is known as direct prediction. In direct prediction no motion vectors are transmitted [1]. The decoder estimates list 0 and
list 1 MVs from the MVs of previously coded MBs. These estimated MVs are used by the decoder to perform bi-predictive motion compensation prediction.

**Transform**

The H.264 standard uses three different transforms depending on the type of encoded prediction residue. A Hadamard transform is used for $4 \times 4$ intra $16 \times 16$ DC luma samples. A Hadamard transform is also used for $2 \times 2$ chroma coefficients. Finally, a $4 \times 4$ block integer transform (IT) is used for all other modes. Unlike older standards which use a discrete cosine transform (DCT), H.264 employs an integer transform that has similar properties to the DCT [23]. In an integer transform, all operations are performed using integer arithmetic, making it possible to ensure zero mismatch between the encoder and decoder inverse transforms. It also has the added simplicity of being implemented using only additions and shifts [1].

**Entropy Coding**

At levels above the slice level, elements are coded as fixed-length or variable length binary codes. At the slice level, elements are encoded using either context-adaptive coding (CABAC) or context-adaptive variable-length codes (CAVLC). The chosen scheme depends on the entropy coding mode [1].

**Deblocking Filter**

A major problem of block-based coding is the possible appearance of visible block structures in the picture frame. This occurs because block edges are reconstructed with less accuracy, giving rise to a block appearance in the picture frame. H.264 employs a deblocking filter to reduce this blocking effect. The deblocking filter measures the absolute difference between neighboring edges of the blocks. A large difference would detect the presence of a blocking effect. The filter would subsequently reduce this difference and smoothen the edges in order to eliminate the blocking effect. Consequently, there is a significant improvement of human subjective quality, and the bit-rate is reduced by 5%-10% [23].
3.1.3 H.264 High Level Design

The H.264 encoded bitstream is considered to be highly flexible in its design in the sense that it makes a clear distinction between a video coding layer (VCL) and a network abstraction layer (NAL). The VCL is implemented in a manner that efficiently represents the coded video content [23], and is mapped to the NAL. The NAL consists of units that contain the payload data and associated header information. A video sequence is represented by a sequence of NAL units that can be transmitted over a packet network, or as a bitstream transmission, or even stored on a disk platform [1]. These design decisions make H.264 highly flexible for use in a broad range of applications.

The VCL follows what is known as a block-based hybrid video coding approach. In this approach, pictures are partitioned into smaller coding units, namely into MBs and slices. At the network level, coded video data consists of organized packets containing an integer number of bytes called network abstraction layer (NAL) units [25]. The structure of a NAL unit consists of a one byte header, followed by the payload data. There are two types of NAL units, VLC NAL and non-VCL NAL units. VCL NAL units contain coded slice or coded slice data partitions, while non-VCL NAL units contain additional associated information [25]. A series of consecutive NAL units with specific properties is called an access unit. The decoding of an access unit results in the decoding of one frame [25]. A series of access units with specific properties form a decodable video sequence.

3.2 The Scalable Video Coding Extension of the H.264/AVC Standard

In the following section, we present a technical overview of the scalable extension of the H.264 standard, H.264/SVC. We first present an overview of the different dimensions of scalability that are provided, along with their technical properties. We then discuss the main issues regarding the high level design of the H.264/SVC encoding process together with the bit stream structure. A good overview of the SVC design has been published by H. Schwarz, D. Marpe, and T. Wiegand [25].
3.2.1 Scalability Overview

In the H.264/SVC standard three dimensions of scalability exist: temporal, spatial, and quality scalability.

Temporal Scalability

Temporal scalability is established when the frame referencing structure allows the frames to be partitioned into a temporal base layer and one or more enhancement layers. We define a temporal layer identifier $T$ such that $T = 0$ for the base layer and is incremented by 1 for each subsequent enhancement layer. For any encoded stream with a maximum temporal layer identifier $T_{\text{max}}$ (highest enhancement layer), an extracted scalable sub-stream should contain all of the frames that have a temporal layer identifier $T$ that is smaller or equal to a given positive integer $k$, where $0 \leq k \leq T_{\text{max}}$.

The frame referencing structure of a temporally scalable bit stream places restrictions on motion compensation and prediction. The current frame (to be predicted) can only use a reference frame that has a temporal layer identifier $T$ that is smaller than or equal to the current frame temporal layer identifier. This implements what can be described as a hierarchical prediction structure.

The hierarchical prediction scheme employed by SVC allows a large variation of different referencing structures. Temporal scalability could be efficiently provided with the use of hierarchical B-frames [26]. Base layer frames are coded as either I-coded or P-coded frames, while enhancement layer frames are coded as B-coded frames. The temporal referencing structure restricts the reference frames in list 0 and list 1 to the temporally proceeding and succeeding frames, respectively. These reference frames must have a temporal layer identifier $T$ that is less than the temporal layer identifier of the current frame (to be predicted). The set of frames between two base layer frames are collectively defined as a group of picture (GOP). The coding order of the hierarchical B-frames structure has to be chosen in such a way that frames are coded before they are used as references for other frames. An example of the hierarchical B-frames structure for a GOP size of 8 can be seen in Fig. 3.5. The frames are placed in display order. The first row is labeled with the coding type (I or P or B) and the coding order. The numbers on the second row specify the temporal layer identifier $T$.

Another common SVC structure is the hierarchical zero-delay structure. Enhancement
Fig. 3.5  Coding with a hierarchical B-frame structure. The frames are placed in display order. The first row is labeled with the coding type (I or P or B) and the coding order. The numbers on the second row specify the temporal layer identifier $T$.

Fig. 3.6  Coding with a zero-delay structure. The frames are placed in display order. The first row is labeled with the coding type (I or P) and the coding order. The numbers on the second row specify the temporal layer identifier $T$. 
layer frames are coded as P-coded frames. List 0 frames are restricted to the temporally preceding frames. Reference frames must have a temporal layer identifier $T$ that is less than the temporal layer identifier of the current frame (to be predicted). This scheme provides a zero structural delay with a coding order that is equal to the display order. An example of the hierarchical zero-delay structure can be seen for a GOP size of 8 in Fig. 3.6. The frames are placed in display order. The first row is labeled with the coding type (I or P) and the coding order. The numbers on the second row specify the temporal layer identifier $T$.

SVC also employs what may look like a hierarchical quantization scheme. For a given base layer quantization parameter $QP_0$, the quantization parameter for an enhancement layer frame $QP_T$ with temporal identifier $T$ can be determined by $QP_T = QP_0 + 3 + T$. The resulting coded pictures are smoothly reconstructed by the decoder, regardless of any large fluctuations in peak signal-to-noise ratio that this variation in quantization might cause.

**Spatial Scalability**

Spatial scalability is established when the frame referencing structure allows a given frame to be represented by a spatial base layer and one or more enhancement layers. We define a spatial layer (or dependency layer) identifier $D$ such that $D = 0$ for the base layer and is incremented by 1 for each subsequent enhancement layer. Each spatial layer (of a frame) can be encoded by employing inter and intra prediction to compute the prediction residue (with other frames of the same spatial layer). The compression efficiency can be further improved by making use of any correlation between different spatial layers, namely through inter-layer prediction.

Inter-layer prediction improves compression efficiency by making use of the lower spatial layers to encode the higher layers. For any given spatial enhancement layer frame with a layer identifier $D > 0$, the prediction can be formed through either motion compensation and prediction with other frames with the same dependency layer identifier $D$ (temporal referencing), or through the use of upsampled information from lower spatial layer (with smaller $D$). There are three main components of inter-layer prediction:

1. **Inter-Layer Motion Prediction**: MB motion information from lower spatial layers (references) can be upsampled to be used by the higher enhancement layers. This information is derived from every 8×8 MB partition in the reference layer. MB
partitioning decisions from lower spatial layers (reference) can be upsampled to be used by the higher enhancement layers. This is accomplished by upsampling the MB mode decisions of every $8 \times 8$ partitioning in the reference frame. If the $8 \times 8$ partition in the reference frame is further partitioned, each $M \times N$ partition in the reference frame block corresponds to an $(2M) \times (2N)$ partition in the enhancement layer, otherwise the upsampled $8 \times 8$ MB is not partitioned in the enhancement layer. SVC also includes the option of upsampling MVs from lower spatial layers. The upsampled MVs are scaled to fit the larger spatial resolution of the enhancement layer frame.

2. **Inter-Layer Residual Prediction**: The size of the prediction residue of an enhancement layer frame can be reduced with the use of inter-layer residual prediction. The prediction residue of every $8 \times 8$ MB partition in the lower spatial layer reference frame is upsampled using a bilinear filter. This upsampled residue is used as a prediction for the enhancement layer residue, by subtracting it from the enhancement layer residue and encoding the difference.

3. **Inter-Layer Intra-Prediction**: When a MB in a lower spatial layer (reference) is intra-coded, inter-layer intra-prediction could be employed. The reconstructed signal (resulting from intra prediction) of every $8 \times 8$ MB partition in the lower layer reference frame is upsampled to be used as a prediction signal for the higher enhancement layer. The luma samples are upsampled using a one-dimensional 4-tap FIR filter that is applied horizontally and vertically. The chroma samples are upsampled using a bilinear filter. Once the intra coded signal is upsampled it is subtracted from the corresponding enhancement layer signal and the difference is encoded.

It is important to note that each of these schemes can be independently applied during the encoding process. This provides the encoder with the flexibility to maximize compression efficiency by applying the inter-layer prediction components that are most suitable for a given video stream.

**Quality Scalability**

Quality scalability refers to the concept of having different layers coded with different quantization parameters, while maintaining the same spatial/temporal resolution. We define a quality layer identifier $Q$ such that $Q = 0$ for the base layer and is incremented by 1 for
each subsequent enhancement layer. The H.264/SVC design provides packet-based quality scalable coding through the implementation of medium-grain quality scalability (MGS). MGS allows an instant switching between different MGS (quality) layers within any access unit (frame).

Some of the older standards such as MPEG-2 employ quality scalability using enhancement quality layers as references for lower quality layers. This scheme provides a high compression efficiency for the enhancement layers, but suffers from drift when the lower layers are independently decoded. Other standards such as MPEG-4 avoid drift by using the base quality layer as reference for the enhancement layers. Although this scheme is considered to be drift free, it provided low compression efficiency for enhancement quality layers.

SVC implements a quality scalability structure that makes use of the temporal hierarchical prediction structure employed in the standard. MGS makes use of the concept of key pictures, with one key picture per GOP. A key picture basically forms the temporal base layer frames in temporally scalable streams. In SVC quality scalability, key pictures only use other key pictures as references and they cannot be discarded by the scalable stream. All other pictures use key pictures, and other base and enhancement frames as references. For each frame, a flag is transmitted indicating whether the base quality layer or the enhancement quality layer is used as a reference. The presence of key pictures helps to refresh the sequence, such that the occurrence of drift would be limited within a GOP.

3.2.2 SVC High Level Design

SVC is designed with the capability to provide three dimensions of scalability: temporal, spatial, and quality. These three types of scalability can be combined within a single SVC bit stream. A set of access units forming a GOP are structured to provide temporal scalability, with each access unit identified by a temporal layer identifier $T$. Each access unit can contain a set of spatial layers, Each spatial layer with a given dependency layer identifier $D$, can contain one or more quality layers, each with a quality layer identifier $Q$. It is important to note that any of the different scalability schemes can be independently employed, such that an SVC bit stream does not need to provide all types of scalability.

Switching between different temporal or spatial layers, such that a sub-stream with lower spatial or/and temporal resolution is extracted, is only possible from one GOP to the
next. Switching between different quality layers is possible within an access unit (frame). Identifiers such as the temporal $T$, dependency $D$, and quality $Q$ layer identifiers have to be specified in each encoded VCL-NAL unit, such that a sub-stream with the desired scalability format could be easily extracted. Thus, the one byte VCL-NAL header specified by the H.264/AVC standard has to be extended by three bytes, such that it would contain the required layer identifier information.

3.3 Transcoding from H.264/AVC to H.264/SVC

Transcoding between different video compression standards involves converting the incoming bit stream from one format to another. An obvious approach would be to simply decode the incoming bit stream, followed by a full re-encoding to the desired format. Unfortunately, such a scheme is coupled with high computational complexity and delay. When transcoding between different compression standards there are several coding and structural components that might differ from one standard to another, and would thus require considerable modification. Components such as prediction structure, MB sizes, transform type, MV format, and many others need to be adjusted in order to be adequately encoded with the output stream. The nature of this adjustment depends on the coding standards involved. The form of modification that is required is also highly dependant on the employed transcoding architecture (pixel-domain or transform-domain), and the amount of coding and motion information that is reused.

Generally, transcoding from one video coding format to another requires the adjustment of the frame spatial/temporal resolution and involves the re-employment of coding and motion information [13]. Other coding components such as transform type and bit stream syntax are usually handled by the re-encoding process. The incoming video stream is first decoded (extent of decoding depends on the architecture). Each decoded frame is subjected to a spatial/temporal resolution adjustment. Coding and motion information is extracted from the decoded sequence and re-employed to encode the output stream. The information extracted from each frame is mapped to an output frame. This information is modified, since the output stream could have a different coding and frame referencing structure. MVs and reference indexes are modified to adhere to the new referencing structure and/or to the adjusted spatial/temporal resolution.

A similar methodology can also be used to perform temporal and/or spatial transcoding
from H.264/AVC to H.264/SVC. The transcoding process would modify the original incoming H.264/AVC bit stream such that it would provide temporal and/or spatial scalability together with adhering to the SVC bit stream syntax. The incoming bit stream would first be decoded, followed by a spatial/temporal resolution adjustment. Any extracted coding and motion information would be re-employed in the re-encoding process. The compatibility of different coding properties between AVC and SVC, such as MB coding modes and sizes helps to simplify the transcoding process. Unfortunately, the broad range of encoding formats supported by the H.264 standard, together with the different scalable options provided by SVC adds a layer of complexity to the process. The required adjustments would be highly dependant on the target scalability form.

Temporal scalability in SVC requires that each group of pictures (GOP) adheres to a frame referencing structure that allows the frames to be partitioned into a temporal base layer and one or more enhancement layers. Any transcoding process would be required to map the incoming bit stream referencing structure to the SVC referencing structure. MVs of the incoming stream have to be modified in order to be used by the re-encoding process to encode the SVC scalable referencing structure.

Spatial scalability in SVC requires a frame structure that allows a given frame to be represented by a spatial base layer and one or more enhancement layers. Incoming video frames would require a spatial resolution adjustment to provide different scalable layers. MB coding modes, sizes, and associated MVs would also require a modification, such that they may be used to encode different spatial resolutions.

Some prior work on transcoding from H.264/AVC to H.264/SVC can be found in the literature. A. Dziri, et al [5], propose a P-picture based temporal transcoder that provides two temporal scalable layers (GOP of size 2). This transcoding scheme performs a full decoding of an incoming IPPP structured H.264/AVC stream, followed by the re-encoding of every other frame (with an even display number) with an H.264/AVC encoder. The re-encoded sequence (with half the frame rate) would form the base layer. The original un-decoded odd numbered frames are then inserted into the stream, thus forming the enhancement temporal layer. The appropriate syntax adjustments are made to the headers, such that the sequence can be fully decoded by an SVC decoder. The problem with such a scheme is that it requires a full encoding of at least half of the sequence, which is still computationally expensive. It is also limited to only providing two temporal layers where if any additional layers were to be provided, the computation time would rise significantly.
Extensive work has been done in the area of transcoding an H.264/AVC stream to a quality scalable H.264/SVC stream. Several H.264/AVC to H.264/SVC transcoding methods and architecture have been proposed by J. De Cock, et al [6] and J. De Cock, et al [7], [27]. They propose a transcoder that converts a single layer H.264/AVC stream to a quality scalable SVC bit stream with CGS layers. Although a considerably good performance is achieved, it is limited to only providing quality scalability through re-quantization in the transform domain. Thus making it inflexible for extending to other scalable forms. We also note that adding quality scalability to an existing H.264/AVC stream is out of the scope of this thesis.

3.4 Summary

In this chapter, a general overview of the H.264 video coding standard was presented. We started off by discussing the main technical details involving the single layer (AVC) form of the standard. It was followed by a brief overview of its scalable (SVC) extension. We also discuss some of the main issues that involve transcoding from single layer H.264/AVC to the scalable H.264/SVC form, together with some of the relevant work found in the literature. In the next chapter we propose a novel implementation of a temporal transcoder of H.264/AVC to H.264/SVC.
Chapter 4

Temporal Transcoding of H.264/AVC to H.264/SVC

Temporal scalability provides an encoded video bit stream with the ability to dynamically support different bit-rates and frame-rates on the fly. The frame-rate could be adjusted by simply discarding parts of the encoded stream, thus reducing the overall bit-rate as well. This provides an encoded bit stream with the flexibility to adjust accordingly with any network requirements.

SVC provides temporal scalability to streams encoded in the H.264 format by adhering the coded sequence to a hierarchical referencing structure. In order to add temporal scalability to an existing single layer H.264/AVC stream, the frame referencing structure of the input single layer stream should be mapped to the hierarchical referencing structure used by SVC. Such a mapping would require computing a set of new motion vectors (MVs) and re-using the coding and motion information contained in the input stream. This mapping would be accomplished through the use of a temporal transcoder.

In this chapter, a novel implementation of a temporal transcoder of H.264/AVC to H.264/SVC is presented. A time profiling of the SVC encoder is performed to categorize the computational complexity of temporal scalability encoding. We also discuss the technical details involved in our transcoder implementation. Finally, we present some experimental simulations of our implementation. It is important to note that for simulation purposes, we use the SVC reference software that was developed by the Joint Video Team [28]. This reference software is used as a platform for simulating and testing the implemented scheme.
4 Temporal Transcoding of H.264/AVC to H.264/SVC

4.1 Time Profiling of Reference Software

4.1.1 Profiling Method

Generally, transcoder architectures rely heavily on reusing the different components found in the encoder to facilitate the transcoding process. When designing any transcoder, it is essential that the computational complexity of the process is minimized. This should be achieved while maintaining a high visual quality for a given bit-rate. In order to categorize the computational complexity of the different encoder components, we perform a time profiling of the SVC reference software. This time profiling provides us with statistics that indicate the fraction of the total encoding time is used by different encoder components. These statistics also detail the fraction of time taken by different inter and intra prediction schemes, thus providing us with a clear picture on how to design a low complexity AVC to SVC temporal transcoder.

We make use of the reference SVC encoder software to produce the necessary statistics [28]. A sampling profiling is performed on the encoder. The reference encoder is sampled while it performs a full video encoding to produce an SVC video sequence with temporal scalability. The profiler captures the current program state by sampling it at specified intervals (clock cycles). This sampled state indicates which particular operations were invoked at the sampling instant. The profiler samples the encoder at a rate of 1 sample per million clock cycles. The total number of samples collected from each operation indicate the fraction of time taken by that particular operation.

The profiling simulation uses the default settings of the reference software JVSM 9.16 [28]. It fully encodes an SVC video sequence with hierarchical B-frames, a sequence length of 81 frames, and a QCIF resolution of 176 by 144 pixels. The encoding process uses a fast motion estimation search algorithm with 32 pixels search range, and standard (equal weights) weighted prediction. We limit the simulation to only the Foreman video sequence [29], since the variations between different sequences were observed to be negligible.

1It is important to note that the reference SVC encoder software that is used might not be optimized. Specific parameters were chosen to provide us with a fast encoding process. Changing some of these parameters might vary the compression efficiency and encoding time.
4 Temporal Transcoding of H.264/AVC to H.264/SVC

4.1.2 Profiling Results

The profiling simulation results are summarized in Fig. 4.1 and Fig. 4.2. The pie chart in Fig. 4.1 illustrates a breakdown of all of the main encoding processes in terms of the fraction of time taken from the total encoding time. It can be seen that all of the different prediction types make up 95% of the encoding process. Inter prediction ME and MC take up 51% of the full encoding process; together with MV quarter pixel approximation, they add up to 69% in total. Intra prediction seems to have a smaller share of the pie with a collective fraction of only 8%. Process such as transform and quantization, along with other operations are accounted for by the 15% prediction post-processing.

These statistics suggest that in order to significantly reduce the encoding computation time it is necessary to reduce the computation time taken by the ME and MC processes. This in turn could also possibly reduce the fraction taken up by quarter pixel approximation. The SVC reference software [28] performs inter prediction macroblock (MB) mode decision by forming a MB prediction residue for all of the different partitioning sizes and comparing them to each other. The partitioning with the smallest prediction residue is used to encode the output stream. The fact that ME, MC, and quarter pixel approximation are performed for all the different MB and sub-MB partitioning sizes, places a huge computational burden on the encoding process.

To further illustrate the consequence of such a strategy, Fig. 4.2 shows a breakdown of the computation time taken by different inter and intra prediction schemes. This pie chart represents 80% of the total encoding time, with the exclusion of any encoding and prediction post-processing. One can observe that each of the inter prediction modes takes up a considerable and almost equal computation fraction. Each of these inter prediction schemes constructs the prediction residue of a MB with a particular partitioning size through ME search, MC, and quarter pixel approximation. This computation could be reduced by predetermining the MB partitioning coding mode, such that only one of the inter prediction schemes is employed. Predetermining whether a MB is inter or intra coded would also reduce the overall computation time since the inter and intra prediction schemes are compared for each processed MB.
Fig. 4.1 Breakdown of computation time for different encoding process.

Fig. 4.2 Breakdown of computation time for different encoding prediction process.
4.2 Proposed Temporal Transcoder

4.2.1 Transcoder Architecture

In our proposed design, we consider using a closed-loop architecture over an open-loop architecture. A closed-loop architecture has the advantage of being unsusceptible to drift. The added complexity of having the input and output streams coded in different formats also makes it necessary to use a closed-loop scheme, since the transcoder has to convert a coded stream from one format to another, which would usually require the use of two independent loops that would operate at different rates (only possible with closed-loop).

When transcoding between two different standards, we are granted the option of using either a transform domain or a pixel domain architecture. A transform domain architecture would require that MC is performed in the transform domain rather than the pixel domain. Even though eliminating forward and inverse transforms would result in reducing the computation time of the transcoding process, it is important to point out that the forward transform was shown to take only a small fraction of the total encoding time. There is also an associated increase in complexity and lack of flexibility associated with having to perform all transcoding operations in the transform domain [2]. It makes the prospect of performing any form of refinement to the MVs considerably more difficult, since a mini ME search would have to be performed in the transform domain [16]. Another disadvantage is that a transform domain transcoder is still susceptible to drift.

Our implementation uses a pixel-domain transcoder that makes use of MB coding modes and MVs that are extracted from the input stream. Since H.264/SVC is standardized as an extension of H.264/AVC [25], they both have a similar syntax. Hence the same MB coding modes are used to encode the output stream. Due to the fact that an H.264/SVC stream would have a different referencing structure, a new set of MVs is computed from the input stream MVs, such that the MVs could be mapped to the referencing structure used by the input H.264/SVC stream.

A schematic of the discussed temporal transcoder is illustrated in Fig. 4.3. The top part is identical to the H.264 decoder discussed in chapter 3, while the bottom part resembles a modified H.264 encoder with the ME component replaced with a motion vector refinement component. A motion vector adjustment component is placed in between the decoder and modified encoder.
An incoming AVC video stream is first completely decoded, and the MB modes and MVs are extracted. The extracted MVs are used by the motion vector adjustment component (shaded in gray between the decoder and encoder) to compute a set of new MVs that adhere to the SVC hierarchical referencing structure. The extracted MB modes are re-employed by the modified encoder through selecting the partitioning sizes for the MBs in each output frame. The MV refinement component subjects each newly computed MV to a refinement process that consists of a ME search with a small search window. Finally, the encoder has the choice between using either the prediction (from inter prediction) constructed from the transoded information or regular intra prediction. A more detailed discussion about the frame referencing structure mapping and MV re-estimation is presented in the subsequent sections.

The reuse of MB modes accounts for the majority of reduction in computational complexity. This can be deduced from time profiling results shown in section 4.1, where each of the inter prediction modes accounts for a fraction of the total prediction time. The additional use of computed MVs would provide further reduction in computation time by substituting for the ME search process.

### 4.2.2 Frame Referencing Structure

In order to support temporal scalability, SVC employs a hierarchical prediction structure that partitions the frames in each group of pictures (GOP) into a temporal base layer and one or more enhancement layers. In SVC, a temporal layer identifier $T$ is used to identify the layer number. There are two hierarchical prediction structures that are commonly used by SVC, a hierarchical B-frame structure and a hierarchical zero-delay structure [25].

In a hierarchical B-frame structure, the enhancement layer frames are coded as B frames. The reference lists (for each frame) containing the frames that are used for motion estimation, list 0 and list 1, are restricted to the temporally preceding and succeeding frames with a smaller temporal layer identifier $T$ [25]. On the other hand in a hierarchical zero-delay structure, enhancement layer frames are coded as P frames. Referencing is restricted to the temporally preceding frame that has a smaller temporal layer identifier $T$ (list 0 only) [25].

Fig. 4.4 and Fig. 4.5 illustrate a hierarchical B-frame structure and a hierarchical zero-delay structure for a size 8 GOP, respectively. The frames are labeled according to frame coding type (P or B) and coding order. The numbers on the second line indicate the tempo-
Fig. 4.3 Detailed schematic of H.264/AVC to H.264/SVC temporal transcoder.
The referencing structure is represented by the arrows pointing from one frame to another, where the arrows are directed (start) from the reference frames and are terminated on the predicted frames. It is important to note that the list 0 referencing (forward) is identical for both hierarchical referencing structures. The hierarchical B-frame structure has additional list 1 references (backward) that are used for enhancement layer frames.

![SVC hierarchical B-frame structure](image)

**Fig. 4.4** SVC hierarchical B-frame structure. The frames are placed in display order. The first row is labeled with the coding type (P or B) and the coding order. The numbers on the second row specify the temporal layer identifier $T$.

Video streams coded with the H.264/AVC standard have a large variety of different frame referencing structures. AVC referencing structures can vary from simple IPPP structures to complex hierarchical forms with B-coded frames. In our implementation, we focus our attention to AVC video streams that are coded with an IPPP structure, since they form the simplest kind and are the most common. IPPP structures use a single reference per frame, that being the temporally preceding frame. This results in a referencing structure in which the coding order is equal to the temporal (or display) order. An example of an IPPP structure is illustrated in Fig. 4.6, where the frames are labeled according to frame coding type (P) and coding order. The referencing structure is represented by the arrows pointing from one frame to another, where the arrows are directed (start) from the reference frame and terminate on the predicted frame.
In our implementation, temporal transcoding of H.264/AVC to H.264/SVC would require that the IPPP referencing structure is mapped to one of the SVC hierarchical referencing structures discussed earlier. This mapping of referencing structures would translate to using MVs from the input AVC stream (that follow the IPPP referencing structure) to compute a set of new MVs (that follow a hierarchical structure) for the SVC stream.

Each new SVC MV is formed from a linear combination of a set of the old AVC MVs. We thus make the assumption that the motion of an object found within a video scene is linear between temporally consecutive frames. This is such that we can describe the movement of an object through a series of consecutive frames with a single MV. We also consider the fact that both the hierarchical B-frame and zero-delay structures have identical list 0 referencing for a given GOP size, hence they would require the same mapping.

In our implementation, list 1 MVs are not computed for the hierarchical B-frame referencing structures. Recall that for the SVC hierarchical B-frame referencing structures, list 1 MVs use temporally succeeding frames as references (backward MVs). The input AVC MVs only use the temporally preceding (forward MVs) frames as references (only contain a list 0). Thus no backward MVs are available in the input stream, such that they may be used to compute a set of new backward MVs for list 1 of the output SVC stream. In our implementation, we resort to setting the SVC list 1 MV components to zero. These zero MVs are further subjected a refinement process.

The MV mapping is done as follows. For every set of $N$ frames, where $N$ is the output SVC stream GOP size, we need to compute a set of forward (list 0) MVs by using the MVs of the temporally corresponding (with same temporal number) set of $N$ input AVC frames. T. Shanableh and M. Ghanbari suggest a temporal transcoding method that computes a linear relation between the input and output sequence MVs [13]. J. Xin, M.-T. Sun, and K. Chun also suggest a similar method [30]. These methods are used for the older MPEG-1 and MPEG-2 standards, but can be re-applied to the H.264 standard. This is due to the fact that ME prediction in both MPEG standards, and H.264 rely on similar principals. An output sequence SVC MV between any two frames that are at temporal locations $i$ and $j$ (within a GOP), is computed by adding input sequence AVC MVs of frames that are between temporal locations $i$ and $j$.

To better illustrate this scheme, consider the case where the output stream is coded with a GOP size of 8. We need to map the AVC (input) sequence referencing structure illustrated
Fig. 4.5  SVC hierarchical zero-delay structure. The frames are placed in display order. The first row is labeled with the coding type (P) and the coding order. The numbers on the second row specify the temporal layer identifier $T$.

Fig. 4.6  AVC IPPP referencing structure. The frames are placed in display order. The row bellow is labeled with the coding type (P) and the coding order.
in Fig. 4.6 to the list 0 (output) SVC referencing structure illustrated in Fig. 4.5\(^2\). The MV between any two frames \(P_i\) and \(P_j\) in the output sequence, where \(i\) and \(j\) represent the temporal frame id(position), is computed by adding all the input AVC MVs between the frames at \(P_i\) to \(P_j\). For example, the MV from frame \(P_4\) to \(P_6\) \((P_4 \rightarrow P_6)\) in the output SVC stream in Fig. 4.5 can be computed by adding the MV \(P_4 \rightarrow P_5\) with the MV \(P_5 \rightarrow P_6\) from the input AVC sequence in Fig. 4.6. The mapping for the output SVC MVs is summarized in Table 4.1. Note that in our implementation, the exact same mapping would be performed for both cases of transcoding an IPPP AVC stream to a hierarchical B-frame and a zero-delay structure, since we omit the mapping of list 1 MVs.

### Table 4.1 List 0 Motion Vector Mapping from AVC to SVC Stream.

<table>
<thead>
<tr>
<th>Output Stream SVC Motion Vector (From Fig. 4.5)</th>
<th>Input Stream AVC Motion Vector(s) (From Fig. 4.6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_0 \rightarrow P_1)</td>
<td>(P_0 \rightarrow P_1)</td>
</tr>
<tr>
<td>(P_0 \rightarrow P_2)</td>
<td>(P_0 \rightarrow P_1 + P_1 \rightarrow P_2)</td>
</tr>
<tr>
<td>(P_2 \rightarrow P_3)</td>
<td>(P_2 \rightarrow P_3)</td>
</tr>
<tr>
<td>(P_0 \rightarrow P_4)</td>
<td>(P_0 \rightarrow P_1 + P_1 \rightarrow P_2 + P_2 \rightarrow P_3 + P_3 \rightarrow P_4)</td>
</tr>
<tr>
<td>(P_4 \rightarrow P_5)</td>
<td>(P_4 \rightarrow P_5)</td>
</tr>
<tr>
<td>(P_4 \rightarrow P_6)</td>
<td>(P_4 \rightarrow P_5 + P_5 \rightarrow P_6)</td>
</tr>
<tr>
<td>(P_6 \rightarrow P_7)</td>
<td>(P_6 \rightarrow P_7)</td>
</tr>
<tr>
<td>(P_0 \rightarrow P_8)</td>
<td>(P_0 \rightarrow P_1 + P_1 \rightarrow P_2 + P_2 \rightarrow P_3 + P_3 \rightarrow P_4) + (P_4 \rightarrow P_5 + P_5 \rightarrow P_6 + P_6 \rightarrow P_7 + P_7 \rightarrow P_8)</td>
</tr>
</tbody>
</table>

The transcoder is expected to perform better in terms of compression efficiency for hierarchical zero-delay structures, since they only consist of the computed list 0 MVs. The hierarchical B-frame structure on the other hand, consists of both list 0 and list 1 MVs, with only the list 0 MVs directly computed.

\(^2\)We consider the zero-delay case since it consists of only list 0 motion vectors. The same discussion applies to B-frame structures as well.
4.2.3 Motion Vector Re-estimation

The main problem with computing a set of new MVs through addition, is that some of the input sequence MVs cannot be directly extracted. Recall that the H.264 standard implements a block based inter prediction scheme with MB partition sizes ranging from $16 \times 16$ down to $4 \times 4$, along with one MV coded per partition [24]. Each MV points to an area in the reference frame that is of the same size as the associated partition. This block area in the reference frame might not be exactly aligned with other blocks in the reference frame. An example of such a scenario is illustrated in Fig. 4.7, where a $16 \times 8$ block area overlaps with 4 MBs, each with a particular partitioning size. Note that a problem similar to this was discussed in Chapter 2.

![Fig. 4.7 An $8 \times 16$ block area overlapping with 4 other macroblocks of different partitioning sizes.](image)

In order to trace a new MV from the reference frame, the MVs from the overlapped reference blocks should be used. We make use of a method called forward dominant vector selection (FDVS) suggested by J. Youn, M. -T. Sun, and C. -W. Lin which is simple to implement and surpasses other methods such as bilinear interpolation in terms of performance [22]. FDVS devises a method to select a MV based on the greatest area of overlap. As we have discussed, the current MV points to a block sized area in the reference frame, which might not be exactly aligned with other reference blocks. This area can overlap with several other reference blocks, each with an associated MV. One of those MVs is selected based on the greatest area of overlap. An example of FDVS is illustrated in Fig. 4.8, where a MV from a MB in the current frame points to an overlapping area in the reference frame. This area overlaps with four MBs in the reference frame, each with an associated MV.
The MV of the reference MB at the bottom right corner is chosen, since that MB has the greatest overlapping area.

![Fig. 4.8 Example of the forwards dominant selection (FDVS) method.](image)

FDVS falls short when it comes to the H.264 standard, since it does not take into consideration that both the current block and the overlapped blocks could be of different sizes (Fig. 4.7). I. Shin, Y. L. Lee, and H. W. Park suggest a method to handle different block sizes, specifically by decomposing the current block and all the overlapped blocks into $4 \times 4$ blocks [31]. The MVs would also be decomposed, such that an $M \times N$ block with a particular MV, when decomposed would have the same MV for each $4 \times 4$ block in terms of the relative displacement. The FDVS method would then be applied for each $4 \times 4$ block to trace the target MV. Once the MVs have been traced, the MVs of each combination of $4 \times 4$ blocks are combined to the original block size by taking the mean of the MVs.

A schematic for this process is illustrated in Fig. 4.9, where an $8 \times 8$ MB partition with an associated MV (at the top left corner) is shown. This MB partition is first decomposed to four corresponding $4 \times 4$ MB partitions, each with an equivalent MV in terms of displacement. Each of these four partitions is subsequently evaluated by tracing a new MV. Once all MVs have been traced for a given block, all the $4 \times 4$ MB partitions are combined to form the original MB partitioning. The MVs of the four $4 \times 4$ MB partitions are averaged to form a new MV for the original $8 \times 8$ MB partition. There exist several other methods in which one can derive a resulting MV from these four partitions. Nevertheless, the simplicity of this scheme and its applicability to our pixel domain transcoder architecture makes it an attractive choice.

In order to achieve a higher compression efficiency (PSNR/bit-rate) [13], the computed MVs are subjected to a search refinement that is performed using the diamond search
algorithm implemented in the reference software with a search range of 3 pixels [28]. The refinement is similar to a small motion estimation process, where the computed MV would be chosen as starting point of the search. The refinement process is expected to yield considerable gain in peak signal-to-noise ratio (PSNR).

Fig. 4.9  Macroblock and motion vector decomposition, evaluation, and combining.

4.3 Simulation Setup and Results

In this section, we present simulation results of our proposed temporal transcoder of H.264/AVC to H.264/SVC.

4.3.1 Simulation Setup

In order to validate the performance of our proposed temporal transcoder with experimental results, it is necessary to compare it to a reference performance. This reference performance can be obtained with the use of a cascaded decoder-encoder setup. The decoder-encoder setup serves as a high quality and computationally expensive method to convert an existing H.264/AVC stream to an H.264/SVC stream with temporal scalability. The main objective of our implementation is to obtain an output visual quality that is as close as possible to the decoder-encoder setup, while reducing the computation time as much as possible.
The proposed temporal transcoder described in the previous section uses a number of different schemes to reduce the computational complexity of the encoding process. MB modes are extracted from the incoming AVC stream and are reused to encode the output SVC stream. A new set of MVs that adhere to the SVC referencing structure are computed from the old extracted MVs. These newly computed MVs are subjected to a 3 pixel refinement for better prediction accuracy.

In order to better validate the effectiveness and the contribution of each of these schemes, it is necessary to show the influence that each of these techniques has on the overall performance of our proposed transcoder. These schemes are summarized in Table. 4.2. All the described schemes perform a full decoding of the incoming AVC bit stream. Note that list 1 MVs are computed through ME for the Decoder-Encoder and Transcoder-Mode schemes. They are set to zero for the Transcoder scheme. For the other transcoding schemes they are set to zero and subjected to a 3 pixel refinement.

The Decoder-Encoder represents a cascaded decoder-encoder setup that performs a full re-encoding to form a temporally scalable SVC stream. It performs a full ME fast search with a 32 pixel range to compute a set of new MVs. The Transcoder-Mode is a transcoding scheme that extracts MB modes from the incoming AVC stream and reuses them to encode the output stream. It also performs a full ME fast search with a 32 pixel search range to compute the MVs. The Transcoder scheme extracts MB modes and computes a set of new MVs from the old input MVs, all to be reused to encode the output SVC stream. The Transcoder-Refine scheme takes the Transcoder scheme to the next step by performing a refinement on the newly computed MVs with a search window of 3 pixels. The Transcoder-Zero on the other hand, extracts the MB modes and reuses them, but resets all MVs to zero and subjects these zero MVs to a 3 pixel refinement.

The input test sequences consist of a single reference AVC IPPP stream that is encoded at a frame rate of 30Hz with 81 frames for both QCIF (176 by 144) and CIF (352 by 288) resolutions. Sequences with limited motion, such as Foreman and Coastguard were tested along with sequences containing intensive motion, such as Football and NBA [29].

Our implementation uses the default settings of the reference software JVSM 9.16 [28]. Both the H.264/SVC hierarchical B-frame and zero-delay structures were tested, for GOP sizes of 8 and 16. The transcoding schemes were simulated with standard (equal weights) weighted prediction.
### Table 4.2

Different transcoding schemes and their properties. Indicates if the coding modes are reused, how list 0 MVs are obtained, and if MV refinement is applied.

<table>
<thead>
<tr>
<th>Scheme/ Parameter</th>
<th>MB Coding Modes</th>
<th>List 0 MV Computation</th>
<th>MV Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoder-Encoder</td>
<td>No</td>
<td>Fast 32 pixel ME search</td>
<td>No</td>
</tr>
<tr>
<td>Transcoder-Mode</td>
<td>Yes</td>
<td>Fast 32 pixel ME search</td>
<td>No</td>
</tr>
<tr>
<td>Transcoder</td>
<td>Yes</td>
<td>New MVs computed from old input MVs</td>
<td>No</td>
</tr>
<tr>
<td>Transcoder-Refine</td>
<td>Yes</td>
<td>New MVs computed from old input MVs</td>
<td>3 Pixel search window</td>
</tr>
<tr>
<td>Transcoder-Zero</td>
<td>Yes</td>
<td>MVs components are set to 0</td>
<td>3 Pixel search window</td>
</tr>
</tbody>
</table>

### 4.3.2 Simulation Results

The compression efficiency of all of the different transcoding schemes was tested by plotting the rate-distortion (RD) curves independently for each scheme. The RD curves consist of the average Y-PSNR (Y being the luminance component) against the coded bit-rate. The computational complexity was measured by calculating the computation time of each of the transcoding schemes as a percentage of the time taken by the *Decoder-Encoder* scheme.

**Compression Efficiency**

The compression efficiency of our implemented schemes was tested for both hierarchical B-frame and zero-delay structures.

**Hierarchical B-Frame Structures**

The QCIF and CIF Foreman (low motion) RD curves for a hierarchical B-frame structure with a GOP size of 8 are shown in Fig. 4.10 and Fig. 4.11, respectively. The QCIF and CIF NBA (high motion) RD curves for a hierarchical B-frame structure with a GOP size
of 8 are shown in Fig. 4.12 and Fig. 4.13, respectively.

The **Decoder-Encoder**, represents the optimum transcoding performance, and the gap it has with the **Transcoder-Mode** scheme represents the loss in compression efficiency resulting from the reuse of MB coding modes (see Table. 4.2). The other implemented transcoding schemes also employ MB coding mode reuse, but would instead compute a new set of MVs from the old extracted MVs. The **Transcoder-Mode** scheme would thus represent the optimum transcoding performance that could be achieved by any of the other transcoding schemes. This is due to the fact that the **Transcoder-Mode** scheme also reuses MB coding modes, but would perform a full ME search to compute any MVs.

It is generally observed that the refinement process in both the **Transcoder-Refine** and **Transcoder-Zero** schemes provides a significantly better performance over the **Transcoder** scheme. For both QCIF and CIF resolutions of the Foreman sequence (Fig. 4.10 and Fig. 4.11), the RD curves for both the the **Transcoder-Refine** and **Transcoder-Zero** schemes can be seen to be almost identical in performance. This is due to the fact that Foreman represents a slow motion sequence, such that the computing of new MVs does not provide a significant gain.

Table. 4.3 and Table. 4.4 show the percentage of motion vectors that are smaller than or equal to 3 pixels for QCIF and CIF sequence encoded by an SVC encoder, respectively. These sequences use a hierarchical B-frame structure and are encoded at a frame rate of 30Hz with 81 frames. They are encoded using fast search with a 32 pixel search range.

For the Foreman sequence, it can be observed that the majority of MVs are found to have sizes smaller than 3 pixels, indicating that the 3 pixel refinement is sufficient to produce an accurate MV from a zero starting point. On the other hand for a motion intensive sequence such as NBA, a smaller percentage of MVs have sizes that are smaller than 3 pixels. This fact is better portrayed when observing the QCIF and CIF RD curves for the NBA sequence in Fig. 4.12 and Fig. 4.13, respectively. It can be seen that the **Transcoder-Refine** scheme has a better performance than the **Transcoder-Zero** scheme. The computing of new MVs with the addition of a refinement process provides approximate gains of 0.4dB and 1dB for QCIF and CIF resolutions, respectively.

The performance of the **Transcoder-Refine** scheme with respect to the optimum **Decoder-Encoder** scheme is observed to be highly dependant on the relative motion and frame resolution of the tested video sequence. The slow motion Foreman sequence is observed to produce a relative performance that is closer to the optimum than the fast motion NBA
sequence.

For both the Foreman and NBA sequences, QCIF simulations show a performance that is closer to the optimum than the CIF sequence. The difference in performance between different resolutions of the same sequence can be accounted for by the fact that CIF sequences have twice the resolution of QCIF sequences with MVs that are approximately scaled by two. Thus a larger percentage of MVs with a size larger than 3 pixels is observed (see Table 4.3 and Table 4.4), which thereby decreases the accuracy of our MV computation and refinement scheme with respect to the Decoder-Encoder scheme.

The GOP size also has an effect on the performance of different transcoding schemes. An increase in the GOP size decreases the accuracy of the MV computation and refinement method, hence larger performance gaps are observed. This can be accounted for by the fact that an increase in the size of the GOP changes the hierarchical referencing structure, such that there is a larger temporal distance between frames and their references. This larger distance decrease the performance of our MV computation method, since it is performed on a longer span of frames. This is evident when observing the RD curves for a GOP of size 16 and QCIF resolution in Fig. 4.14 and Fig. 4.15 for the Foreman and NBA sequences respectively, and comparing them to their GOP 8 counterparts in Fig. 4.10 and Fig. 4.12. Consequently, any decrease in GOP size would also decrease the size of the performance gaps between different schemes.

Hierarchical Zero-Delay Structures

The QCIF and CIF Foreman RD curves for a hierarchical zero-delay structure with a GOP size of 8 are shown in Fig. 4.16 and Fig. 4.17, respectively. The QCIF and CIF NBA RD curves for a hierarchical B-frame structure with a GOP size of 8 are also shown in Fig. 4.18 and Fig. 4.19, respectively.

When comparing the different transcoding schemes to their hierarchical B-frame counterparts (see Fig. 4.11-4.13), a similar trend is observed in terms of the performances for both low motion and high motion sequences, as well as for different frame resolutions. The Transcoder-Refine scheme is observed to perform much closer to the optimum Decoder-Encoder for the zero-delay structures. This is accounted for by the fact that all of the list 0 MVs are computed, unlike the hierarchical B-frame structure, where the list 1 MV components are set to zero.
Fig. 4.10  Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical B-frames structure with a GOP size of 8.

Fig. 4.11  Plot of PSNR/Bitrate of CIF Foreman for a hierarchical B-frames structure with a GOP size of 8.
Fig. 4.12  Plot of PSNR/Bitrate of QCIF NBA for a hierarchical B-frames structure with a GOP size of 8.

Fig. 4.13  Plot of PSNR/Bitrate of CIF NBA for a hierarchical B-frames structure with a GOP size of 8.
Table 4.3  Decoder-Encoder percentage of motion vectors smaller than or equal to 3 pixels for QCIF.

<table>
<thead>
<tr>
<th>Sequence/Vector</th>
<th>List 0</th>
<th></th>
<th>List 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal (%)</td>
<td>Vertical (%)</td>
<td>Horizontal (%)</td>
<td>Vertical (%)</td>
</tr>
<tr>
<td>Foreman</td>
<td>94</td>
<td>97</td>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>Coastguard</td>
<td>86</td>
<td>95</td>
<td>92</td>
<td>96</td>
</tr>
<tr>
<td>Football</td>
<td>92</td>
<td>93</td>
<td>93</td>
<td>95</td>
</tr>
<tr>
<td>NBA</td>
<td>69</td>
<td>83</td>
<td>71</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 4.4  Decoder-Encoder percentage of motion vectors smaller than or equal to 3 pixels for CIF.

<table>
<thead>
<tr>
<th>Sequence/Vector</th>
<th>List 0</th>
<th></th>
<th>List 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Horizontal (%)</td>
<td>Vertical (%)</td>
<td>Horizontal (%)</td>
<td>Vertical (%)</td>
</tr>
<tr>
<td>Foreman</td>
<td>86</td>
<td>90</td>
<td>91</td>
<td>95</td>
</tr>
<tr>
<td>Coastguard</td>
<td>76</td>
<td>94</td>
<td>76</td>
<td>94</td>
</tr>
<tr>
<td>Football</td>
<td>56</td>
<td>72</td>
<td>53</td>
<td>72</td>
</tr>
<tr>
<td>NBA</td>
<td>49</td>
<td>52</td>
<td>53</td>
<td>69</td>
</tr>
</tbody>
</table>
Fig. 4.14 Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical B-frames structure with a GOP size of 16.

Fig. 4.15 Plot of PSNR/Bitrate of QCIF NBA for a hierarchical B-frames structure with a GOP size of 16.
Fig. 4.16  Plot of PSNR/Bitrate of QCIF Foreman for a hierarchical zero-delay structure with a GOP size of 8.

Fig. 4.17  Plot of PSNR/Bitrate of CIF Foreman for a hierarchical zero-delay structure with a GOP size of 8.
4 Temporal Transcoding of H.264/AVC to H.264/SVC

Fig. 4.18  Plot of PSNR/Bitrate of QCIF NBA for a hierarchical zero-delay structure with a GOP size of 8.

Fig. 4.19  Plot of PSNR/Bitrate of CIF NBA for a hierarchical zero-delay structure with a GOP size of 8.
Computational Complexity

The computational complexity was tested for each of the different transcoding schemes for both the hierarchical B-frame and zero-delay structures. An average among the Foreman, Coastguard, Football, and NBA sequences was computed with a GOP of size 8. The average computational complexity for sequences with a QCIF resolution for hierarchical B-frame and zero-delay structures is illustrated in Fig. 4.20 and Fig. 4.21, respectively. The average computational complexity for sequences with a CIF resolution for hierarchical B-frame and zero-delay structures is illustrated in Fig. 4.22 and Fig. 4.23, respectively.

For the QCIF case, the reuse of MB coding modes in the Transcoder-Mode scheme is observed to reduce the computation time by 48.8% and 28.1%, for the B-frame and zero-delay structures, respectively. The additional computation of new MVs in the Transcoder scheme saves an additional 7.4% and 6.1% in cost. The 3 pixel refinement in the Transcoder-Refine schemes approximately adds an extra 1% of cost for both hierarchical structures. This results in an overall reduction in computation time of 55.2% for B-frame structures, and 33.5% for the zero-delay structures.

The CIF case performs slightly better for both hierarchical structures. The reuse of MB coding modes in the Transcoder-Mode scheme is observed to reduce the computation time by 55.5% and 34.7%, for the B-frame and zero-delay structures, respectively. The additional computing of new MVs in the Transcoder scheme saves an additional 10.1% and 6.8% in cost. The 3 pixel refinement in the Transcoder-Refine scheme adds an extra 3.5% and 1% of cost. This results in an overall reduction in computation time of 62.1% for B-frame structures, and 40.5% for the zero-delay structures.

It is observed that for the different hierarchical structures and the different frames resolutions, there is only a minimal increase in computational complexity associated with the 3 pixel refinement. The refinement process was observed to add a maximum of 3.5% in computation time and yet has shown to add a considerable gain in PSNR for all the different cases. We can thus conclude that the use of a 3 pixel refinement is essential to the transcoding process, since it results in a considerable gain in PSNR while adding only a negligible increase computation time.

The transcoding schemes show a much better computational performance for the Hierarchical B-frame over the hierarchical zero-delay structures, with a different of over 20% for both QCIF and CIF cases. This can be accounted for by the fact that in the B-frame case
the encoder has to choose between using one list 0 MV, or one list 1 MV, or bi-prediction with both list 0 and list 1 MVs. On the other hand in the zero-delay case, the encoder only has the choice of using a list 0 MV. Thus in the B-frame case, a larger share of the encoding time is spent in forming a prediction with respect to the time spent on intra prediction or prediction post-processes such as quantization and transform. Hence it is expected that the transcoding process for the B-frame case would be more computationally efficient than the zero-delay case.

It can also be observed that the CIF case performs slightly better than the QCIF case, with a difference of over 7% for both hierarchical structures. This is due to the fact that each CIF resolution frame has four times the number of MBs that a QCIF frame has. Hence a slightly larger share of the encoding time is spent in forming a prediction for a CIF frame, rather than other encoding overhead. CIF sequences would also require a more extensive search to find the best prediction due to the greater detail found in the frames, which would result in making the transcoding mechanism more efficient by further reducing the computational complexity.

The computational complexity plots in Fig. 4.20 to Fig. 4.23 show the averages over different sequences. No variations in the complexity plots where observed for other GOP sizes, nor for longer or shorter sequences, since encoding is done on a frame by frame basis. However, it was observed that for low resolution QCIF sequences, Foreman, Coastguard, and NBA only deviated from each other by around 1%, while Football on the other hand differed by 2-3%. Similarly for the high resolution CIF sequences, Foreman, Coastguard, and NBA only deviated from each other by around 2-4%, while Football on the other hand differed by up to 7%. Football was observed to have a lower computational complexity with respect to the other three sequences. This can be explained by the nature of the Football sequence, which requires more time for ME search, and hence the implemented transcoding mechanism would further reduce the computational complexity. This is evident when observing Table. 4.5 to Table. 4.8, where Football always took the longest encoding time. CIF sequences were observed to deviate more in terms of computational time, simply because they would require a more extensive search to find the best prediction due to the greater detail found in the frames.

When observing Table. 4.5 to Table. 4.8, it is evident that sequences with larger motion require a higher encoding time. It can be seen that low motion sequences like Foreman and Coastguard are comparably close, while high motion sequences such as Football and NBA
require a noticeably larger encoding time. This occurs because sequences with high motion require a more extensive search to find the optimal prediction. It is also observed that the required transcoding time for the different sequences is very close, which is attributed to the fact that the transcoding mechanism eliminates the ME search overhead, and requires only a 3 pixel refinement.

4.4 Summary

In this chapter, a novel implementation of a temporal transcoder of H.264/AVC to H.264/SVC was presented. A time profiling of the SVC encoder was performed to categorize the computational complexity of temporal encoding. A detailed discussion of our proposed implementation was presented, together with simulation results to validate the performance of our proposed scheme. It was shown that a significant decrease in computational complexity is achieved with a reduction in computation time (For high resolution CIF frames) of over 60% for B-frame structures, and 40% for the zero-delay structures. This high reduction in computation time was achieved while maintaining a small loss in compression efficiency. In the next chapter, we further extend our proposed scheme to provide support of spatial scalability as well. We propose a transcoder implementation that provides combined spatial-temporal scalability to an existing H.264/AVC stream.

<table>
<thead>
<tr>
<th>Sequence/ Scheme</th>
<th>Decoder-Encoder (Seconds)</th>
<th>Transcoder-Refine (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>25.66</td>
<td>11.92</td>
</tr>
<tr>
<td>Coastguard</td>
<td>26</td>
<td>11.79</td>
</tr>
<tr>
<td>Football</td>
<td>30.88</td>
<td>13.11</td>
</tr>
<tr>
<td>NBA</td>
<td>29.03</td>
<td>13.25</td>
</tr>
</tbody>
</table>

Table 4.5 Average coding time taken by different schemes for QCIF sequences encoded with a B-frame structure.
Table 4.6  Average coding time taken by different schemes for CIF sequences encoded with a B-frame structure.

<table>
<thead>
<tr>
<th>Sequence/Scheme</th>
<th>Decoder-Encoder (Seconds)</th>
<th>Transcoder-Refine (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>108.4</td>
<td>43.83</td>
</tr>
<tr>
<td>Coastguard</td>
<td>108</td>
<td>39.26</td>
</tr>
<tr>
<td>Football</td>
<td>133.4</td>
<td>42.93</td>
</tr>
<tr>
<td>NBA</td>
<td>112</td>
<td>44.06</td>
</tr>
</tbody>
</table>

Table 4.7  Average coding time taken by different schemes for QCIF sequences encoded with a zero-delay structure.

<table>
<thead>
<tr>
<th>Sequence/Scheme</th>
<th>Decoder-Encoder (Seconds)</th>
<th>Transcoder-Refine (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>14.46</td>
<td>9.78</td>
</tr>
<tr>
<td>Coastguard</td>
<td>14.87</td>
<td>9.98</td>
</tr>
<tr>
<td>Football</td>
<td>17.2</td>
<td>11</td>
</tr>
<tr>
<td>NBA</td>
<td>16.36</td>
<td>11.04</td>
</tr>
</tbody>
</table>

Table 4.8  Average coding time taken by different schemes for CIF sequences encoded with a zero-delay structure.

<table>
<thead>
<tr>
<th>Sequence/Scheme</th>
<th>Decoder-Encoder (Seconds)</th>
<th>Transcoder-Refine (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>55.32</td>
<td>34.34</td>
</tr>
<tr>
<td>Coastguard</td>
<td>61.64</td>
<td>37.77</td>
</tr>
<tr>
<td>Football</td>
<td>73.7</td>
<td>38.99</td>
</tr>
<tr>
<td>NBA</td>
<td>62.99</td>
<td>38.76</td>
</tr>
</tbody>
</table>
Fig. 4.20 Average computation time for QCIF streams encoded with a hierarchical B-frames structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder scheme.

Fig. 4.21 Average computation time for QCIF streams encoded with a hierarchical zero-delay structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder scheme.
Fig. 4.22  Average computation time for CIF streams encoded with a hierarchical B-frames structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder scheme.

Fig. 4.23  Average computation time for CIF streams encoded with a hierarchical zero-delay structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder scheme.
Chapter 5

Combined Spatial-Temporal Transcoding of H.264/AVC to H.264/SVC

Our discussion so far has been centralized around the concept of temporal scalability. Our proposed implementation in the previous chapter focuses on providing an existing H.264/AVC stream with temporal scalability through the use of a temporal transcoder of H.264/AVC to H.264/SVC. This temporal transcoder performs a modification of the input stream frame referencing structure, followed by a partial re-encoding of the output stream to the desired H.264/SVC format. This temporal transcoder could be further extended to incorporate spatial scalability as well.

Spatial scalability provides an encoded video bit stream with the ability to dynamically support different bit-rates and spatial resolutions on the fly. The spatial dimensions could easily be adjusted by simply discarding parts of the encoded stream, thus reducing the overall bit-rate as well. This provides the bit stream with the flexibility to adjust according to any network constraints and requirements. SVC provides spatial scalability to streams encoded with the H.264 syntax by incorporating different resolutions into the same coded stream.

Combined spatial-temporal scalability can be added to a single layer H.264/AVC stream by extending our discussed temporal transcoder to support spatial scalability as well. The implemented scheme would be extended by downsizing the decoded high resolution video
frames such that they could be used to encode lower resolution layers. In addition, the extracted and modified (by the temporal transcoding scheme) coding and motion information is further downsampled and reused to encode the lower spatial layers.

In this chapter, a novel implementation of a combined spatial-temporal transcoder of H.264/AVC to H.264/SVC is presented. A time profiling of the SVC encoder is performed to categorize the computational complexity of combined spatial-temporal scalability encoding. We also discuss some the most important technical details that are involved in the transcoding process. Finally, we present some experimental simulation of our implementation. Similarly to the temporal transcoder case, we use the SVC reference software that was developed by the Joint Video Team for simulation purposes [28]. Again, we limit the simulation to only the Foreman video sequence [29], since the variations between different sequences were observed to be negligible.

5.1 Time Profiling of Reference Software

5.1.1 Profiling Method

In the previous chapter, a time profiling was performed on the SVC encoder to categorize the computational complexity of different encoder components. This profiling was performed on a full SVC encoding process with support for temporal scalability.

In order to extend our temporal transcoder implementation, we need to categorize the computational complexity of a combined spatial-temporal encoding process. We thus perform another profiling of the SVC reference software. This profiling provides us with statistics that indicate the fraction of the total encoding time used by different spatial scalability encoding components. This further helps to provide us with a guideline on how to design a low complexity AVC to SVC combined spatial-temporal transcoder.

We make use of the reference SVC encoder software to produce the necessary statistics in a similar manner to the profiling done in the temporal case [28]. The reference encoder is sampled while it performs a full video encoding of an SVC video sequence with spatial-temporal scalability. The profiling simulation uses the default settings of the reference software JVSM 9.16 [28]. It fully encodes an 81 frame video sequence with hierarchical scalability.

\footnote{Again, it is important to note that the reference SVC encoder software that is used might not be optimized. Specific parameters were chosen to provide us with a fast encoding process as a reference. Changing some of these parameters might vary the compression efficiency and encoding time.}
B-frames. Two spatial layers are employed, a spatial base layer with a resolution of 176 by 144 pixels (QCIF) and a spatial enhancement layer with a resolution of 352 by 288 pixels (CIF), both at a 30 Hz frame-rate. The encoding process uses a fast motion estimation search algorithm, with a 32 pixel search range and standard (equal weights) weighted prediction.

Two different spatial prediction schemes are employed, with and without inter-layer prediction. We will refer to the scheme with no inter-layer prediction as Mode 0, and the scheme with inter-layer prediction as Mode 1. Mode 0 encodes each spatial layer independently by relying on temporal inter prediction and intra prediction. In Mode 1 on the other hand, each enhancement spatial layer is completely dependent on the corresponding spatial base layer (or the layer below it) through inter-layer prediction.

5.1.2 Profiling Results

A breakdown of the overall encoding process in terms of the fraction of time taken from the total encoding time is illustrated in Fig. 5.1 and Fig. 5.2 for Mode 0 and Mode 1, respectively. For Mode 0, it can be seen that all of the different prediction types make up 95% of the encoding process. Inter prediction motion estimation (ME) and motion compensation (MC) take up 53% of the encoding process; together with motion vector (MV) quarter pixel approximation, they add up to 71% in total. Intra prediction seems to have a smaller share of the pie with a collective fraction of only 7%. Processes such as transform and quantization, among other operations could be accounted for by the 14% prediction post-processing. When inter-layer prediction is employed, as in the case of Mode 1, it is evident that ME and MC take up a smaller share with 37%. This can be accounted for by the fact that inter-layer prediction significantly reduces the time taken by encoding the spatial enhancement layer. This further results in an increase in the overall overhead prediction to 26%, since the overhead due to the enhancement layer is the same for Mode 0 and Mode 1.

A better illustration of the overall contribution of inter-layer prediction to the encoding process can be seen when observing Fig. 5.3 and Fig. 5.4 for Mode 0 and Mode 1, respect-

---

2We note that a third and optimal prediction scheme that adaptively chooses to employ inter-layer prediction on a MB basis does exist. We do not consider this scheme in our discussion since the associated encoding computation time is substantially higher than the other two schemes, and it does not provide a fast conversion from one standard to another.
5 Combined Spatial-Temporal Transcoding of H.264/AVC to H.264/SVC

Fig. 5.3 shows a breakdown of the overall time taken by the prediction process in terms of the base layer and the enhancement layer. It can be observed that the majority of the computation time is attributed to the enhancement layer encoding, while a much smaller fraction is taken up by the base layer. In Fig. 5.4, a similar breakdown is illustrated and it is evident that almost all of the encoding time is attributed to encoding the base layer.

It is important to note that a significant part of the total encoding process still involves ME, MC, and quarter pixel approximation for both modes. We can thus conclude that in order to significantly reduce the computation time of the encoding process, it is still necessary to reduce to computation time of the ME and MC process. This, in turn, could also possibly reduce the fraction taken up by quarter pixel approximation.

Fig. 5.5 and Fig. 5.6 illustrate a breakdown of the computation time taken by different inter and intra prediction schemes, for Mode 0 and Mode 1, respectively. For both cases, each of the inter prediction modes takes up a considerable and almost equal computational fraction. Each of these inter prediction schemes constructs the prediction residue of a macroblock (MB) with a particular partitioning size through ME search, MC, and quarter pixel approximation. This computation could be reduced through predetermining the MB partitioning coding mode, such that only one of the inter prediction schemes is employed. Predetermining whether a MB is inter or intra coded would also reduce the overall computation time since the inter and intra prediction schemes are compared for each processed MB.

5.2 Proposed Spatial-Temporal Transcoder

5.2.1 Transcoder Architecture

Our discussion of the proposed spatial-temporal transcoder architecture extends the discussion from the previous chapter by incorporating additional components into the design. These components provide the necessary spatial adjustments to decoded frames and the associated extracted coding and motion information. The general architecture of the temporal transcoder discussed in the previous chapter consists of an H.264 decoder, a MV adjustment component, and a modified temporal SVC encoder (with a single spatial layer). This modified encoder reuses the decoded frames together with the modified motion infor-
Fig. 5.1 Breakdown of computation time for different encoding process, with no inter-layer prediction (Mode 0).

Fig. 5.2 Breakdown of computation time for different encoding process, with inter-layer prediction (Mode 1).
Fig. 5.3 Comparing computation total computation time of base and enhancement layer, with no inter-layer prediction (Mode 0).

Fig. 5.4 Comparing computation total computation time of base and enhancement layer, with inter-layer prediction (Mode 1).
Fig. 5.5 Breakdown of computation time for different encoding prediction processes, with no inter-layer prediction (Mode 0).

Fig. 5.6 Breakdown of computation time for different encoding prediction processes, with inter-layer prediction (Mode 1).
information to re-encode the output stream. Such a setup could be extended to support spatial scalability by adding two additional components between the decoder and encoder.

Fig. 5.7 illustrates the layout of our proposed combined spatial-temporal transcoder. The structure assumes a spatial scalability of two layers, a base layer and an additional enhancement layer. It is also assumed that the input is an H.264/AVC IPPP structured stream (single reference) that is coded at the same spatial resolution as the target spatial enhancement layer.

In this transcoder, the input high resolution AVC stream is fully decoded. Since only two spatial layers are encoded, the decoded video stream needs to be downsized such that it may be used as the lower spatial resolution input. This is necessary since the SVC encoder requires an independent video stream input for each spatial layer. The input sequence needs to be at the desired output resolution as well. Thus Fig. 5.7 includes a video frame downsizer component that reduces the spatial resolution of the decoded video frames. These downsized video frames are subsequently used as the input to the base layer SVC encoder. Our implementation makes use of the downsizer provided and recommended by the reference software [28], which uses a method based on the Sine-windowed Sinc-function.

Similarly to the case of the temporal transcoder, MB modes and MVs are extracted from the decoded input stream. These MB modes and MVs (that are associated with the coded high resolution input stream) need to be further downsampled in order to be used for re-encoding the base layer spatial resolution. Fig. 5.7 details this scheme. The MB modes and MVs extracted are processed by a motion vector adjustment component, such that they may adhere to the SVC referencing structure. This adjusted information follows two paths. The first path (upper) feeds the information to the modified enhancement layer encoder, such that it may be used to reduce the computational complexity of encoding the enhancement layer. The second path (lower) subjects the information to additional processing via a macroblock mode and motion vector downsampler component. This processed information is subsequently passed to the modified base layer encoder, such that it may be used to reduce the complexity of encoding the base layer. A more detailed discussion of the functionalities of the downsampler is presented in the next section.

Fig. 5.7 shows a schematic of a two spatial layer SVC encoder. It consists of two main components, a modified base layer encoder and a modified enhancement layer encoder.

Footnote:

3The computation time taken up by the downsizer component is negligible.
Each encoder is used to encode one of the spatial layers. Inter-layer prediction is possibly employed between the base and enhancement layer encoders. The output of each spatial encoder are then combined using multiplexer and are formed into an output SVC stream.

5.2.2 Downsampler

The proposed spatial-temporal transcoder contains a downsampler component, as is illustrated in Fig. 5.7. The main task of this downsampler is to process the MB modes and MVs that are associated with the decoded high resolution frames, such that they could be associated with the downsized low resolution frames. Recall that our proposed spatial-temporal transcoder considers the case of two spatial layers, a spatial base layer and a spatial enhancement layer. In this discussion, we will consider the case of having a (2:1) ratio between the enhancement and base layer resolutions. This is such that an enhancement layer frame has exactly twice the horizontal and vertical spatial resolution of the base layer. The downsampling with such a configuration would result in reducing the associated MB modes and MVs of every four MBs into one MB. The same principals could be used to implement a mechanism that supports multiple spatial layers with a fixed downsizing factor.

In the following, we discuss the problem of reducing the information associated with a single enhancement layer frame. There are two main issues that need to be considered, MB coding modes and MVs.

Macroblock Coding Modes

Recall that in the H.264 standard, MBs can have three different coding types. A MB can either be inter predicted or intra predicted or skipped. Inter prediction employs what can be described as a tree structured motion compensation scheme. Each 16×16 luma MB can be partitioned in four different ways. It can be represented by one 16×16, or two 8×16, or two 16×8, or four 8×8 partitions. If a MB is split to four 8×8 partitions, each 8×8 partition could be further represented by one 8×8, or two 4×8, or two 8×4, or four 4×4 partitions [1]. Intra predicted MBs on the other hand, represents each 16×16 luma MB by one 16×16 or sixteen 4×4 partitions [1]. Note that we only discuss the case for luma MBs since chroma MBs would follow the same methodology.
Fig. 5.7 Detailed schematic of the H.264/AVC to H.264/SVC combined spatial-temporal transcoder.
**MB Coding Types**

When reducing four MBs into one, each of these four MBs can be either inter predicted or intra predicted or skipped. For the case when all four MBs are of the same coding type, as in all four MBs are either inter predicted or intra predicted or skipped, it is straightforward to conclude that the target MB should also be of that same coding type. A problem arises when the four MBs are not all of the same coding type. An example of such a situation is displayed in Fig. 5.8, where a mixture of two inter predicted, one intra predicted, and one skipped MB has to be reduced to a single MB. It is evident that in order to handle such mixed cases priority has to be given for one coding type over the other when choosing the target MB coding type.

![Diagram of four macroblock modes downsampling to one](image)

**Fig. 5.8** Four macroblock modes downsampled to one.

In order to provide an insight on how to choose this priority, Table 5.1 shows the percentage of MBs coded in each coding mode for a number of different H.264/AVC sequences. These sequences are IPPP structured with 81 frames of length and are encoded with a QCIF (176 by 144) resolution. The encoding process uses a fast motion estimation search algorithm, with 32 pixels as a search range, and with standard (equal weights) weighted prediction. These sequences resemble what would be the input to the base layer SVC encoder. The statistics are shown for four different sequences, Foreman, Coastguard, Football, and NBA. These results were provided as a courtesy of Mr. Serdar Burak Solak.

Generally, it is observed in Table 5.1 that for all the different sequences, the inter prediction modes (considering all the partitioning sizes) dominate in terms of the percentage
of coded MBs. Inter predicted MBs are shown to compose a minimum of 62.5\% of the total MBs encoded. This is succeeded by MBs being skipped. Finally, intra predicted MBs (for all partition sizes) are shown to have a negligible share that does not exceed 2\%.

These statistics show that for the case of MBs with mixed coding types priority should be given to inter prediction, followed by skip, followed by intra prediction. We thus devise a method to compute the MB coding type in mixed cases. In the case when the mixture contains at least one inter predicted MB, the target MB is inter predicted, regardless of the other MB types in the mixture. When a MB consists of a mixture of only skip and intra predicted coding types, the target MB is coded as skip. Finally, the only case in which the target MB is intra predicted, is the case when all four MBs in the mixture are intra predicted.

It is important to note that different combinations with different priority orders were tested. The above mentioned methodology proved to provide the best performance.

<table>
<thead>
<tr>
<th>MB Coding Mode/Video Sequence</th>
<th>Foreman %</th>
<th>Coastguard %</th>
<th>Football %</th>
<th>NBA %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip</td>
<td>37.3</td>
<td>21.9</td>
<td>14.9</td>
<td>2.3</td>
</tr>
<tr>
<td>Inter Prediction 16×16</td>
<td>24.9</td>
<td>35.7</td>
<td>12.7</td>
<td>20.4</td>
</tr>
<tr>
<td>Inter Prediction 16×8</td>
<td>10.4</td>
<td>14.4</td>
<td>6.6</td>
<td>7.9</td>
</tr>
<tr>
<td>Inter Prediction 8×16</td>
<td>13.3</td>
<td>13.0</td>
<td>6.8</td>
<td>7.1</td>
</tr>
<tr>
<td>Inter Prediction 8×8</td>
<td>13.9</td>
<td>14.9</td>
<td>57.0</td>
<td>61.7</td>
</tr>
<tr>
<td>Intra Prediction All Partitions</td>
<td>0.1</td>
<td>0.2</td>
<td>2.0</td>
<td>0.5</td>
</tr>
</tbody>
</table>
**MB Partitioning Sizes**

The downsizing of four MBs into one effectively reduces four source 16×16 MBs to one target 16×16 MB. Each of these source 16×16 MBs, whether inter or intra predicted can be partitioned in several different ways. This large variation in different partitioning sizes provides great flexibility, such that source MBs with large partitions can be represented by smaller partitions in the target MB. We make use of a method suggested by P. Zhang, et al [32].

For the intra prediction case, when all the four source MBs are intra predicted with a 16×16 partitioning, the target MB is also intra predicted with a 16×16 partitioning. Otherwise the target MB is always intra coded with a 4×4 partitioning.

For the inter prediction case, each partitioning in each of the four source MBs is downsampled to a partitioning in the target MB. In other words, each 16×16 source MB (with its partitions) is mapped to one of the 8×8 partitions in the target MB. For example a 16×16 source MB is mapped to an 8×8 partition in the target MB, a 16×8 source partition is mapped to a 8×4 partition, and so on. Table 5.2 provides a summary of the downsampling of different partitioning sizes of the source MB to different sizes in the target MB. An exception to the downsampling in Table 5.2 is when all four source MBs are inter predicted with a 16×16 partitioning size, the target MB is also inter predicted with a 16×16 partitioning. The 8×8 partitioning of the source MB in Table 5.2 includes all the different smaller partitioning sizes such as 4×8, 8×4, and 4×4. Any source MB partitionings with a size smaller than 8×8 are also reduced to a 4×4 size in the target MB. In the case of mixed MB coding types, where intra predicted or skipped MBs are present, they are treated as inter coded MBs with 16×16 partitioning sizes.

<table>
<thead>
<tr>
<th>Source MB Partition</th>
<th>Target MB Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>16×16</td>
<td>8×8</td>
</tr>
<tr>
<td>16×8</td>
<td>8×4</td>
</tr>
<tr>
<td>8×16</td>
<td>4×8</td>
</tr>
<tr>
<td>8×8</td>
<td>4×4</td>
</tr>
</tbody>
</table>
Motion Vectors

Each inter predicted MB partition has a MV associated with it. This MV requires down-sampling as well, such that it may be used to re-encode the base layer of the output stream. There are three main cases that need to be considered:

- In the case where the source MB has a partitioning size of 16×16 or 16×8 or 8×16 or 8×8, the target partition would have an 8×8 or 8×4 or 4×8 or 4×4 size, respectively. In such a case, a (1:1) mapping of partitions is performed, where a larger sized partition is reduced to a smaller sized partition. Hence the MV associated with the source MB partition can be associated with the target MB partition [32].

- In the case where the source MB has an 8×8 partitioning size that is further sub-partitioned to sizes such as 8×4 or 4×8 or 4×4, the target partition would always have a 4×4 partitioning size. This cannot be considered as a (1:1) mapping of partitions, since we would have multiple partitions in the source MB that are mapped to a single partition in the target MB. These MVs need to be combined, such that they could be associated with the target 4×4 MB partition. In our implementation, we combine these MVs by simply computing their mean, which is consistent with our previous approach in combing MVs. In the special case when four 16×16 source MBs are mapped to a single 16×16 target MB, the four MVs of the source MBs are averaged to compute a MV for the target 16×16 MB.

- The (2:1) downsampling requires that all MVs are further down-scaled by a factor of two. The MVs are down-scaled by dividing both the horizontal and vertical components by two [13].

Recall that in the case of mixed MB coding types, where intra predicted or skipped MBs are present, they are treated as an inter coded MB with a 16×16 partitioning. A MV with zero horizontal and vertical components is associated with these 16×16 partitioning sizes. Other obvious cases, such as combining MBs that have equal MV components could also be considered. Such cases were not addressed in our implementation since the associated PSNR/bit-rate gain is expected to be negligible.
5.3 Simulation Setup and Results

In this section, we present simulation results of our proposed combined spatial-temporal transcoder of H.264/AVC to H.264/SVC.

5.3.1 Simulation Setup

In order to validate the performance of our proposed spatial-temporal transcoder with experimental results, it is necessary to compare it to a reference performance. In a similar manner to the temporal transcoder proposed in the last chapter, this reference performance can be obtained with the use of a cascaded decoder-encoder setup. An input stream would first be decoded, followed by spatial downsizing. Finally, both the original and the downsized resolutions are used as inputs to the encoder. The decoder-encoder setup serves as a high quality and computationally expensive method to convert an existing H.264/AVC stream to an H.264/SVC stream with combined spatial-temporal scalability. The main objective of our implementation is to obtain an output visual quality that is as close as possible to the decoder-encoder setup, while reducing the computation time as much as possible.

In order to better evaluate the proposed transcoding method, we show its effectiveness for encoding with and without inter-layer prediction (Mode 1 and Mode 0). When in Mode 0, the encoder encodes each spatial layer by employing inter prediction between frames in the same spatial layer (no inter-layer prediction between the base and enhancement layers is performed). On the other hand when in Mode 1, each spatial enhancement frame is encoded through employing inter-layer prediction with the base layer. Recall that when inter-layer prediction is employed, prediction information is upsampled by the SVC encoder from the base layer to be used to encode the enhancement layer.

The Decoder-Encoder-Mode 0 scheme represents a cascaded decoder-encoder setup without inter-layer prediction, while the Decoder-Encoder-Mode 1 scheme represents a cascaded decoder-encoder setup with inter-layer prediction. These two schemes perform a full ME fast search with a 32 pixel range.

The Transcoder-Mode 0 scheme represents our proposed transcoding setup without inter-layer prediction. The extracted high resolution information, along with the downsampled information are passed to the enhancement and base layers encoders, respectively. Each layer is encoded using this passed information. The Transcoder-Mode 1 scheme on
the other hand, represents our proposed transcoding setup with inter-layer prediction. The passed information is used to encode the base layer, from which coding and motion information is subsequently upsampled to encode each enhancement layer frame via inter-layer prediction. It is important to note that both transcoding schemes reuse the MB modes and newly computed MVs. A 3 pixel refinement is also performed to find a better estimation of the MVs.

The input test sequences consist of single reference AVC IPPP structured streams that are encoded at a frame rate of 30Hz with 81 frames and a CIF (352 by 288) resolution. Sequences with limited motion, such as Foreman and Coastguard, were tested along with sequences containing intensive motion, such as Football and NBA [29].

Our implementation uses the default settings of the reference software JVSM 9.16 [28]. Both the H.264/SVC hierarchical B-frame and zero-delay structures were tested, for a GOP size of 8. We use a QCIF resolution (176 by 144) for the spatial base layer and a CIF (352 by 288) for the spatial enhancement layer. The transcoding process was simulated with standard (equal weights) weighted prediction.

5.3.2 Simulation Results

The compression efficiency of the different schemes was tested by plotting the rate-distortion (RD) curves independently for each scheme. The RD curves consist of the average Y-PSNR (Y being the luminance component) against the coded bit-rate. The RD curves were plotted for the decoded SVC high resolution (enhancement) reconstructed sequences. The computational complexity was measured by calculating the computation time of each of the transcoding schemes as a percentage of the time taken by the Decoder-Encoder-Mode 0 scheme.

Compression Efficiency

The compression efficiency of our implemented schemes was tested for both hierarchical B-frame and zero-delay structures.

Hierarchical B-Frame Structures

The Foreman (low motion) and NBA (high motion) RD curves for a hierarchical B-frame structure with a GOP size of 8 are shown in Fig. 5.9 and Fig. 5.10, respectively. The
5 Combined Spatial-Temporal Transcoding of H.264/AVC to H.264/SVC

Decoder-Encoder-Mode 0 and Decoder-Encoder-Mode 1 represent the optimum transcoding performance for Mode 0 and Mode 1, respectively. From observing the RD curves of both the Foreman and NBA sequences, it is evident that Mode 1 provides a better compression efficiency than Mode 0 for the decoder-encoder setup. Hence the same trend is observed for the transcoding schemes, where Mode 1 provides a better compression efficiency than Mode 0 as well.

For the Foreman sequence, it can be observed in Fig. 5.9 that there is approximately a 0.6dB of difference between the Decoder-Encoder-Mode 0 and the Transcoder-Mode 0 schemes. A 0.4dB gap is approximately observed between the Decoder-Encoder-Mode 1 and Transcoder-Mode 1 schemes. For the NBA sequence in Fig. 5.10, there is approximately a 2dB difference between the Decoder-Encoder-Mode 0 and the Transcoder-Mode 0 schemes. A 1dB gap is observed between the Decoder-Encoder-Mode 1 and Transcoder-Mode 1 schemes.

It is evident that larger gaps are observed (for both Mode 0 and Mode 1) between the cascaded setup and the transcoder scheme for high motion sequence such as NBA with respect to low motion sequences such as Foreman. This is due to the fact that for high motion sequences there is a greater inaccuracy associated with computing a set of new MVs. It is important to note that the small fluctuations observed in the plots occur due to a limitation of the rate-control mechanism in the reference software that was used to produce these plots [28].

Hierarchical Zero-Delay Structures

The Foreman (low motion) and NBA (high motion) RD curves for a hierarchical zero-delay structure with a GOP size of 8 are shown in Fig. 5.11 and Fig. 5.12, respectively. The Decoder-Encoder-Mode 0 and Decoder-Encoder-Mode 1 represents the optimum transcoding performance for each mode. Again, from observing the RD curves of both sequences, it is evident that Mode 1 provides a better compression efficiency than Mode 0 for the decoder-encoder setup. Hence the same trend is observed for the transcoding scheme, where Mode 1 provides a better compression efficiency than Mode 0 as well.

For the Foreman sequence, it can be observed in Fig. 5.11 that there is (approximately) a 0.3dB difference between the Decoder-Encoder-Mode 0 and the Transcoder-Mode 0 schemes. A 0.1dB gap is observed between the Decoder-Encoder-Mode 1 and Transcoder-Mode 1
schemes for low bit-rates, while the Transcoder-Mode 1 scheme performs slightly better for high bit-rates, which again may be accounted for by some limitation in the rate-control mechanism used in the reference software [28]. The reference software controls the bit-rate by predicting the required quantization parameters. The predicted quantization parameters are then used to encode the video stream. The cascaded setup is only optimal in the sense that it theoretically provides the best conversion. Due to the added uncertainty associated with the rate-control, which is by no means a perfect mechanism, the performance of the cascaded setup might not necessarily result in the best performance.

It is also noticeable that the Transcoder-Mode 1 generally performs better than the Decoder-Encoder-Mode 0 scheme. For the NBA sequence in Fig. 5.12, there is (approximately) a 1dB difference between the Decoder-Encoder-Mode 0 and the Transcoder-Mode 0 schemes. A 0.2dB gap is observed between the Decoder-Encoder-Mode 1 and Transcoder-Mode 1 schemes for low bit-rates, and is negligible for high bit-rates.

A similar trend to the hierarchical B-frame case is observed, where the transcoding schemes have a better performance for low motion sequences over high motion sequences, although this difference is less noticeable. Again this accounted for by the fact that for high motion sequences there is a greater inaccuracy associated when computing a set of new MVs.

When comparing the hierarchical zero-delay transcoding schemes to their hierarchical B-frame counterparts (see Fig. 5.9-5.10), it is observed that they show a similar trend in terms of their performances for both low motion and high motion sequences. Both transcoding schemes, Transcoder-Mode 0 and Transcoder-Mode 1, are observed to perform much closer to the optimum decoder-encoder setup for zero-delay structures. The is accounted for by the fact that all of the list 0 MVs are computed, unlike in the hierarchical B-frame structure, where the list 1 MVs are set to zeros.

**Computational Complexity**

The computational complexity was tested for each of the different transcoding schemes with both the hierarchical B-frame and zero-delay structures. An average among the Foreman, Coastguard, Football, and NBA sequences was computed for a GOP of size 8. The average computational complexity for hierarchical B-frame and zero-delay structures is illustrated in Fig. 5.13 and Fig. 5.14, respectively.
Fig. 5.9  Plot of PSNR/Bitrate of Foreman for a hierarchical B-frames structure with a GOP size of 8.

Fig. 5.10  Plot of PSNR/Bitrate of NBA for a hierarchical B-frames structure with a GOP size of 8.
Fig. 5.11 Plot of PSNR/Bitrate of Foreman for a hierarchical zero-delay structure with a GOP size of 8.

Fig. 5.12 Plot of PSNR/Bitrate of NBA for a hierarchical zero-delay structure with a GOP size of 8.
The implemented transcoding scheme, Transcoder-Mode 0, reduces the overall computation time of the Decoder-Encoder-Mode 0 scheme by 70.2% and 61.1% for the B-frame and zero-delay structures, respectively. This due to the fact that the Transcoder-Mode 0 reduces the complexity of encoding both the spatial base and enhancement layers through MB mode reuse and the computing of new MVs.

With respect to the Decoder-Encoder-Mode 0 scheme, the Decoder-Encoder-Mode 1 scheme is observed to have an overall lower computation time by 46% and 26.9%, for the B-frame and zero-delay structures, respectively. The use of inter-layer prediction significantly reduces the time taken to encode the enhancement layer frames, and thus reduces the overall encoding time.

The Transcoder-Mode 1 scheme further reduces the computational complexity by reducing the encoding time of the base layer through MB mode reuse and the computing of new MVs. The use of the Transcoder-Mode 1 scheme results in reducing the computation time taken up by the Decoder-Encoder-Mode 1 scheme by a further 29.5% and 38.9%, for the B-frame and zero-delay structures, respectively. This results in a collective reduction, with respect to the Decoder-Encoder-Mode 0 scheme, of 75.5% and 65.8% for the B-frame and zero-delay structures, respectively.

The Transcoder-Mode 0 and Transcoder-Mode 1 schemes show a much better computational performance for the hierarchical B-frame over the hierarchical zero-delay structures, with an approximate difference of 10%. Similarly to the single spatial layer case discussed in the previous chapter, this difference can be accounted for by the fact that for the B-frame case, the encoder has to choose between using one list 0 MV, or one list 1 MV, or bi-prediction with both list 0 and list 1 MVs. On the other hand in the zero-delay case, the encoder only has the choice of using a list 0 MV. Thus in the B-frame case, a larger share of the encoding time is spent in forming a prediction with respect to the time spent on intra prediction or prediction other encoding overhead. Hence it is expected that the transcoding process for the B-frame case would be more computationally efficient than in the zero-delay case.

The computational complexity plots in Fig. 5.13 and Fig. 5.14 show the averages over different sequences. No variations in the complexity plots where observed for other GOP sizes, nor for longer or shorter sequences, since encoding is done on a frame by frame basis. We note that a similar trend to the single layer case in Chapter 4 was observed. Slightly larger deviations were observed for the Football sequence, and sequences with more motion
were observed to require a greater encoding time.

5.4 Summary

In this chapter, we presented a novel implementation of a combined spatial-temporal transcoder of H.264/AVC to H.264/SVC. A time profiling of the SVC encoder was performed to categorize the computational complexity of spatial-temporal encoding. A detailed discussion of our proposed implementation was presented, together with simulation results to validate the performance of our proposed scheme. It was shown that a significant decrease in computational complexity is achieved with a reduction in computation time of over 75% for B-frame structures, and 65% for the zero-delay structures with respect to the Transcoder-Mode 0 scheme. This high reduction in computation time was achieved while maintaining a small loss in compression efficiency. In the next chapter, we conclude this thesis and present some suggestions for future work.
Fig. 5.13 Average computation time for streams encoded with a hierarchical B-frames structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder-Mode 0 scheme.

Fig. 5.14 Average computation time for streams encoded with a hierarchical zero-delay structure. Each bar represent computation time as a percentage of the time taken by the Decoder-Encoder-Mode 0 scheme.
Chapter 6

Conclusion

6.1 Summary

This thesis presents a transcoding mechanism that adds scalability to an existing H.264/AVC stream by converting it to an H.264/SVC stream. This conversion is facilitated with the use of a closed-loop pixel-domain transcoder. Temporal scalability is provided to streams encoded in the H.264 format by mapping the input sequence to support a hierarchical frame referencing structure. The frame referencing structure of the input stream is mapped to a hierarchical referencing structure that is used by SVC. This mapping requires that a new set of motion vectors (MVs) is computed from the old input MVs, and that MB coding modes are reused. Spatial scalability was further provided by extending the temporal transcoder implementation to support spatial scalability as well. The implemented scheme was extended by performing a downsizing of the decoded full resolution video frames, such that they are used to encode lower resolution layers. In addition, the extracted and modified coding and motion information was further downsampled and reused to encode the lower spatial layers.

Our simulations have shown that the proposed transcoding scheme, even though simple, achieve a high PSNR that is close to the optimum cascaded decoder-encoder setup. They also have shown that a large reduction in computation time was provided in comparison to a cascaded decoder-encoder setup. These performance results show that we have successfully implemented a H.264/AVC to H.264/SVC temporal transcoder, and have further extended it to provide combined spatial-temporal scalability as well.
6.2 Future Work

To take this work a step further, there are several areas that could still be improved, such that our transcoder may provide better performance and flexibility. The compression efficiency could be improved for hierarchical B-frame structures by devising a method to effectively compute a set of new MVs for list 1. The use of a more optimized refinement mechanism to refine the newly computed MVs could also provide a possible increase in PSNR. MVs could also be combined using other methods to see the effect on the output PSNR. The transcoder could be extended to reuse intra MB mode information to encode the output intra predicted MBs, similarly to what has been shown by S. Moiron and M. Ghanbari [33]. This would reduce the computation time required by intra prediction. Although it might not have a significant effect on the overall performance, since it was shown that intra prediction constitutes only a maximum of 8% of the total encoding time.

The flexibility of our proposed mechanism could be increased by adhering it to adaptively support different referencing structures, rather than strictly IPPP. The temporal transcoder could also be made to adaptively support different types of SVC referencing structures. The proposed transcoder could be extended to support more than two spatial layers. The downsampling of coding and motion information could be modified to adaptively support any type of resolution conversion, since previous work for such a problem can be found in the literature [34],[35]. Finally, the prospect of adding quality scalability could also be explored.
References


