TRANSCODING H.265/HEVC VIDEO

SINA TAMANNA

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Sina Tamanna 860414-3811
E-mail: Sina.Tamanna@gmail.com

Supervisor:
Thomas Rusert, Ph.D.
Visual Technology, Ericsson, Stockholm

Academic Supervisor:
Hussein Aziz
School of Computing, BTH, Karlskrona

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet : www.bth.se/com
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
Video transcoding is the process of converting compressed video signals to adapt video characteristics such as video bit rate, video resolution, or video codec, so as to meet the specifications of communication channels and endpoint devices. A straightforward transcoding solution is to fully decode and encode the video. However, this method is computationally expensive and thus unsuitable in applications with tight resource constraints such as in software-based real-time environment. Therefore, efficient transcoding methods are required to reduce the transcoding complexity while preserving video quality.

Prior transcoding methods are suitable for video coding standards such as H.264/AVC and MPEG-2. H.265/HEVC has introduced new coding concepts, e.g., the quad-tree-based block structure, that are fundamentally different from those in prior standards. These concepts require existing transcoding methods to be adapted and novel solutions to be developed.

This work primarily addressed the issue of efficient HEVC transcoding for bit rate adaptation (reduction). The goal is to understand the transcoding behaviour for some straightforward transcoding strategies, and to subsequently optimize the complexity/quality trade-off by providing heuristics to reduce the number of coding options to evaluate.

A transcoder prototype is developed based on the HEVC reference software HM-8.2. The proposed transcoder reduces the transcoding time compared to full decoding and encoding by at least 80% while inducing a coding performance drop within a margin for 5%.

The thesis has been carried out in collaboration with Ericsson Research in Stockholm.

Keywords: Transcoding, Transrating, High Efficiency Video Coding, HEVC/H.265, Tree Sub-space Search
# Contents

Abstract i

1 Introduction 1
  1.1 Thesis Scope 2
  1.2 Research Questions 3
  1.3 Reference Software 3
  1.4 Thesis Organization 3

2 Video Coding and Transcoding 5
  2.1 Digital Video 5
  2.2 Video Coding Basics 6  
    2.2.1 Prediction Model 7  
    2.2.2 Spatial Model 7  
    2.2.3 Statistical Model 8
  2.3 Hybrid Video Coding Model 8
  2.4 Video Coding Standards Based on The Hybrid Model 9  
    2.4.1 High Efficiency Video Coding 10
  2.5 Video Quality Measurement 13  
    2.5.1 Peak Signal-to-Noise Ratio 14  
    2.5.2 Bit-rate 14  
    2.5.3 RDO Model of Reference HEVC Implementation 14
  2.6 Video Transcoding 15  
    2.6.1 Video Transcoding Challenges 15  
    2.6.2 Video Transcoding Architectures 15
  2.7 Related Work on Video Transcoding 18

3 Transcoding High Efficiency Video Coding 22  
  3.1 Full Prediction Reuse Transcoding Model 24  
  3.2 Intra Prediction Re-estimation Transcoding Model 24  
  3.3 MV Re-estimation Transcoding Model 25  
  3.4 Advanced Transcoding Model 26  
  3.5 Test Sequences and Encoder Configuration 27  
  3.6 Transcoder Evaluation Concepts 29  
    3.6.1 Bit rate ratio 30
4 Results
4.1 Base Bit-stream 36
4.2 Simple Cascaded Transcoding Model 37
4.3 Full Prediction Reuse Model 39
4.4 Intra Prediction Re-estimation Model 41
4.5 Motion Re-estimation Model 43
4.6 Advanced Transcoding Model 45
4.7 Class Average Performance 47
4.8 Time 51

5 Conclusions
5.1 Overview of Proposed Transcoding Models 54
5.2 Overview of Transcoders Performance 54
5.3 Future Work 55

A Algorithms of The Proposed Transcoder Models 56

B Spatial Resolution Reduction 61
Chapter 1

Introduction

Video content is produced daily through variety of electronic devices, however, storing and transmitting video signals in raw format is impractical due to its excessive resource requirement. Today popular video coding standards such as MPEG-4 and H.264 are used to compress the video signals before storing and transmitting. Accordingly, efficient video coding plays an important role in video communications. While video applications become wide-spread, there is a need for high compression and low complexity video coding algorithms that preserve image quality.

Standard organizations ISO, ITO, VCEG of ITU-T, and collaboration of many companies have developed video coding standards in the past to meet video coding requirements of the day. The Advanced Video Coding (AVC/H.264) standard is the most widely used video coding method. AVC is commonly known to be one of the major standards used in Blue Ray devices for video compression. It is also widely used by video streaming services, TV broadcasting, and video conferencing applications. Currently the most important development in this area is the introduction of H.265/HEVC standard which has been finalized in January 2013 (Ohm and Sullivan, 2013). The aim of standardization is to produce video compression specification that is capable of compression twice as effective as H.264/AVC standard in terms of coding complexity and quality.

There is a wide range of platforms that receive digital video. TVs, personal computers, mobile phones, and tablets each have different computational, display, and connectivity capabilities, thus video has to be converted to meet the specifications of target platform. This conversion is achieved through video transcoding. For transcoding, straightforward solution is to decode the compressed video signal and re-encode it to the target compression format, but this process is computationally complex. Particularly in real-time applications, there is a need to exploit the information that is already available through the compressed video bit-stream to speed-up the conversion (Xin et al., 2005a).

The objective of this thesis is to investigate efficient transcoding methods for HEVC. Using decode/re-encode as the performance reference, methods for advanced transcoding will be investigated.
1.1 Thesis Scope

A simple categorization of transcoding applications is provided in Figure 1.1.

![Transcoding methods categorization](image)

*Figure 1.1: Transcoding methods categorization*

As Figure 1.1 suggests, transcoding could be *homogeneous* or *heterogeneous*. Homogeneous transcoding is defined as converting bit-streams of equal format, e.g., AVC to AVC transcoding, whereas, heterogeneous transcoding requires change of bit-stream format, e.g., AVC to HEVC transcoding.

Transcoding could reduce the bit rate or spatial resolution. In this thesis, the focus is on transcoding methods to reduce bit rate for bit-streams encoded with HEVC standard.

Temporal resolution reduction transcoding was popular in the past (Shanableh and Ghanbari, 2000). However with the introduction AVC and HEVC standards, it has lost significance since it is rather simple to reduce the temporal resolution by removing parts of stream that results in valid sub-stream (Hellge et al., 2008). Therefore, temporal resolution transcoding will not be considered in this thesis.

In applications where reliable transmission channel is unavailable, error-resilience transcoding could be used; see Ahmad et al. (2005). This type of transcoding is out of scope of this thesis.

It is important to note that a similar goal as in video transcoding is pursuit in Scalable Video Coding (SVC), where a base-layer video is coded and enhancement layers are formed to provide additional levels of quality on demand (Schwarz et al., 2007). SVC is complex to implement and computationally expensive. Therefore this thesis is focused on video transcoding exclusively.
Chapter 1. Introduction

Aims and Objectives

Aim
- Investigate efficient transcoding methods for HEVC using decode/re-encode as a performance reference

Objective
- Implementing and evaluating transcoding methods from HEVC to HEVC

1.2 Research Questions

Research Question 1. What has been done in terms of state-of-the-art video transcoding?

Research Question 2. How does cascaded transcoding (decode and re-encode) performs for HEVC to HEVC transcoding in terms of coding complexity and quality?

Research Question 3. How does transcoding based on motion vector reuse performs for HEVC to HEVC transcoding?

Research Question 4. How does transcoding based on motion vector recalculation performs for HEVC to HEVC transcoding?

1.3 Reference Software

The implementation of the developed transcoding models were carried on top of the reference implementation of the HEVC standard, which was iterated as far as version 8.2 as the time of writing this thesis\(^1\). The software is written in C++ and includes a reference implementation for decoder and the encoder. It supports variety of configurations through the use of configuration files and passing arguments to the executables.

It must be noted that the implementation is not optimized for speed but it is meant to demonstrate the performance of the complaint compressed video stream created using the HEVC standard.

1.4 Thesis Organization

Chapter 2 presents a general description of the underlying topics of video coding and transcoding, and a summary of previous works on transcoding.

Chapter 3 presents a description of the developed transcoding algorithms. Which

\(^1\)http://hevc.kw.bbc.co.uk/svn/jctvc-hm/tags/HM-8.2
includes four transcoding models: 1) Full Prediction Re-use; 2) Intra Prediction Re-estimation; 3) MV Re-estimation; and 4) Advanced Transcoding Model.

Chapter 4 presents the design of transcoding simulations and performance measurement methods in addition to the simulation results and discussions on transcoders’ performance.

Chapter 5 presents the overview of the thesis and its contributions along with possible tracks for future works.
Chapter 2

Video Coding and Transcoding

A comprehensive account of digital video coding and related techniques would exhausted the limits of this thesis, hence, the intention is to provide a short introduction to relevant topics of digital videos, and video coding in particular. See Richardson (2011) for a comprehensive coverage of video coding concepts.

2.1 Digital Video

Digital video is a discrete representation of real world images sampled in spatial and temporal domain. In temporal domain samples are commonly taken at the rate of 25, 30, or more, frames per second. Each video frame is a still image composed of pixels bounded by spatial dimensions. Typical video spatial-resolutions are $1280 \times 720$ (HD) or $1920 \times 1080$ (Full HD) pixels.

A pixel has one or more components according to a color space. Commonly used color spaces are RGB and YCrCb. RGB color space describes the relative proportions of Red, Blue, and Green in a pixel. RGB components are commonly measured in the range of 0-255, that is 8-bits for each component and 24-bits in total. The YCrCb color space is developed with the human visual system in mind. Human visual perception is less sensitive to colors compared to brightness, hence by exploiting this fact, reduction in number of bits required to store images could be achieved by reducing the chroma resolution. In YCrCb color space, Y is the luminance and it is calculated as the weighted average ($k_r, k_g, k_b$) of RGB:

$$Y = k_r R + k_g G + k_b B$$

(2.1)

The color information is calculated as the difference between Y and RGB:

$$Cr = R - Y$$
$$Cg = G - Y$$
$$Cb = B - Y$$

(2.2)

Observe that since $Cr + Cg + Cb$ is constant, storing $Cr$ and $Cb$ is sufficient. As mentioned before, YCrCb frames could have pixels sampled with different
resolution for luma and chroma. These differences are noted in the sampling format as 4:4:4, 4:2:2, and 4:2:0. In the 4:4:4 format, each pixel is sampled with equal resolution. In the 4:2:2 format, chroma is at the half rate of luma. And in 4:2:0 format, chroma is recorded at the quarter rate of luma.

There are many choices for sampling a video at different spatial and temporal resolution. Standards are defined to support common requirements of video formats. A base format called Common Intermediate Format, CIF, is listed in Table 2.1 with high resolution derivatives; see BT.601-6 (2007).

<table>
<thead>
<tr>
<th>Format</th>
<th>Luminance Resolution</th>
<th>Pixels per Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIF</td>
<td>352 × 288</td>
<td>101,376</td>
</tr>
<tr>
<td>4CIF</td>
<td>704 × 576</td>
<td>405,504</td>
</tr>
<tr>
<td>720p</td>
<td>1280 × 720</td>
<td>921,600</td>
</tr>
<tr>
<td>1080p</td>
<td>1920 × 1080</td>
<td>2,073,600</td>
</tr>
<tr>
<td>2540p</td>
<td>4520 × 2540</td>
<td>11,480,800</td>
</tr>
</tbody>
</table>

Table 2.1: Video resolution and bit rate for standard formats

2.2 Video Coding Basics

According to the Table 2.1, number of required pixels per frame is huge, therefore storing and transmitting raw digital video requires excessive amount of space and bandwidth. To reduce video bandwidth requirements compression methods are used. In general, compression is defined as encoding data to reduce the number of bits required to present the data. Compression could be lossless or lossy. A lossless compression preserves the quality so that after decompression the original data is obtained, whereas, in lossy compression, while offering higher compression ratio, the decompressed data is unequal to the original data. Video signals are compressed and decompressed with the techniques discussed under the term video coding, with compressor often denoted as enCOder and decompressor as DECoder, which collectively form the term CODEC. Therefore a CODEC is the collection of methods used to compress and decompress digital videos. The general process of encoding and decoding of video signal in transmission chain is given in Figure 2.1.

Figure 2.1: Video coding process
sion efficiency by exploiting temporal, spatial, and statistical redundancies. A common encoder model is illustrated in Figure 2.2.

According to Figure 2.2 there are three models: 1) Prediction Model; 2) Spatial Model; and 3) Statistical Model. These models are explained further in Section 2.2.1, Section 2.2.2, and Section 2.2.3.

2.2.1 Prediction Model
This model exploits the temporal (inter-prediction) and spatial (intra-prediction) redundancies. Availability of inter-prediction through temporal redundancy is due to the motion, uncovered regions, and luminance changes in the pictures. Usually inter-prediction is carried out in two steps: 1) Motion Estimation (ME): finding the best match between regions of reference and past or future frames; and 2) Motion Compensation (MC): finding the difference between matching regions. To increase the efficiency of ME and MC, the picture is divided into regions called blocks. A cluster of neighbouring blocks is called a macroblock, and its size could vary. The output of Prediction Model is residuals and motion vectors. Residual is the difference between a matched region and the reference region. Motion vector is a vector that indicates the direction in which block is moving.

2.2.2 Spatial Model
Usually this model is responsible for transformation and quantization. Transformation is applied to reduce the dependency between the sample points, and quantization reduces the precision at which samples are represented. A commonly used transformation in video coding is the Discrete Cosine Transform (DCT) that operates on a matrix of values which are typically residuals from prediction model (Ahmed et al., 1974). The output from DCT is coefficients that are farther quantized to reduce the number of bits required for coding it. Quantization could be scalar or vector based. Scalar quantization maps the range of values to scalers, while vector quantization maps a group of values, such as image samples, into a codeword.
Because the output matrix from quantization is composed of many zeros, it is beneficial to group the zero entities. Due to the nature of DCT coefficients, a zigzag scan of the matrix of quantized coefficients will reorder the coefficients to a string of numbers with the most of non-zero values in the beginning. This string could be stored with fewer bits by using Run-Length Encoding (RLE), that is storing consecutive occurrences of a digit as a single value together with digit’s count.

2.2.3 Statistical Model

The outputs from Prediction and Spatial Models are combination of symbols and numbers. Although significant amount of redundancy is removed by exploiting temporal and spatial redundancy, there are still statistical redundancies that could be exploited: It is highly probable that a previously coded motion vector will be similar to the current one, hence Statistical Models usually use predictive coding to reuse these vectors and code the residual instead of the original motion vector. Finally, Variable Length Coding could also be used to assign smaller codewords to frequent symbols and numbers to maximize the coding efficiency. The output of Statistical Model is a compressed bit-stream suitable for storage or transmission.

2.3 Hybrid Video Coding Model

Several video coding standards has been developed to meet the requirements of various platforms such as real-time applications, wireless networks, and home media. The common factor between these standards is that they incorporate motion estimation and compensation, transforms and quantization, and entropy encoding. This general design model is commonly referred to as the hybrid model. A block-diagram of the hybrid model is given in Figure 2.3.
Figure 2.3: Encoding based on the hybrid video coding model

The input frame to the hybrid model in Figure 2.3 follows two different paths. A path follows the encoding route and the other follows the reconstruction route.

The encoding path: In the prediction model, each block is compared with regions of the same (intra-prediction) or past and future pictures/frames (inter-prediction) to find similar areas. Intra picture estimation finds the similar regions in the current image and intra picture prediction produces the prediction using the matching region. Similarly, motion compensation finds the matching blocks in past or future pictures. When a match is found, the offset between two blocks is saved. Using the offset a motion compensated prediction is created and the residual is formed by subtracting the prediction from the original block. The direction of motion and direction of intra prediction combined with the residuals are passed to the spatial model. The residual is transformed in spatial model and each transformation is quantized. The quantized coefficients of the transformation is re-ordered and coded.

The reconstruction path: Reconstruction starts by inverse quantization and continues by inverse transformation to reverse the operations of spatial model and produces residuals. Notice that these residuals will be unequal to the original residuals. The residuals are added to the motion compensated prediction to obtain a frame. This frame is passed through loop filter to improve the frame quality. The results is stored for future predictions.

2.4 Video Coding Standards Based on The Hybrid Model

First video coding standard that pioneered the hybrid model was developed in 1990 by International Telecommunication Union (ITU) and it was called H.261. Main features of H.261 were 4:2:0 sampling, 16×16 pixel macroblocks, 8×8 pixel
DCT, scalar quantization, zigzag scanning, and Huffman-coding. In 1992, Moving Picture Experts Group-1, MPEG-1, standard was designed by ISO/IEC to compress the conventional VHS recordings down to 1.5 MBits/sec while keeping the visual quality to an acceptable level (ISO/IEC-11172, 1993). The down side of the standard was that it was restricted to progressive compression, whereas, PAL and NTSC standards were using interlaced frames so there were compatibility issues. The follow-up was the development of MPEG-2/H.262 by the joint effort of ISO and ITU (ISO/IEC-13818, 1993). MPEG-2 supported both Standard Definition (SD), $720 \times 576$ pixels, and High Definition (HD), $1920 \times 1080$ pixels, videos.

Consecutively, ITU released the H.263 standard in 1995 with the goal of high quality and low bit rate video coding targeted at video conferencing and wireless network applications (ITU-T/SG15, 1995). At the same time, MPEG-4 standardization was started in 1995 and it has been extended ever since. MPEG-4 has encompassed the features of MPEG-1 and MPEG-2 with support for 3D rendering, global motion compensation, and more (ISO/IEC-JTC1/SC29/WG11-N4668, 2002).

In 2003, H.264, or commonly called Advanced Video Coding (AVC), video coding standard has been finalized by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC MPEG. AVC has been developed with the goal of enhancing compression performance while meeting the requirements of wide range of applications over online and offline media (Wiegand et al., 2003). The successor to H.264/AVC video coding standard is H.265/HEVC and it has been developed by VCEG and ISO/IEC MPEG and finalized in 2013 (Ohm and Sullivan, 2013).

Similar to previous video coding standards, AVC and HEVC are based on the hybrid model. AVC has improved compression efficiency up to 50% compared to previous standards (Ostermann et al., 2004). In succession, among the many capabilities added to HEVC, see (Wiegand et al., 2011) for details, perhaps the most important one is the improved compression efficiency. As it is reported it can achieve same subjective quality using half the bit rate required by AVC (Li et al., 2012).

### 2.4.1 High Efficiency Video Coding

An overview of the HEVC standard relevant to this work is provided in this section. In depth explanation of each concept can be found in the published works by Sole et al. (2012); Lainema et al. (2012); Sjoberg et al. (2012). For the complete set of tools and techniques incorporated in HEVC standard see the latest specification by Bross et al. (2012) and overview paper by Ohm and Sullivan (2013). HEVC is based on the hybrid model explained in Section 2.4. Concepts relevant to this thesis are explained as follows.
Coding Tree Units

The core of coding standards prior to HEVC was based on a unit called macroblock. A macroblock is a group of 16×16 pixels which provides the basics to do structured coding of a larger frame. This concept is translated into Coding Tree Unit (CTU) with HEVC standard, this structure is flexible compared to macroblock. A CTU could be of size 64×64, 32×32, or 16×16 pixels.

Coding Units

Each CTU is organized in a quad-tree form for further partitioning to smaller sizes called Coding Unit (CU). An example of partitioning a CTU into CUs is given in Figure 2.4.

![Figure 2.4: HEVC Coding Tree Unit](image)

The tree is traversed in depth-first order and the corresponding nodes of the tree is visible on the CTU structure in Figure 2.4.

Prediction Modes

Each CU could be predicted using three prediction modes: 1) Intra-predicted CU; 2) Inter-predicted CU; 3) and Skipped CU. Intra-prediction uses pixel information available in the current picture as prediction reference, and a prediction direction is extracted. Inter-prediction uses pixel information available in the past or future frames as prediction reference, and for that purpose motion vectors are extracted as the offset between the matching CUs. A skipped CU is similar to an inter-predicted CU, however there is no motion information, hence skipped CUs reuse motion information already available from previous or future frames.
In contrast to eight possible directional prediction of intra blocks in AVC, HEVC supports 34 intra prediction modes with 33 distinct directions, and knowing that intra prediction block sizes could range from $4 \times 4$ to $32 \times 32$, there are 132 combinations of block size and prediction direction defined for HEVC bit-streams.

**Prediction Units**

A leaf CU in the CTU can be farther split into regions of homogeneous prediction called Prediction Units (PU). A CU can be split into one, two, or four PUs. The possible PU modes depend on the prediction mode. For intra-prediction there is can be two possible modes, whereas inter-prediction can be done using one of eight possible modes. Figure 2.5 presents all possible PU modes available in HEVC where N determines the number of pixels in the block.

![Prediction Unit partitioning modes](image)

**Figure 2.5:** Prediction Unit partitioning modes

**Transform Units**

A Transform Unit (TU) is defined for each CU in a similar manner to PU and it is organized in a quad-tree form. Each TU is responsible for transformation of the residuals from PU. For each TU, residual coefficients are calculated by applying one integer transform.
Motion Vectors

A Motion Vector (MV) is used for inter prediction and it determines the offset by which a reference block to the current block is located in the past or future picture. MVs are determined per PU and each PU references one or two blocks.

Picture and Slice

A picture is the image captured in a time \( t \) and it can be partitioned into one or more slices. Slices are encoded separately from other parts of the picture. Slice and frame has three types: I, P, and B. In I-slices only intra prediction is allowed. In P-slices intra prediction and inter prediction can be used. But only one MV per PU is allowed for inter-prediction. B-slices allow inter and intra prediction with one or two MVs per PU. The picture types are illustrated in Figure 2.6.

![Figure 2.6: Example HEVC picture types and the connection between](image)

Group of Pictures

A Group of Pictures (GOP) is a series of pictures ordered independently from the input pictures order to enable capabilities such as random-access to video pictures. A GOP starts with an I-picture and follows by P- and B-pictures. I- and P-picture are used for referencing while B-pictures only reference I- and P-pictures.

2.5 Video Quality Measurement

It is important to compress the video signal as much as possible and keep the video quality close to the original. There are, generally, two categories of methods for measuring video quality: 1) subjective quality is based on the test procedures devised by ITU, as detailed in BT.500-11 (2002), to quantify the quality using human agents; and 2) objective quality is measured using mathematical models
to approximate the subjective quality. Subjective quality assessment of digital videos requires human agents, which is expensive, hence objective quality is a suitable alternative.

The main mathematical model used by the researchers is Rate-Distortion Optimization (RDO) model (Sullivan and Wiegand, 1998). Based on this model, distortion is optimized based on changes in rate which is the amount of data required for encoding the input data. In video coding, every decision usually affects the Rate-Distortion (RD) values, and the challenge is to find the optimal solution. Commonly used RD criteria in video coding is PSNR–bit rate pair (Sullivan and Wiegand, 1998).

### 2.5.1 Peak Signal-to-Noise Ratio

The most common objective quality measure is Peak Noise-to-Source Ratio (PSNR). It is measured in decibel ($dB$) as follows:

$$PSNR(Img_1, Img_2) = 10 \log_{10} \frac{(2^n - 1)^2}{MSE(Img_1, Img_2)}$$  \hspace{1cm} (2.3)

As shown in Equation 2.3, PSNR is measured based on the Mean Square Error (MSE) between two images, of which, one image is the original image and another is the compressed image. $n$ is the number of bits used to represent each pixel, which is typically 8 bits. High PSNR values indicate that the input and output images are similar. For typical cases, PSNR values range from 30 dB to 50 dB (Barni, 2006). Note that only the PSNR for luma channel is measured (Y-PSNR)

### 2.5.2 Bit-rate

The bit rate ($R$) of a bit-stream is calculated by averaging the total number of bits in the bit-stream by the length of the bit-stream measured in seconds. The results is usually measured with kilobits per-second ($kbit/s$) or megabits per-second ($Mbit/s$). The common method to control bit rate by the encoder is to adjust the Quantization Parameter (QP). QP determines how rough the quantizer will approximate the coefficients. Higher QP will reduce the bit rate and conversely lower QP will increase the bit rate.

### 2.5.3 RDO Model of Reference HEVC Implementation

The video coding standards define the specification of bit-stream that is compatible with the standard. The methods by which such bit-stream is obtained is out of the scoop of the standard. This is true for HEVC standard as well. HM-8.2 reference model implementation of HEVC standard uses RDO method detailed by Li et al. (2011) and Tan and Bossen (2011).
Higher PSNR usually requires higher bit rate and the challenge is to develop an encoder model that could keep the bit rate as low as possible while achieving high PSNR.

2.6 Video Transcoding

Video transcoding could change the video format or change video characteristics such as resolution or bit rate. The focus of this thesis is on transcoding for bit rate reduction.

Before discussing transcoding methods, most common challenges of designing transcoding algorithms are discussed in Section 2.6.1. Later on, in Section 2.6.2 various architectures for resolving transcoding challenges are summarized. A summary of prior work on transcoding is provided in Section 2.7.

2.6.1 Video Transcoding Challenges

Transcoding could have two main issues: transcoding could increase distortion and complexity (Chen and Ngan, 2007). Distortion, in the context of video coding, is the picture quality at a given bit rate. Complexity refers to processing time and memory requirements.

Distortion

Transcoding is based on video sequence that is created by decoding and input bit-stream. Encoding of images that has degraded quality will farther reduce its quality.

Computational Complexity

When constrained by resources, time complexity introduces delay, and memory complexity limits the maximum quality. In some applications, such as real-time video broadcasting, time complexity of transcoding is important and it is required to be minimized.

2.6.2 Video Transcoding Architectures

A basic solution to transcoding is the cascaded decoder-encoder transcoding, also referred to as pixel-domain transcoding, that is fully decoding the input bit-stream and re-encoding it with new parameters based on the target specifications, for illustration see Figure 2.7. Note that complete decode and re-encoding is demanding both in memory consumption and complexity.
According to Ahmad et al. (2005) video transcoding could be \textit{open-loop} or \textit{closed-loop}. In open-loop architecture, transcoding is without feed-back. A video picture is transcoded without buffering and the next picture is transcoded independently from previous pictures. Figure 2.8 (a) illustrates open-loop transcoding architecture. In contrasted, closed-loop transcoding uses a buffer to store pictures. Figure 2.8 (b) illustrates closed-loop transcoding architecture.

Categorizing farther, transcoding could be carried out in: (a) \textit{spatial-domain}; (b) \textit{frequency-domain}; or (c) \textit{hybrid-domain}. These categories are explained in the following Sections.
Spatial-domain Transcoding

This method provides a way to reuse motion vectors so that the most time consuming operation is skipped. Motion vector calculations account for up to 70% of the calculations of encoders (Shanableh and Ghanbari, 2000). Transcoding is carried out in spatial domain by decoding the bit-stream, inverse quantization, inverse transforms, motion compensation and then re-applying them. This is illustrated in Figure 2.9. Two additional units are needed to transcode the input video with lower spatial resolution: Spatial Resolution Reduction (SRR) and Motion Vector Composition and Refinement (MVCR). If SRR modifies the resolution, MVCR adjusts the motion vectors to accommodate the changes. Transcoding in spatial-domain is drift free, but since every step of cascaded transcoding is carried out, except MV calculations, it is computationally expensive.

Frequency-domain Transcoding

As shown in Figure 2.10, the input bit-stream is decoded and inverse quantized then re-quantization is done with different parameters in frequency-domain. Motion data is reused to reduce the computational complexity. Frequency-domain Spatial Resolution Reduction (FSRR) can be used in conjunction with MVCR to achieve spatial resolution reduction. In this architecture the transcoding is done through quantization of higher frequencies and keeping the old motion data. This could severely affect the video quality because prediction data is not in cor-
respondents with motion data. Compared to Spatial-domain transcoding, this architecture requires less memory since the buffer at the decoder side is avoided.

![Frequency-domain transcoding architecture](image)

**Figure 2.10:** Frequency-domain transcoding architecture

**Hybrid-domain Transcoding**

It is a combination of spatial- and frequency-domain architectures. Spatial-domain architecture is drift-free but it is also computationally expensive. In contrast, frequency-domain architecture requires less computation but it suffers from drift. Hybrid-domain transcoding tries to exploit the benefits of each transcoding method. For example, I-pictures of a video bit-stream could be transcoded in frequency-domain since they do not require motion data. Accordingly, P- and B-pictures would be transcoded in spatial-domain to maintain accurate prediction.

### 2.7 Related Work on Video Transcoding

Early research efforts in video transcoding dates back to 1966 at the age of color televisions where the goal was to adapt analogue video signals based on color system of a country, such as transcoding from PAL to NTSC (Watson, 1966). A challenge with analogue video transcoding was separation of chroma and luma signals. Progress in digital technologies helped achieve higher accuracy transcoding and removing the need for chroma and luma segregation (Kinuhata et al., 1978); however, new problems have arose. Particularly, accumulation of quantization error has been noted in late 80’s (Nishitani, 1986). Quantization were
usually carried out to reduce bit rate to accommodate to the needs of a channel with limited capacity. Quantization introduces error which would degrade the visual quality. A solution was discussed by Ayanoglu and Gitlin (1992) based on predictive vector quantization that was shown to improve the quality of transcoding.

Variable channel capacity inspired researchers to focus on transcoding that reduces bit rate, which is referred to as transrating. Introduction of hybrid video coding model helped the researchers to provide systematic solutions to transcoding. Considering the dependencies between the stages in hybrid model, a challenging task is to manage the delay due to the need to reverse some of the stages of hybrid model to retrieve the signal in the form that is suitable for transcoding.

Delay in transcoding was first studied by Morrison (1993). A notable strategy, based on cascaded transcoding architecture, was proposed by Morrison et al. (1994) to reduce the bit rate while maintaining a low delay. The idea was based on the re-quantization process in frequency domain (Ayanoglu and Gitlin, 1992); see Figure 2.11 for illustration.

As it is shown in Figure 2.11, the Motion Vectors (MVs) are reused without modification and transcoding is only applied in the frequency domain, but, quantization is lossy, hence the prediction using the old MV will introduce drift. The drift could be reduced by addition of a buffer and calculating the error from quantization and compensating for it, however this will increase the delay (Morrison et al., 1994). To minimize the delay, Chang and Messerschmitt (1995) proposed a method for reduction of calculations required for MVs. The approach is described as applying MV recalculations only in small areas of the image instead of the whole image. The results shows improvement in the transcoding time, however, error in areas where MV recalculations is not applied is propagated.
Keesman et al. (1996) applied the frequency-domain transcoding of Morrison et al. (1994) to video coded with MPEG-1 standard. After availability of MPEG-2 standard, Swann and Kingsbury (1996) investigated error-resilient transcoding. Assuncao and Ghanbari (1997) proposed a frequency-domain transcoder for MPEG-2 bit-streams with a new quantization method. They showed that it is possible to achieve better quality video from transcoding compared to the quality of video directly encoded using quantization of MPEG-2 standard at the same bit rate. This idea was discussed farther by Assuncao and Ghanbari (1998).

Transcoding has also been studied as an optimization problem under transcoding distortion versus bit rate reduction (Assuncao and Ghanbari, 1997). Tudor and Werner (1997) highlighted the real-time transcoding of MPEG-2 bit-streams by using bit rate statistics to adjust the transcoding complexity.

Next-generation standards based on hybrid-model were defined (H.263) and the research changed focus on reducing the transcoding complexity based on the most important aspects of the technology, namely MV reuse. Bjork and Christopoulos (1998) discussed bit rate and spatial resolution reduction for H.263 bit-streams with the reuse of MV data. Bjork and Christopoulos (1998) investigated down-sampling methods for MV mapping for H.263 bit-streams and in line with this work Shen et al. (1999) reused MVs and weighted them adaptively to achieve better quality. Youn et al. (1999) focused on the MV refinement for transcoding, and Werner (1999) provided analytical results for transcoding and proposed transcoding based on new re-quantization methods to reduce distortion.

Availability of bit-streams coded with MPEG-2 standards motivated researchers to do heterogeneous transcoding from MPEG-2 to H.263 (Shanableh and Ghanbari, 2000). New spatial resolution reduction method with MV reuse proposed by Yin et al. (2000). Following the goal of bit rate reduction transcoding, Lin and Lee (2001) improved the computational complexity of such transcoders. Transcoding in frequency-domain usually involves many matrix multiplications which is computationally heavy. This complexity was reduced by using decomposition methods proposed by Merhav (1999) which could be applied to bit rate and spatial reduction.

Yin et al. (2002) investigated the drift problem for spatial resolution reduction transcoding. In this work, two MB mode selection method was described: Intra-Inter and Inter-Intra. Former maps every mode to inter mode, later maps every mode to intra mode. Both methods require the picture to be fully re-constructed, which could increase the transcoding complexity.

Proposal of H.264 standard, promoted heterogeneous transcoding from MPEG-2 to H.264. For this matter, exhaustive search for MB mode selection was investigated by Chen et al. (2004). Xin et al. (2005b) derives a 2D transform to convert DCT-coefficients from MPEG-2 bit-stream to H.264 compatible bit-stream. A theoretical analysis of such transforms is provided by Chen et al. (2005). Block size estimation based on DCT-coefficients for MPEG-2 to AVC transcoding is investigated by Tang et al. (2008).
Recent works mostly have focused on transcoding AVC/H.264 bit-streams (Su et al., 2005; Xin et al., 2005b; Zhou et al., 2005; Lefol et al., 2006). Fernández et al. (2006) exploited the correlation between MPEG-2 residuals and MB mode of H.264 to develop a decision tree based method to train a classifier and categorize MB modes efficiently. Same idea was applied to H.263 to H.264 transcoding by Fernández-Escribano et al. (2007). Shen (2007) analysis the effects of re-quantization step-size of bit rate reduction transcoding and shows that it has different effect on rate-distortion compared to direct coding. The results provide insight on how to choice appropriate quantization parameter.

To provide farther insight into the transcoding performance Goldmann et al. (2010) investigated the effects of transcoding on subjective quality of the video with specific attention for the effects of drift and transcoding artifacts. To reduce drift, De Cock et al. (2010) improved drift-compensation in transcoding H.264 bit-streams, the idea has farther analysed by Cheng et al. (2011).

Lately, on the topic of AVC transcoding a new method for MB mode selection for AVC transcoding was provided by Wu and Lin (2009). Kwon (2012) showed that using smaller quantization parameter for regions of the picture that has non-zero MV has a positive effect on the transcoding quality.

HEVC/H.265 standard is new and there are few works on transcoding HEVC bit-streams. Zhang et al. (2012) investigated AVC to HEVC transcoding with the goal of exploiting the information that are shared between two standards. For example a part of solution was concentrated on merging smaller blocks of AVC to produce bigger blocks for HEVC and looking at possible estimations of new MV. Another challenge for AVC to HEVC transcoding is the intra mode selection. By using machine learning methods, Zhang et al. (2013) investigated the efficiency of predicting the suitable modes for AVC to HEVC transcoding.
Chapter 3

Transcoding High Efficiency Video Coding

A simple drift free transcoding could be achieved by cascading a decoder and encoder, where at the encoder side the video is encoded with regards to target platform specifications. This solution is computationally expensive, however the video quality is preserved (Lefol et al., 2006; Bross et al., 2012). The preservation of video quality is an important characteristic, since it provides a benchmark for more advanced transcoding methods. This solution is key-worded Simple Cascaded Transcoding (SCT), so to differentiate it from advanced cascaded transcoding methods proposed in this work.

Figure 3.1: The SCT transcoder model

Down-sampler is used for spatial-resolution reduction. If the SCT model is used for bit-rate reduction then down-sampling factor is 1 therefore the down-sampled Pixel Data is equal to the original Pixel Data. Bit-rate reduction is achieved through the used of higher Quantization Parameter (QP) at the HEVC Encoder.

The proposed transcoding models for bit-rate reduction are designed with two goals: 1) understanding the extend by which the transcoding time is reduced by exploiting the information available from input bit-stream; 2) reducing the
bit-rate and producing video quality as close as to the video quality of SCT model while minimizing the transcoding time. To preserve the video quality a closed-loop architecture is used. Closed-loop architectures decode the motion information therefore they are drift free (Cheng et al., 2011).

Four transcoding models are proposed for the bit-rate reduction. These models are based on the idea that the input bit-stream contains valuable information for fast re-encoding. Figure 3.2 illustrates the general model that the proposed transcoder models follow. This model is based on the cascaded model with the addition of Search Space Determiner (SSD). SSD restricts the search space of Transcoder Encoder by re-using the Frame Data created by Transcode Decoder through decoding of HEVC Bit-stream1.

The difference between the proposed models is the way by which the search space is determined by the Search Space Determiner. Proposed transcoder models are key-worded: Full Prediction Reuse (FPR); Intra Prediction Re-estimation (IPR); MV Re-estimation (MR); and Advanced Transcoding (AT).

FPR is spatial-domain transcoding which reuses all the information available in the spatial-domain. IPR is similar to FPR, with one major difference. Intra-prediction is carried out fully for intra frames, because in applications with random access requirement there are I-frames at the beginning of each GOP, and it seems that these I-frames will have a great impact on the quality of following B- and P-frames. To measure this impact the IPR transcoding model is developed.

The MR transcoding model is similar to the IPR model with the addition of MV re-estimation. This change is made with the goal of understanding how much the video quality could be improved if the transcoder was free to search for new MVs at the CU level. The AT model is designed to get as close as possible to cascaded transcoding quality and bit-rate with minimum transcoding time.
3.1 Full Prediction Reuse Transcoding Model

The idea is to recursively traverse the Coding Tree Unit (CTU) and if a leaf node is reached, which is determined by examining the depth of decoded CU from input bit-stream \((CU_I)\), the CU is encoded by re-using input \(CU_I\) structure. Thus, the input CTU structure is replicated in the transcoded CTU. A pseudo code describing the FPR transcoding model is provided in Appendix A, Algorithm 2.

![Figure 3.3: Example CTU structure and Frame Data that is re-used by the FPR transcoding model](image)

Figure 3.3 illustrates an input CTU and the Frame Data that is re-used for transcoding. In this Figure, depth of CU is denoted by \(D_0, D_1, \text{etc.}\) Prediction modes are shown by INTRA, INTER, and SKIP flags. PU splits are shown by \(2N \times 2N\) and so on. In this example, transcoder starts at the root node with the largest CU size, \(64 \times 64\). By copying the CU splits recursively, the tree structure of the input CTU is replicated in the output. At the leaf nodes the following CU information (Frame Data) is used to encode the output CU: 1) CU prediction mode; 2) PU split mode; 3) MVs for inter predicted CU; and 4) intra directions for intra predicted CU.

3.2 Intra Prediction Re-estimation Transcoding Model

The model structure is same as FPR, however for intra coded CUs intra directions and modes are re-estimated in the same manner as the reference encoder,
which is an exhaustive search by evaluating each direction using Rate-Distortion Optimization (model) at the maximum CU depth. The input CTU structure is replicated in the transcoded CTU. A pseudo code describing the IPR transcoding model is provided in Appendix A, Algorithm 3.

### 3.3 MV Re-estimation Transcoding Model

In succession to the IPR model, MR model expands the search space for optimal CU encoding by addition of Motion Vector re-estimation at each node of the tree. The re-estimation is exhaustive, meaning, CUs of past and future frames are searched thoroughly for the closest match. The difference with the SCT model is that MV re-estimation is only carried out at the leafs of the tree structure and the number of MVs are determined by the corresponding input PU splits. Therefore the search space is very small compared to the SCT model where every combination of PU split and MVs are examined in every node. The MR model is described by the pseudo code in Appendix A, Algorithm 4.

![CTU structure and Frame Data](image)

**Figure 3.4:** Example CTU structure and Frame Data that is re-used by the MR transcoding model

Figure 3.4 shows a CTU structure and underlying Frame Data that is re-used. Compared to the FPR model example in Figure 3.3, the MVs of inter predicted CUs are omitted, therefore, an exhaustive search will be carried out to re-estimate the optimal MVs. Notice that the tree structure of the input CTU is replicated in the output CTU.
3.4 Advanced Transcoding Model

The AT model is built on the MR model with one major difference. While traversing the nodes of input tree structure specific combination of CU modes and PU splits are examined before reaching the leaf node, therefore, the transcoded tree structure could be shallower compared to the input but never deeper. In a sense, to achieve higher transcoding performance the search space of possible CTU coding has been extended.

Three observations are important for understanding this model: 1) Skipped blocks require the least bits to encode; 2) Merging blocks reduces the number of bits; 3) $2N \times 2N$ splitting requires one motion vector to be signalled, hence it is very possible that it will require less bits to encode the block. The heuristics build upon these observations are: 1) Try skip and merge combinations on the root node of the tree and the node before the leaf node; 2) Try Inter- and Intra-coding with the size of $2N \times 2N$ on each node. The Algorithm 5 in Appendix A illustrates this transcoding model. Note that the input CU modes and PU modes are still tested at the tree leaf.

The AT model will traverse the input tree structure and before reaching the leaf node on each recursive iteration the current CU will be encoded with: 1) $2N \times 2N$ intra prediction by re-calculating intra prediction direction; 2) Inter prediction with $2N \times 2N$ PU split and re-calculating the motion vector; 3) Inter prediction by re-using PU split mode of the corresponding leaf node and re-calculating motion vector; and 4) Skip mode. When the leaf node is reached, the corresponding CU will be encoded by re-using PU mode on leaf node and re-calculating the intra prediction directions and motion vectors.

Figure 3.5: Example CTU structure and Frame Data that is re-used by the AT transcoding model
Figure 3.5 illustrates an example of possible CU splits and encoding combinations that will be considered by the AT model while traversing the input tree structure. Unlike the example in Figure 3.3 for the FPR model, where each split and PU/CU mode was determined from the input, in Figure 3.5 each split is only a possibility—denoted by dashed lines. The output CU structure will be a sub-set of input CU structure, meaning, there will be no input tree node that will be split farther. Therefore, every CU will be shallow or equal depth compared to input CUs. The final CU coding will be chosen as the one that is best in regards to Rate-Distortion criteria, which is the encoding that has the least distortion, measured by PSNR, to the input picture given the rate of encoding it, measured by bit rate.

3.5 Test Sequences and Encoder Configuration

Table 3.1 details the video sequences used in simulations. Test sequences are chosen from the standard set defined by JCT-VC (ITU and MPEG) for evaluating encoder models (Bossen, 2011), these sequences cover wide range of real world scenarios including: complex textures and motion. In this table, spatial and temporal resolutions are determined by Size and Frames per Second (FPS), respectively. Class is determined by the spatial resolution and could be: A) 2560 × 1600; B) 1920 × 1080; C) 1280 × 720; D) 832 × 480; or E) 416 × 240. All the sequences use 4:2:0 YUV color sampling.
Two main encoder configurations are used with each simulation: Low-Delay Main (LDM) and Random-Access Main (RAM). Important characteristics of these configurations are detailed in Table 3.2.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>MaxCUWidth</th>
<th>MaxCUHeight</th>
<th>MaxTreeDepth</th>
<th>GOP Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Access Main</td>
<td>64</td>
<td>64</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Low Delay Main</td>
<td>64</td>
<td>64</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.2: Encoder configurations used for encoding and transcoding

An important difference between LDM and RAM is the GOP structure. In RAM configuration, there is a periodic I-picture starting of each GOP, whereas in LDM configuration, there is a single I-picture for the whole bit stream. Presence of periodic I-picture in RAM configuration facilitates the random access behaviour.
3.6 Transcoder Evaluation Concepts

Transcoding performance is measured by calculating bit rate and PSNR for the input video and transcoded video. For illustration purposes, two set of plots are created: 1) PSNR-bit rate plots; and 2) Average overhead and bit rate ratio. Three PSNR and bit rate pairs are calculated as illustrated in Figure 3.6.

![Figure 3.6: PSNR calculations for transcoders](image)

In Figure 3.6, $PSNR_S$ represents the quality of the input video encoded with $QP_1$ at the bit rate $R_S$. $PSNR_T$ is the quality of the transcoded video with $QP_2$ at bit rate $R_T$ compared to the input video. $PSNR_B$ is the quality of the input video encoded with $QP_3$ at bit rate $R_B$ such that $PSNR_B$ will be equal to $PSNR_T$. To find $R_B$ it is necessary to find $QP_3$, however since it is difficult to find $QP_3$ an alternative approach is described in Figure 3.7.
Notice that in Figure 3.7 $PSNR$ of transcoded point is equal to the base point. Hence to obtain $R_B$, as suggested by Bjontegard (2001), it is sufficient to fit a curve to the RD points on the base plot and interpolate it for $PSNR$ of transcoded video, $PSNR_T$. This curve is an approximation to the encoder performance based on different QP values. For example, if there is a set of three RD pairs, polynomial of degree 3 is fit to obtain $f(PSNR_B)$:

$$R_B = f(PSNR_B) = c_4PSNR_B^3 + c_3PSNR_B^2 + c_2PSNR_B^1 + c_1$$

(3.1)

$R_B$ is obtained by setting $PSNR_B = PSNR_T$. In this work, monotonic piecewise cubic interpolation of Fritsch and Carlson (1980) is used to find the coefficients in the Equation 3.1. This interpolation method is chosen since it produces smooth curves that are easier to interpret.

### 3.6.1 Bit rate ratio

For bit rate reduction transcoding, it is convenient to quantify the reduction in inverse fraction of input bit rate. Denote the bit rate of an input bit-stream $R_S$ and bit rate after transcoding as $R_T$, the bit rate ratio is defined as:

$$r = \frac{R_S}{R_B}$$

(3.2)

This determines the ratio at which bit rate has been reduced in the transcoding process.
3.6.2 Overhead

It is interesting to know the bit rate of the video sequence that is encoded with the quality of the transcoded video. The difference between these bit rates, $R_B$ and $R_T$, are captured with the term \textit{overhead}. For given quality, overhead is the bit rate cost due to not having the original video available.

To further clarify the meaning of overhead, first define transcoding loss ($\Delta R$) as the difference between transcoded bit rate ($R_T$) and the base bit rate ($R_B$):

$$\Delta R = R_T - R_B$$  \hspace{1cm} (3.3)

Then overhead ($O$) is defined as the ratio between transcoding loss ($\Delta R$) and the base bit rate ($R_B$):

$$O = \frac{\Delta R}{R_B}$$  \hspace{1cm} (3.4)

The overhead demonstrates the price that has to be paid in terms of bit rate to maintain the PSNR. Lower overhead shows that the transcoder coding performance, for equal PSNR values, was close to that of initial encoder, in other words, if the raw video sequence was available and it has been encoded the RD performance was close to ($R_B, PSNR_B$). Notice that Overhead is defined for a fixed PSNR so to show the price paid for having to encode the video sequence with degraded PSNR due to not having access to the original video sequence.

3.6.3 Bit rate Ratio and Overhead Plots

To provide a reliable estimate for $r$ and $O$, each video sequence is encoded with $N$ different $QP$ values, and then the resulting bit-streams are transcoded with $M$ different $QP$ values each. Plotting the pairs $(r, O) = \{(r_1^n, O_1^n), ..., (r_m^n, O_m^n)\}$ provides a convenient way to summarize the performance of transcoding a single bit-stream.

As an example, Figure 3.8 shows the SCT model performance for the Kimono video sequence where the sequence is encoded with $N = 5$ different $QP$ and later transcoded with $M = 5$ different $QP$. Kimono video sequence is a full HD sequence with slow motion and high tone colors used commonly in coding performance evaluation. It is observed that for larger bit rate reduction the overhead of SCT model converges. This means for higher $QP$ values transcoding performance reaches that of lower $QP$. However, even in the best case for base $QP$ of 18, reducing the bit rate by factor of 4 requires 5% higher bit rate compared to direct encoding of original sequence.
From performance perspective, the ideal case is a transcoding model that has no overhead while reducing the bit rate, i.e., produces the same bit stream as an encoder that has access to original raw video. In reality this is impossible, since HEVC encoding is lossy and decoding the bit-stream will produce a raw video sequence that is unequal to the original.

### 3.6.4 Average Bit Rate Ratio and Overhead Plots

Assume there is a video sequence encoded with $QP^1$. In the simulations, the transcoder will produce a set of new bit-streams encoded with $QP_{1}^{1}, ..., QP_{m}^{1}$. For transcoded bit-streams there is a function fit to $RD$ points, $f(r^1, O^1) = \{(r_1^1, O_1^1), ..., (r_m^1, O_m^1)\}$. If the same video sequence is encoded with $QP^2$ and then transcoded with $QP_{1}^{2}, ..., QP_{m}^{2}$, there will be another chain of transcoded bit-streams, and a corresponding bit rate ratio and overhead curves, $f(r^2, O^2) = \{(r_1^2, O_1^2), ..., (r_m^2, O_m^2)\}$. Ultimately, for $n$ encoded bit-streams there will be $N$ corresponding $f(r^n, O^n)$. This is illustrated in Figure 3.9.

**Figure 3.8:** Bit rate ratio and overhead performance for Kimono 1920 × 1080 video sequence
To provide a suitable way to summarize the transcoder performance using the set of $n \times m$ pairs of $(r, O)$, a simple solution is to fit a curve to each $(r, O)$ set and average the result. The result is called average bit rate ratio and overhead, denoted by $\bar{f}(r, O)$.

In the reported results in this thesis five base and transcoded $QP$ values, as shown in Table 3.3, are used in simulations. Each row corresponds to an input bit-stream encoded with $QP_n$ and then transcoded with $QP_{nm}$. Base $QP$ values of 38, 42 and 46 are included to extend the PSNR–bit rate curve so to avoid extrapolation in the $RD$ plots.

<table>
<thead>
<tr>
<th>Base $QP$</th>
<th>Transcoder $QP$</th>
<th>$(r_{nm}, O_{nm})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$QP_1 = 18$</td>
<td>$QP_{1m} = {18,20,22,24,26}$</td>
<td>$(r_{1m}, O_{1m})$</td>
</tr>
<tr>
<td>$QP_2 = 22$</td>
<td>$QP_{2m} = {22,24,26,28,30}$</td>
<td>$(r_{2m}, O_{2m})$</td>
</tr>
<tr>
<td>$QP_3 = 26$</td>
<td>$QP_{3m} = {26,28,30,32,34}$</td>
<td>$(r_{3m}, O_{3m})$</td>
</tr>
<tr>
<td>$QP_4 = 30$</td>
<td>$QP_{4m} = {30,32,34,36,38}$</td>
<td>$(r_{4m}, O_{4m})$</td>
</tr>
<tr>
<td>$QP_5 = 34$</td>
<td>$QP_{5m} = {34,36,38,40,42}$</td>
<td>$(r_{5m}, O_{5m})$</td>
</tr>
<tr>
<td>38</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>42</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>46</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.3: $QP$ values used for base encoding and transcoding

$(r, O)$ pairs in Table 3.3 could also be visualized through a matrix:

$$
\begin{pmatrix}
(r_{11}, O_{11}) & \ldots & (r_{15}, O_{15}) \\
\vdots & \ddots & \vdots \\
(r_{51}, O_{51}) & \ldots & (r_{55}, O_{55}) \\
\end{pmatrix}
$$

(3.5)

In Matrix 3.5, each row corresponds to the input bit-stream encoded with the same $QP$. It is possible to fit a curve to model $f(r, O)$ in each row. Then averaging
the coefficients of these curves gives the average bit rate ratio and overhead curve, \( \bar{f}(r, O) \), as illustrated before in Figure 3.9. For example averaging the curves in Figure 3.8 results in one curve as shown in Figure 3.10.

![Figure 3.10: Average bit rate ratio and overhead performance for Kimono 1920 × 1080 video sequence](image)

The average curve corresponds to a video sequence created from concatenation of sub-sequences, where each sub-sequence corresponds to concatenation of sequences in each column of Matrix 3.5. This is also demonstrated in Figure 3.9. \( \bar{f}(r, O) \) corresponds to a single video sequence. It is also possible to group video sequence with similar characteristics and equal spatial resolution and calculate the overall \((r, O)\) presenting the transcoder performance for the group. Assume there is \( p \) sequences in class A, hence there is \( p \) average bit rate ratio and overhead functions. It is reasonable to average these functions to provide an overview performance curve of a transcoder. This average is denoted as:

\[
\bar{f}_A(r, O) = \frac{\bar{f}_1(r, O) + \cdots + \bar{f}_p(r, O)}{p}
\]  

(3.6)

The final curve, shown in Equation 3.6, could be derived for each transcoding method and drawing them in one plot provides a convenient way to compare the transcoding performance over several sequences where each sequence is encoded and transcoded with series of \( QP \) values. A hypothetical example of such plot is shown in Figure 3.11.
Figure 3.11: Transcoders performance through average bit rate ratio and overhead

By taking a quick glance at Figure 3.11, it is obvious that Transcoder B has performed better than Transcoder A in terms of RD. Using Transcoders A and B to reduce the bit rate to half the original amount requires 28% and 25% increased bit rate compared to encoding of original video sequence. Therefore transcoder B is more efficient than transcoder A.
Chapter 4

Results

The result for the performance of the developed video transcoder models, described in Chapter 3, is included in this chapter. At the time of writing this thesis, the latest reference HEVC encoder model was HM-8.2. The developed transcoders are based on this software test model.

4.1 Base Bit-stream

Each raw video sequence, shown in Table 3.1, is encoded and decoded twice (once for each encoder configuration) to produce the base curves of HEVC reference-encoder performance. The PSNR and bit rate for each encoding is measured and denoted as Rate-Distortion (RD) pair:

\[
\{(R_1^B, PSNR_1^B), (R_2^B, PSNR_2^B), \ldots, (R_8^B, PSNR_8^B)\},
\]

where each pair corresponds to the following quantization parameters:

\[
QP = \{18, 22, 26, 30, 34, 38, 42, 46\}.
\]

Total of 20 sequences were encoded twice with each QP value, hence 320 RD points for base curve has been produced. The chosen sequences (ref. Table 3.1) cover a wide range of real world scenarios. These scenarios range over sequences with complex textures, fast motion, video conferencing setup, vibrant colors, and more.

To demonstrate the transcoder performance a sequence is chosen and discussed through this chapter. A single sequence is used to be able to contrast between differences of the transcoding models while keeping the input video constant. An important transcoding application is the video conferencing scenario. The sequence Johnny demonstrates a case in point. RD values for sequence Johnny encoded with Low Delay Main (LDM) configuration is listed in Table 4.1. Low delay configuration is mostly used in video conferencing applications to minimize the delay between participants.
Chapter 4. Results

<table>
<thead>
<tr>
<th>RD</th>
<th>QP</th>
<th>$R_S$ (kbit/s)</th>
<th>$PSNR_S$ (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RD^1$</td>
<td>18</td>
<td>5848.13</td>
<td>44.38</td>
</tr>
<tr>
<td>$RD^2$</td>
<td>22</td>
<td>1508.13</td>
<td>42.96</td>
</tr>
<tr>
<td>$RD^3$</td>
<td>26</td>
<td>516.31</td>
<td>41.53</td>
</tr>
<tr>
<td>$RD^4$</td>
<td>30</td>
<td>252.09</td>
<td>39.89</td>
</tr>
<tr>
<td>$RD^5$</td>
<td>34</td>
<td>143.7</td>
<td>38.03</td>
</tr>
<tr>
<td>$RD^6$</td>
<td>38</td>
<td>87.68</td>
<td>35.91</td>
</tr>
<tr>
<td>$RD^7$</td>
<td>42</td>
<td>54.19</td>
<td>33.7</td>
</tr>
<tr>
<td>$RD^8$</td>
<td>46</td>
<td>34.77</td>
<td>31.72</td>
</tr>
</tbody>
</table>

Table 4.1: Base performance for sequence Johnny (1280 × 720) with LDM configuration

$QP \geq 38$ in Table 4.1 produces poor quality videos. Such high $QP$ values are included for completeness purposes in the base RD plots, and they are unused in simulations.

4.2 Simple Cascaded Transcoding Model

Each encoded sequence is sent to the transcoder to reduce the bit rate. Simple Cascaded Transcoding (SCT) model decodes the sequence and follows the procedure of reference encoder, with only difference being higher $QP$ value. The encoder will code the sequence with the goal of achieving highest coding performance, and since the encoder is not restricted by any means, it is reasonable to assume that the coding performance is the best possible transcoding performance.

Each coded sequence with the base $QP_B = \{18, 22, 26, 30, 34, 38\}$ is decoded and re-encoded with $QP_B + \Delta QP$, where $\Delta QP = \{0, 2, 4, 6, 8\}$. In addition to PSNR and bit rate, transcoding time (seconds) is also measured.

The SCT model performance for Johnny is shown in Table 4.2. Base bit rate, $R_B$, is calculated as described in Section 3.6.2 by fitting a curve to $RD_1$ through $RD_5$ values from Table 4.1 and solving it for $PSNR_T$ values from Table 4.2.

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B$ (kbps)</th>
<th>$R_T$ (kbit/s)</th>
<th>$PSNR_T$ (dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>786.4</td>
<td>1271.12</td>
<td>42.29</td>
<td>1.92</td>
<td>61.64</td>
</tr>
<tr>
<td>2</td>
<td>574.91</td>
<td>803.28</td>
<td>41.74</td>
<td>2.62</td>
<td>39.72</td>
</tr>
<tr>
<td>4</td>
<td>410.03</td>
<td>496.09</td>
<td>41.10</td>
<td>3.68</td>
<td>20.99</td>
</tr>
<tr>
<td>6</td>
<td>301.37</td>
<td>338.16</td>
<td>40.40</td>
<td>5</td>
<td>12.21</td>
</tr>
<tr>
<td>8</td>
<td>229.56</td>
<td>247.28</td>
<td>39.62</td>
<td>6.57</td>
<td>7.72</td>
</tr>
</tbody>
</table>

Table 4.2: SCT model performance with LDM configuration for sequence Johnny–Base $QP:22$, $R_S:1508.13$ (kbit/s), $PSNR_S:42.96$ (dB)

To achieve higher bit rate reduction, $QP$ value is increased. For example, as
it is noted in Table 4.2, $\Delta QP = 6$ reduces the bit rate 5-folds while requiring $\sim 12\%$ overhead. In other words, to reduce the bit rate to one fifth, SCT model requires $\sim 12\%$ higher bit rate compared to direct encoding of original raw video sequence with matching bit rate and PSNR.

In addition to bit rate ratio and overhead plots, a common method to illustrate the coding performance in details is through the PSNR–bit rate curves (Lefol et al., 2006). The plots illustrating the RD transcoding performance for Johnny sequence is shown in Figure 4.1. The red-line in this plot corresponds to $(R_T, PSNR_T)$ pairs in Table 4.2.

![Figure 4.1: SCT model Rate-Distortion plot for Johnny (1280×720)](image)

An observation from Figure 4.1 is that the RD performance of SCT model gets closer to base curve as the transcoding $QP$ increases. One explanation for this behaviour could be that the base RD performance is decreasing for high $QP$ values faster than SCT model performance. This is because for high $QP$ values most of the coding information will be lost during quantization and no
matter how precise the search for MV and modes is, the residuals will be mostly
truncated to zero in quantization stage and it will be very difficult to maintain
the original video quality. Hence, SCT model that has access to lower quality
video will, nevertheless, produce similar results.

4.3 Full Prediction Reuse Model

Full Prediction Reuse (FPR) transcoding model has been explain in Section 3.1.
The aim is to re-use all the information available in the input bit-stream about:
Coding Tree Unit and Coding Units. In principle this model will be the fastest
model since most time consuming stages, e.g., motion estimation, of encoding are
bypassed. It is also interesting to observe the changes in transcoding performance
compared to SCT model. The FPR transcoding performance for Johnny sequence
is shown in Table 4.3.

<table>
<thead>
<tr>
<th>∆QP</th>
<th>$R_P$ (kbit/s)</th>
<th>$R_T$ (kbit/s)</th>
<th>$PSNR_T$ (dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>901.07</td>
<td>1367.44</td>
<td>42.48</td>
<td>1.67</td>
<td>51.76</td>
</tr>
<tr>
<td>2</td>
<td>605.99</td>
<td>983.48</td>
<td>41.84</td>
<td>2.49</td>
<td>62.29</td>
</tr>
<tr>
<td>4</td>
<td>418.92</td>
<td>727.08</td>
<td>41.14</td>
<td>3.60</td>
<td>73.56</td>
</tr>
<tr>
<td>6</td>
<td>303.04</td>
<td>606.9</td>
<td>40.41</td>
<td>4.98</td>
<td>100.27</td>
</tr>
<tr>
<td>8</td>
<td>228.38</td>
<td>542.42</td>
<td>39.61</td>
<td>6.60</td>
<td>137.51</td>
</tr>
</tbody>
</table>

Table 4.3: FPR transcoding performance with LDM configuration for sequence
Johnny–Base $QP:22$, $R_S:1508.13$ (kbit/s), $PSNR_S:42.96$ (dB)

A striking observation of Table 4.3 is the increasing overhead trend of FPR
model compared to decreasing overhead trend of SCT model in Table 4.2. Over-
head values over 100% means that the transcoding has reduced the video quality
so much that it requires the same input bit rate to compensate for the loss.

The results in Table 4.3 suggests that FPR transcoding performance for bit
rate ratio over 3.6 ($\Delta QP \geq 6$) is inefficient, which in turn means that reusing
motion information and tree structure for such high $QP$ values produces sub-
optimal coding performance.
PSNR–bit rate performance for several base $Q_P$ values for FPR transcoding is plotted in Figure 4.2. As expected, compared to SCT model the RD curves are diverting from the base curve with increase in $Q_P$ and such behaviour results in higher overhead, since the quality is dropping faster than bit rate. However notice that the RD performance for higher base $Q_P$ values are closer to the base curve compared to lower base $Q_P$. This could be better understood by examining Table 4.4.
Chapter 4. Results

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B (kbps)$</th>
<th>$R_T (kbit/s)$</th>
<th>$PSNR_T (dB)$</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>125.11</td>
<td>137.78</td>
<td>37.47</td>
<td>1.15</td>
<td>10.13</td>
</tr>
<tr>
<td>2</td>
<td>97.07</td>
<td>126.57</td>
<td>36.37</td>
<td>1.48</td>
<td>30.4</td>
</tr>
<tr>
<td>4</td>
<td>77.54</td>
<td>116.27</td>
<td>35.35</td>
<td>1.85</td>
<td>49.96</td>
</tr>
<tr>
<td>6</td>
<td>64.32</td>
<td>109.49</td>
<td>34.49</td>
<td>2.23</td>
<td>70.24</td>
</tr>
<tr>
<td>8</td>
<td>52.79</td>
<td>104.6</td>
<td>33.58</td>
<td>2.72</td>
<td>98.13</td>
</tr>
</tbody>
</table>

Table 4.4: FPR transcoding performance with LDM configuration for sequence Johnny–Base $QP:34$, $R_S:143.7$ ($kbit/s$), $PSNR_S:38.03$ ($dB$)

Compared to the performance reported in Table 4.3, FPR transcoding performance is better with base $QP = 34$. For example, for $\Delta QP = 6$ the overhead is 70.24% compared to 100.27%. However it should be noted that the overhead depends both on the base and transcoded RD performance, hence lower overhead could be the result of worse RD performance of the base encoding. This behaviour is also observed for SCT model in Figure 4.1. RD performance for SCT model from higher base $QP$ values are better than lower base $QP$.

4.4 Intra Prediction Re-estimation Model

Intra Prediction Re-estimation (IPR) transcoding model is similar to FPR model. The difference is in the way that I-slices are handled. IPR transcoding model re-estimates the I-slice prediction data. The motivation for developing this method is that I-slices could improve the coding efficiency of other slices by providing accurate prediction data. This is specially true for random access configuration of the transcoder where each GOP is composed of eight frames and each starting frame is an I-frame.

To evaluate the IPR model, two tests are considered: 1) a sequence is transcoded with FPR model and IPR model using random access configuration; and 2) a sequence is transcoded with FPR model using random access and low delay configurations. Random access configuration is important for video playback and video editing. BQTerrace ($1920 \times 1080$) video sequence is used in these tests since it provides a typical public scene with motion and textures.
By comparing the overheads reported in Table 4.5, it is obvious that IPR transcoding model demonstrates a better performance, since the overheads are lower in every case. IPR model performs even better for higher QP values, 18.82% overhead difference ($\Delta QP = 8$) compared to 5.16% ($\Delta QP = 2$). Which means coding I-slices with better quality is important for higher bit rate reduction.

To farther support the claim that FPR model has higher transcoding performance for random access configuration compared to low delay, transcoding performance of sequence Johnny with both configurations is reported in Table 4.6.

LDM configuration has single I-slice for a bit-stream opposed to one I-slice per GOP of RAM configuration. Based on this fact, roughly 50% lower overhead of IPR transcoding, as noted in Table 4.6, is due to motion re-estimation for higher number of I-slices.
Chapter 4. Results

4.5 Motion Re-estimation Model

Motion Re-estimation (MR) transcoding model is developed to understand the transcoder performance when the transcoder is granted with the flexibility to re-estimate the motion data for every frame, however constrained by the quad-tree structure that is copied from the input bit-stream.

Naturally one would expect that granting transcoder the ability to optimize the block coding would increase transcoding performance. However it is observed that reusing tree structure and PU splitting of input bit-stream has negativity effected the performance for $\Delta QP = 0$, which is shown in Table 4.7.

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B (kbps)$</th>
<th>$R_T (kbit/s)$</th>
<th>$PSNR_T (dB)$</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>754.41</td>
<td>1324.5</td>
<td>42.22</td>
<td>2</td>
<td>75.57</td>
</tr>
<tr>
<td>2</td>
<td>553.61</td>
<td>880.77</td>
<td>41.66</td>
<td>2.72</td>
<td>59.09</td>
</tr>
<tr>
<td>4</td>
<td>387.49</td>
<td>579.03</td>
<td>40.99</td>
<td>3.89</td>
<td>49.43</td>
</tr>
<tr>
<td>6</td>
<td>285.36</td>
<td>419.46</td>
<td>40.25</td>
<td>5.29</td>
<td>46.99</td>
</tr>
<tr>
<td>8</td>
<td>215.98</td>
<td>324.03</td>
<td>39.44</td>
<td>6.98</td>
<td>50.03</td>
</tr>
</tbody>
</table>

Table 4.7: MR transcoding performance with LDM configuration for sequence Johnny –Base $QP: 22$, $R_S: 5848.13 \ (kbit/s)$, $PSNR_S: 44.38 \ (dB)$

Overhead for FPR model with $\Delta QP = 0$ was reported 51.76% in Table 4.3, whereas the corresponding overhead for MR model has increased by 23.81%. The reason could be that local optimization of motion data has a negative effect on the global coding performance. The tree structure extracted from input bit-stream has been optimized in conjunction with the appropriate motion data in a global manner, and reusing that structure with different motion data leads to poor performance.

Figure 4.3 illustrates the average bit rate ratio and overhead results for MR and FPR transcoding. It is clear that the rising trend for FPR model has been reversed with MR model. Using RAM and LDM configurations the bulk of frames are encoded with inter prediction mode, hence correct MV has an important factor in transcoder performance. It is also noted that the main difference between MR and FPR transcoding models is the motion vector re-estimation for MV and motion vector re-use for FPR. By increasing the $QP$ value to achieve higher bit rate reduction ratios, re-using MV produces poor quality as it is seen by the increasing trend of overhead for FPR model, whereas re-estimating motion vectors by MV model produces better performance by adapting to the higher $QP$ values.
Comparing the PSNR–bit rate performance of FPR and MV models also shows a big improvement. It is observed in Figure 4.4 that the divergent behaviour of FPR model for higher \( \Delta QP \) values has been fixed in MV model.

**Figure 4.4:** FPR and MV transcoding performance with LDM configuration for sequence Johnny
4.6 Advanced Transcoding Model

Advanced Transcoding model is designed with the goal of reducing the overhead of previous transcoding models while maintaining good time complexity performance. To accomplish such goal it is necessary to extend the search space of CU structures, and try other block encoding combinations that are different from block information of input bit-stream. It is also important to note that the space of possible combinations between CU tree structures, PU splits, and block modes is huge.

To reduce the search space the following observations about HEVC standard are considered: 1) Skipped blocks require least number of bits to encode, hence, for higher QP values it is reasonable to try to transcode as many skipped blocks as possible; 2) $2N \times 2N$ PU split requires one MV to be signaled, therefore it has a higher chance of reducing the overhead, since it requires less bits; 3) Having access to the CU tree structure of input bit-stream it is possible to produce a shallow tree structure by pruning the input tree. Pruned tree will have less details and require less bits to be coded.

In Section 4.5, it was observed that MV model has demonstrated better transcoding performance compared to FPR due to re-estimation of MV for inter blocks and re-estimation of motion directions for I-frames. Improvement of AT transcoding performance over MV model is shown in Table 4.8.

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B$ (kbps)</th>
<th>$R_T$ (kbps)</th>
<th>$PSNR_T$ (dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>754.41</td>
<td>1324.5</td>
<td>42.22</td>
<td>2</td>
<td>75.57</td>
</tr>
<tr>
<td>2</td>
<td>553.61</td>
<td>880.77</td>
<td>41.66</td>
<td>2.72</td>
<td>59.09</td>
</tr>
<tr>
<td>4</td>
<td>387.49</td>
<td>579.03</td>
<td>40.99</td>
<td>3.89</td>
<td>49.43</td>
</tr>
<tr>
<td>6</td>
<td>285.36</td>
<td>419.46</td>
<td>40.25</td>
<td>5.29</td>
<td>46.99</td>
</tr>
<tr>
<td>8</td>
<td>215.98</td>
<td>324.03</td>
<td>39.44</td>
<td>6.98</td>
<td>50.03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B$ (kbps)</th>
<th>$R_T$ (kbps)</th>
<th>$PSNR_T$ (dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>746.98</td>
<td>1253.78</td>
<td>42.21</td>
<td>2.02</td>
<td>67.85</td>
</tr>
<tr>
<td>2</td>
<td>552.65</td>
<td>810.49</td>
<td>41.66</td>
<td>2.73</td>
<td>46.66</td>
</tr>
<tr>
<td>4</td>
<td>393.5</td>
<td>506.68</td>
<td>41.02</td>
<td>3.83</td>
<td>28.76</td>
</tr>
<tr>
<td>6</td>
<td>291.48</td>
<td>349.99</td>
<td>40.31</td>
<td>5.17</td>
<td>20.08</td>
</tr>
<tr>
<td>8</td>
<td>221.77</td>
<td>257.8</td>
<td>39.52</td>
<td>6.8</td>
<td>16.25</td>
</tr>
</tbody>
</table>

*Table 4.8:* MV model (top) and AT (bottom) model transcoding performance with LDM configurations for sequence Johnny–Base $QP:22$, $R_S:1508.13$ (kbps), $PSNR_S:42.96$ (dB)

The design goal of AT model was to produce performance close to SCT model. Table 4.9 demonstrates that the performance of these two models are very close. This is also observed in Figure 4.5, where SCT and AT exhibit very close performance.
<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B$(kbps)</th>
<th>$R_T$(kbit/s)</th>
<th>$PSNR_T$(dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>786.4</td>
<td>1271.12</td>
<td>42.29</td>
<td>1.92</td>
<td>61.64</td>
</tr>
<tr>
<td>2</td>
<td>574.91</td>
<td>803.28</td>
<td>41.74</td>
<td>2.62</td>
<td>39.72</td>
</tr>
<tr>
<td>4</td>
<td>410.03</td>
<td>496.09</td>
<td>41.1</td>
<td>3.68</td>
<td>20.99</td>
</tr>
<tr>
<td>6</td>
<td>301.37</td>
<td>338.16</td>
<td>40.4</td>
<td>5</td>
<td>12.21</td>
</tr>
<tr>
<td>8</td>
<td>229.56</td>
<td>247.28</td>
<td>39.62</td>
<td>6.57</td>
<td>7.72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta QP$</th>
<th>$R_B$(kbps)</th>
<th>$R_T$(kbit/s)</th>
<th>$PSNR_T$(dB)</th>
<th>$r$</th>
<th>Overhead %</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>746.98</td>
<td>1253.78</td>
<td>42.21</td>
<td>2.02</td>
<td>67.85</td>
</tr>
<tr>
<td>2</td>
<td>552.65</td>
<td>810.49</td>
<td>41.66</td>
<td>2.73</td>
<td>46.66</td>
</tr>
<tr>
<td>4</td>
<td>393.5</td>
<td>506.68</td>
<td>41.02</td>
<td>3.83</td>
<td>28.76</td>
</tr>
<tr>
<td>6</td>
<td>291.48</td>
<td>349.99</td>
<td>40.31</td>
<td>5.17</td>
<td>20.08</td>
</tr>
<tr>
<td>8</td>
<td>221.77</td>
<td>257.8</td>
<td>39.52</td>
<td>6.8</td>
<td>16.25</td>
</tr>
</tbody>
</table>

Table 4.9: SCT (top) and AT (bottom) transcoding performance with LDM configurations for sequence Johnny–Base $QP:22$, $R_S:1508.13(kbit/s)$, $PSNR_S:42.96(dB)$

Figure 4.5: SCT and AT transcoding performance with LDM configuration for sequence Johnny
Chapter 4. Results

4.7 Class Average Performance

As described in Section 3.6.4, transcoder performance could be averaged across sequences of the same class. This will provide a better overview of how each transcoding method performs given video sequences with certain characteristics.

According to Table 3.1, 20 sequences are classified into five categories depending on the spatial resolution. Upper Class sequences have higher spatial resolution, therefore they will contain higher number of CTU. Figure 4.6 illustrates the class performance of each developed transcoder model in conjunction with SCT model.

It is observed that using higher $Q_P$ to reduce the bit rate, FPR and IPR models are unable to maintain high bit rate reduction. This is observed for average bit rate ratio values above 2.0 in Figure 4.6. For every sequence class, FPR and IPR has a rising overhead trend, which supports the observations made in Section 4.3 and Section 4.4 that the input block information without modification becomes less efficient for reducing bit rate over half the input bit rate.

FPR and IPR models has demonstrated worse performance for class C sequences. For class A, those models have shown better transcoding performance upto average bit rate ratio of 2.0 compared to MV model. Except the performance of FPR and IPR models for sequence class C, in each case the overhead is below 50% regardless of bit rate ratio.

MV model has consistent performance trend through every class, where approximately equal overhead is required for reducing the bit rate independent from the bit rate ratio. This trend falls between rising trends of FPR-IPR models and falling trends of SCT-AT models. This is interesting, since the re-use of CU tree structure and re-estimation of MV and intra prediction directions provides an equal trade-off between quality loss due to re-use and quality gain due to re-estimations.

The goal of AT model design was to exhibit a close performance to that of SCT model. It is clear that this goal is achieved by observing and comparing the AT and SCT model performance in Figure 4.6. In general, compared to FPR, IPR, and MV models, AT model has performed the best. Across the sequence classes, AT has performed worse for sequence class C and best for class D, with an approximate difference of 4% in overhead.

As expected, for bit rate ratios above 2.0, SCT has maintained the best transcoding performance with lowest overhead, whereas, AT model was second. bit rate ratio above 2.6 was unachievable with FPR and IPR models.
Figure 4.6: Class average bit rate ratio and overhead transcoding performance with LDM configuration
Per sequence class transcoding performance with RAM configuration is illustrated in Figure 4.7.

The major difference between transcoder performance with RAM configuration compared to LDM configuration is the lower overhead of IPR model. RAM configuration incorporated an I-frame every eight frames (one GOP of 8 frames) compared to single I-frame of LDM configuration. These I-frames are used as reference frames for inter prediction, and since they are only intra predicted with higher quality, the following frames will have better reference quality. The effect of such GOP structure for RAM configuration is the lower overhead for IPR model compared to LDM configuration. Since IPR re-calculates the intra prediction directions for I-frames.

MV model is similar to IPR in a sense that I-frames are encoded with re-estimated intra prediction directions. Therefore in a similar manner to IPR, as it is observed in Figure 4.7, the overhead of MV model for RAM configuration is approximately 10% lower than LDM configuration. This observation is consistent for every sequence class.

FPR model has lower overhead for lower $Q_P$ values with RAM configuration compared to LDM, as it is observed for $r < 2.0$. Overhead for $r < 2.0$ corresponds to transcoding with equal $Q_P$ to base encoder. Therefore, reusing block information results in better global coding performance compared to other transcoding models which locally optimize the block structure. For higher $Q_P$ values, however, similar to RAM configuration the transcoder performance drops significantly.
Figure 4.7: Class average bit rate ratio and overhead transcoding performance with RAM configuration
Chapter 4. Results

4.8 Time

In Section 4.7, it was observed that SCT model has the best coding performance compared to the other transcoding models, which justifies its use as the reference. The problem with SCT model is the high computational requirements. To further illustrate this point, transcoding time differences are illustrated in Figure 4.8 for each sequence classes transcoded with LDM configuration. In each plot, the maximum transcoding time is used to normalize the others into the range 0-100. In every case, the maximum transcoding time corresponds to SCT model with $\Delta QP = 0$. The tests were run on Linux operating system with Intel Xeon 2.2 GHz processor. The transcoding program execution time is measured with the Linux’s built in time command.

Time is measured with relativeness to SCT model time. Therefore exact measurement in seconds

There is a visible decreasing trend for SCT and AT models for higher $QP$ values. High $QP$ usually produces CU tree structures that are shallow compared to lower $QP$ values, which is due to the fact that higher $QP$ will cause most of the coefficients from prediction stage to be truncated to zero, therefore the encoder will most likely encode more blocks with SKIP mode compared to lower $QP$, and as a result, the CU will stop splitting to smaller sub-CUs and therefore the final tree will be shallow. Finally, stopping the branching in lower levels of CU structure will make the search space smaller which leads to shorter transcoding time.

As expected, transcoder models with highest information re-use require the least transcoding time. FPR and IPR models require approximately equal transcoding time which is about 1% of SCT transcoding time. Motion vector re-estimation of MV model increases the transcoding time to 5%. AT model performs differently based on sequence class. Fastest transcoding time for AT model was recorded for class C with 10%. Around 20% transcoding time was observed for highest resolution sequence class for AT model. In general, transcoding time for AT model compared to SCT model has shown a decrease between 80% to 90%.

Per sequence class transcoding time with RAM configuration is shown in Figure 4.9. There is no visible difference between the transcoding times compared to LDM model in terms of percentage to maximum SCT time. However transcoding RAM configuration is faster than LDM, since RAM incorporates many I-frames, and intra prediction is faster than inter prediction. For example encoding the video sequence SteamLocomotiveTrain with spatial resolution of $2560 \times 1600$ with $QP = 22$ requires an average of 228 seconds per frame for LDM configuration which is greater compared to 159 seconds per frame of RAM configuration.
Chapter 4. Results

Figure 4.8: Transcoding time with LDM configuration
Chapter 4. Results

Figure 4.9: Transcoding time with RAM configuration
Chapter 5

Conclusions

Transcoding is necessary to enable interoperability of devices with heterogeneous computational resources. Previously developed transcoding methods are insufficient for transcoding bit-streams compatible with H.265/HEVC video coding standard, which is sought to be an important part of video communication systems in near future (Li et al., 2012).

An important part of H.265/HEVC design is the quad-tree based structure of CUs. This structure provides flexible coding design and higher coding performance (Kim et al., 2012), however, searching the space of possible structures is computationally expensive. This work has investigated the transcoding methods that reduces the search space by reusing the CU structure information available through the input bit-stream.

5.1 Overview of Proposed Transcoding Models

This worked focused on transcoding H.265/HEVC bit-streams for bit rate reduction and transcoder evaluation methods. In this regard, four transcoding methods were developed: 1) Full Prediction Re-use (FPR) model: this model re-uses CU information for intra- and inter-prediction but re-calculates residuals and coefficients; 2) Intra Prediction Re-estimation (IPR): this model is similar to FPR with the difference of re-estimating prediction data for intra coded frames; 3) MV Re-estimation (MR): this model is similar to IPR with the difference of re-estimating motion vectors after copying the CU structure from input bit-stream; and 4) Advanced Transcoding (AT): this model is a combination of previous models with specific additions to push the transcoder performance farther by efficiently extending the search space of block coding structure.

5.2 Overview of Transcoders Performance

It has been observed that re-using motion data in conjunction to CU structure, FPR an IPR models, has limited performance for bit rate reduction below half the input bit rate. However, as it was expected, FPR and IPR models that re-use
CU information were very fast. Compared to SCT model, FPR and IPR models required 1% of the transcoding time. It was also noted that motion vector re-estimation model, MR, has inverted the increasing trend of overhead for FPR and IPR with small addition of computational complexity that is approximately 5% of SCT model.

Finally, AT model was designed with consideration of the following observations: 1) Using skip mode is likely to reduce the overhead since it requires the least number of bits for encoding; 2) PU split mode with one motion vector \((2N \times 2N)\) requires only one motion vector to be signaled; and 3) Merging blocks reduces the number of bits. It was observed that AT has competitive performance to that of SCT within margin of 5% difference while reducing the transcoding time by approximately 80%.

## 5.3 Future Work

Research questions in the area of bit rate reduction for transcoding HEVC bit-streams has been investigated in this work. Transcoding for spatial-resolution reduction is also important subject, for which the performance of Simple Cascaded Transcoding is included in Appendix B. However the are specific challenges for spatial-resolution reduction that requires farther investigations. For example the motion vectors of the input bit-stream can not be used directly, since the input image blocks are smaller, therefore new policies, such as averaging motion vectors, could be developed and investigated.

The reference implementation of HEVC encoder (HM-8.2) is not meant to be a real-time encoder. It is interesting to farther investigate the performance implications of using the proposed transcoder models in the real-time encoder implementations.

Another interesting transcoding task is the conversion between CODECs of AVC and HEVC. For example, it would be interesting to investigate the performance of a transcoder that uses Skip CU mode at the root of tree structure that corresponds to a macroblock smaller than \(16 \times 16\) pixel.
Appendix A
Algorithms of The Proposed Transcoder Models

Each transcoder model is detailed with the help of pseudo-code. Table A.1 details the functions used to describe the transcoding algorithms.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbsAddr(CU, addr)</td>
<td>Absolute address of CU relative to the address of LCU</td>
</tr>
<tr>
<td>D(CU, absAddr)</td>
<td>Depth of CU at absAddr</td>
</tr>
<tr>
<td>Sub(CU, d, absAddr)</td>
<td>Sub-partition CU or Slice relative to depth d at absAddr</td>
</tr>
<tr>
<td>Count(CU)</td>
<td>Number of sub-CUs in CU or slice</td>
</tr>
<tr>
<td>PartSize(CU, absAddr)</td>
<td>Partition size of CU at absAddr</td>
</tr>
<tr>
<td>PredMode(CU, absAddr)</td>
<td>Prediction mode of CU at absAddr</td>
</tr>
<tr>
<td>IntraData(CU, absAddr)</td>
<td>Intra prediction directions and modes of CU at absAddr</td>
</tr>
<tr>
<td>MV(CU, absAddr)</td>
<td>Motion vector of CU at absAddr</td>
</tr>
<tr>
<td>SkipFlag(CU, absAddr)</td>
<td>Skip flag of CU at absAddr</td>
</tr>
<tr>
<td>MergeFlag(CU, absAddr)</td>
<td>Merge flag of CU at absAddr</td>
</tr>
<tr>
<td>SelectBest(CU, CU_T, ...)</td>
<td>Select the best CU_T, based on Rate-Distortion and copy to CU</td>
</tr>
<tr>
<td>Copy(CU, ...)</td>
<td>Copy the given CU information into CU</td>
</tr>
<tr>
<td>EstPredData(CU, PartSize, ...)</td>
<td>Estimate the prediction data for CU with PartSize</td>
</tr>
<tr>
<td>Encode(CU)</td>
<td>Encode CU</td>
</tr>
</tbody>
</table>

**Table A.1:** Function definitions used for describing transcoding models

Each frame is made of slices. Slice is made of CTUs of equal sizes. Algorithm 1 loops over these CTUs to transcode the slice. A CTU is made of CUs, and since it is the largest of those CUs it is also called LCU (Largest Coding Unit). **TranscodeCU** function could evoke any of the proposed transcoding models described by Algorithm 2, 3, 4, and 5.
Algorithm 1 Transcoder Main Loop

**Input:** \(\text{Slice}_I = \) Input bit-stream slice

**Output:** \(\text{Slice}_O = \) Transcoded slice

1. **function** Transcode(\(\text{Slice}_O, \text{Slice}_I\))
2. \(\text{numCTU} \leftarrow \text{Count} (\text{Slice}_O)\)
3. \(i \leftarrow 0\)
4. **while** \(i < \text{numCTU} \) do \(\triangleright\) loop over every LCU in the frame
5. \(\text{CTU}_I \leftarrow \text{Sub} (\text{Slice}_I, 0, i)\)
6. \(\text{CTU}_O \leftarrow \text{Sub} (\text{Slice}_O, 0, i)\)
7. TranscodeCU(\(\text{CTU}_O, \text{CTU}_I, 0\)) \(\triangleright\) Transcode CTU
8. \(i \leftarrow i + 1\)
9. **end while**
10. **end function**

Algorithm 2 FPR Transcoding Model

**Input:** \(\text{CU}_I = \) CU from input bit-stream, \(d_O = 0\)

**Output:** \(\text{CU}_O = \) Transcoded CU

1. **function** TranscodeCU(\(\text{CU}_O, \text{CU}_I, d_O\))
2. \(\text{absAddr} \leftarrow \text{AbsAddr} (\text{CU}_O, 0)\)
3. \(d_I \leftarrow \text{D}(\text{CU}_I, \text{absAddr})\)
4. **if** \(d_I = d_O\) \(\triangleright\) branching continues until leaf node is reached
5. \(\text{predMode} \leftarrow \text{PredMode} (\text{CU}_I, \text{absAddr})\)
6. \(\text{partSize} \leftarrow \text{PartSize} (\text{CU}_I, \text{absAddr})\)
7. **if** \(\text{predMode} = \text{Intra}\)
8. \(\text{intraData} \leftarrow \text{IntraData} (\text{CU}_I, \text{absAddr})\)
9. \(\text{Copy} (\text{CU}_O, \text{Intra}, \text{intraData}, \text{partSize})\) \(\triangleright\) intra mode \(\text{CU}_O\)
10. **else**
11. \(\text{mv} \leftarrow \text{MV} (\text{CU}_I, \text{absAddr})\)
12. \(\text{skipFlag} \leftarrow \text{SkipFlag} (\text{CU}_I, \text{absAddr})\)
13. \(\text{mergeFlag} \leftarrow \text{MergeFlag} (\text{CU}_I, \text{absAddr})\)
14. \(\text{Copy} (\text{CU}_O, \text{Inter}, \text{mv}, \text{skipFlag}, \text{mergeFlag}, \text{partSize})\) \(\triangleright\) inter mode \(\text{CU}_O\)
15. **end if**
16. **end if** \(\triangleright\) complete encoding including: prediction, transforms, quantization, entropy coding
17. \(\text{CU}_O \leftarrow \text{Encode} (\text{CU}_O)\)
18. **else if** \(d_O < d_I\) \(\triangleright\) current CU is split to four sub-CUs
19. \(i \leftarrow 0\)
20. \(d_O \leftarrow d_O + 1\)
21. **while** \(i < 4\) do
22. \(\text{subCU}_O \leftarrow \text{Sub} (\text{CU}_O, d_O, \text{absAddr})\)
23. \(\text{TranscodeCU} (\text{subCU}_O, \text{CU}_I, d_O)\)
24. \(\text{absAddr} \leftarrow \text{absAddr} + \text{Count} (\text{subCU}_O)\)
25. \(i \leftarrow i + 1\)
26. **end while**
27. **end if**
28. **end function**
Algorithm 3 IPR Transcoding Model

1: function $\text{TranscodeCU}(CU_O, CU_I, d_O)$
2: \hspace{1em} $\text{absAddr} \leftarrow \text{AbsAddr}(CU_O, 0)$
3: \hspace{1em} $d_I \leftarrow D(CU_I, \text{absAddr})$
4: \hspace{1em} if $d_I = d_O$ then \quad \triangleright \text{branching continues until leaf node is reached}
5: \hspace{2em} $\text{partSize} \leftarrow \text{PartSize}(CU_I, \text{absAddr})$
6: \hspace{1em} if $CU_O$ in I-slice then \quad \triangleright \text{intra directions are recalculated for I-slices}
7: \hspace{2em} $\text{intraData} \leftarrow \text{EstPredData}(CU_O, \text{Intra}, \text{partSize})$
8: \hspace{2em} $\text{Copy}(CU_O, \text{Intra}, \text{intraData}, \text{partSize})$
9: \hspace{1em} else
10: \hspace{2em} $\text{predMode} \leftarrow \text{PredMode}(CU_I, \text{absAddr})$
11: \hspace{2em} if $\text{predMode} = \text{Intra}$ then \quad \triangleright \text{intra mode } CU_O
12: \hspace{3em} $\text{intraData} \leftarrow \text{IntraData}(CU_I, \text{absAddr})$
13: \hspace{2em} $\text{Copy}(CU_O, \text{Intra}, \text{intraData}, \text{partSize})$
14: \hspace{1em} else
15: \hspace{2em} $\text{mv} \leftarrow \text{MV}(CU_I, \text{absAddr})$
16: \hspace{2em} $\text{skipFlag} \leftarrow \text{SkipFlag}(CU_I, \text{absAddr})$
17: \hspace{2em} $\text{mergeFlag} \leftarrow \text{MergeFlag}(CU_I, \text{absAddr})$
18: \hspace{2em} $\text{Copy}(CU_O, \text{Inter}, \text{mv}, \text{skipFlag}, \text{mergeFlag}, \text{partSize})$ \quad \triangleright \text{inter mode } CU_O
19: \hspace{1em} end if
20: \hspace{1em} \quad \triangleright \text{complete encoding including: prediction, transforms, quantization, entropy coding}
21: \hspace{2em} $\text{Encode}(CU_O)$
22: \hspace{1em} else if $d_O < d_I$ then \quad \triangleright \text{current CU is split to four sub-CUs}
23: \hspace{2em} $i \leftarrow 0$
24: \hspace{2em} $d_O \leftarrow d_O + 1$
25: \hspace{2em} while $i < 4$ do
26: \hspace{3em} $\text{subCU}_O \leftarrow \text{Sub}(CU_O, d_O, \text{absAddr})$
27: \hspace{3em} $\text{TranscodeCU}(\text{subCU}_O, CU_I, d_O)$
28: \hspace{3em} $\text{absAddr} \leftarrow \text{absAddr} + \text{Count}(\text{subCU}_O)$
29: \hspace{3em} $i \leftarrow i + 1$
30: \hspace{2em} end while
31: \hspace{1em} end if
32: \hspace{1em} end function
Algorithm 4: MR Transcoding Model

1: function TranscodeCU(CU₀, CU₁, d₀)
2:     absAddr ← AbsAddr(CU₀, 0)
3:     d₁ ← D(CU₁, absAddr)
4:     skipFlag ← SkipFlag(CU₁, absAddr)
5:     partSize ← PartSize(CU₁, absAddr)
6:     if d₁ = d₀ then
7:         if skipFlag = true then
8:             interData ← EstPredData(CU₀, Inter, skip, merge, 2N × 2N)
9:             Copy(CU₀, Inter, skip, merge, partSize, interData)
10:     else if CU₀ not in I-Slice then
11:         mergeFlag ← MergeFlag(CU₁, absAddr)
12:         interData ← EstPredData(CU₀, Inter, partSize)
13:         Copy(CU₀, Inter, mergeFlag, partSize, interData)
14:     else
15:         intraData ← EstPredData(CU₀, Intra, partSize)
16:         Copy(CU₀, Intra, partSize, intraData)
17:     end if
18:     ⊳ branching continues until leaf node is reached
19:     if d₁ = d₀ then
20:         ⊳ re-calculate MV index
21:         ⊳ re-calculate MV
22:         ⊳ re-calculate intra directions
23:     end if
24:     ⊳ complete encoding including: prediction, transforms, quantization, entropy coding
25:     CU₀ ← Encode(CU₀)
26:     if d₀ < d₁ then
27:         ⊳ current CU is split to four sub-CUs
28:         ⊳ current CU is split to four sub-CUs
29:         ⊳ current CU is split to four sub-CUs
30:     end function
Algorithm 5 AT Model

1: \textbf{function} TranscodeCU(CU\textsubscript{O}, CU\textsubscript{I}, d\textsubscript{O})
2: \quad absAddr \leftarrow \text{AbsAddr}(CU\textsubscript{O}, 0)
3: \quad d\textsubscript{I} \leftarrow \text{D}(CU\textsubscript{I}, absAddr)
4: \quad partSize \leftarrow \text{PartSize}(CU\textsubscript{I}, absAddr)
5: \quad predMode \leftarrow \text{PredMode}(CU\textsubscript{I}, absAddr)
6: \textbf{if} d\textsubscript{O} < d\textsubscript{I} \textbf{then}
7: \quad i \leftarrow 0
8: \quad d\textsubscript{SO} \leftarrow d\textsubscript{O} + 1
9: \textbf{while} i < 4 \textbf{do}
10: \quad \text{subCU}\textsubscript{O} \leftarrow \text{Sub}(CU\textsubscript{O}, d\textsubscript{SO}, absAddr)
11: \quad \text{TRANSCODECU}(\text{subCU}\textsubscript{O}, CU\textsubscript{I}, d\textsubscript{SO})
12: \quad absAddr \leftarrow absAddr + \text{Count}(\text{subCU}\textsubscript{O})
13: \quad i \leftarrow i + 1
14: \textbf{end while}
15: \quad \textbf{if} (\text{predMode} = \text{Intra}) \textbf{then} \quad \triangleright \text{try intra mode for non-leaf CU}
16: \quad \text{intraData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Intra}, 2N \times 2N)
17: \quad \text{Copy}(CU\textsubscript{O}, \text{Intra}, 2N \times 2N, \text{intraData})
18: \quad CU\textsubscript{T1} \leftarrow \text{Encode}(CU\textsubscript{O})
19: \textbf{end if}
20: \textbf{if} (CU\textsubscript{O} \textbf{not in I-slice}) \textbf{then}
21: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter}, 2N \times 2N)
22: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter}, 2N \times 2N, \text{interData})
23: \quad CU\textsubscript{T2} \leftarrow \text{Encode}(CU\textsubscript{O})
24: \quad \triangleright N \times N \text{ PU split is only permitted for the smallest CU size}
25: \quad \textbf{if} (\text{partSize} \neq 2N \times 2N) \& \& (\text{partSize} \neq N \times N) \textbf{then}
26: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter}, \text{partSize})
27: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter}, \text{partSize}, \text{interData})
28: \quad CU\textsubscript{T3} \leftarrow \text{Encode}(CU\textsubscript{O})
29: \textbf{end if}
30: \textbf{end if}
31: \quad \triangleright \text{for none I-slices: at the root and one node before leaf try SKIP mode}
32: \textbf{if} (CU\textsubscript{O} \textbf{not in I-slice}) \& \& ((d\textsubscript{I} - 1 = d\textsubscript{O}) \& \& (d\textsubscript{I} = 0)) \textbf{then}
33: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter}, \text{skip, merge}, 2N \times 2N)
34: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter, skip, merge, 2N \times 2N, \text{interData}})
35: \quad CU\textsubscript{T4} \leftarrow \text{Encode}(CU\textsubscript{O})
36: \textbf{if} \textbf{else}
37: \textbf{if} (CU\textsubscript{O} \textbf{not in I-slice}) \textbf{then}
38: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter, skip, merge, 2N \times 2N})
39: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter, skip, merge, 2N \times 2N, \text{interData}})
40: \quad CU\textsubscript{T5} \leftarrow \text{Encode}(CU\textsubscript{O})
41: \textbf{end if}
42: \textbf{if} \textbf{else}
43: \textbf{if} (\text{predMode} = \text{Inter}) \textbf{then}
44: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter, partSize})
45: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter, partSize, \text{interData}})
46: \quad CU\textsubscript{T6} \leftarrow \text{Encode}(CU\textsubscript{O})
47: \quad \textbf{if} (\text{partSize} \neq 2N \times 2N) \triangleright \text{try no split PU for leaf}
48: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter, 2N \times 2N})
49: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter, 2N \times 2N, \text{interData}})
50: \quad CU\textsubscript{T7} \leftarrow \text{Encode}(CU\textsubscript{O})
51: \textbf{end if}
52: \textbf{else}
53: \quad \text{interData} \leftarrow \text{EstPredData}(CU\textsubscript{O}, \text{Inter, partSize})
54: \quad \text{Copy}(CU\textsubscript{O}, \text{Inter, partSize, \text{interData}})
55: \quad CU\textsubscript{T8} \leftarrow \text{Encode}(CU\textsubscript{O})
56: \textbf{end if}
57: \textbf{end if}
58: \textbf{end if}
59: \textbf{end function}
Appendix B

Spatial Resolution Reduction

Simple Cascaded Transcoding model for spatial-resolution reduction could be realized as illustrated in Figure B.1.

![Diagram of Simple Cascaded Transcoding model](image)

*Figure B.1: SCT model for spatial-resolution reduction transcoding*

As it is shown, the input bit-stream is completely decoded at SCT and then down-sampled before full encode. The down-sampling filter used here is the sine-window method of the extension provided with the reference AVC implementation.

There are subtle differences between efficient spatial-resolution reduction transcoding and bit rate reduction transcoding. The block information of input bit-stream can no longer directly be mapped to the output bit-stream. For example consider the CU prediction modes of Figure B.3. In this example, the input video has been downscaled by factor of 2 vertically and horizontally. The corresponding CU between input and output bit-stream is shown to the left and right of Figure B.3. The question is, which mode should be re-used to encode the output bit-stream?
Another example of the differences between spatial-resolution and bit rate reduction transcoding is the motion vector conversion illustrated in Figure B.3. The motion vectors could be mapped with 1 to 1 or 4 to 1, where each would produce different results.

Investigating efficient methods for spatial-resolution transcoding is out of scope of this work. However, the spatial-resolution reduction model of SCT is investigated. Reference implementation of HEVC (HM-8.2) with the addition of resolution reduction model are used for implementing spatial-resolution reduction SCT.
Figure B.4 and Figure B.5 show the SCT performance for spatial-resolution reduction with LDM and RAM configurations, subsequently. It is observed that SCT transcoding performance has a declining trend in every case. Transcoding with LDM configuration requires about 2% greater overhead compared to RAM configuration. Transcoding sequence class C has the lowest overhead since the spatial resolution is reduced to $208 \times 120$. 
Appendix B. Spatial Resolution Reduction

Figure B.4: Class average bit rate ratio and overhead transcoding performance with LDM configuration
Appendix B. Spatial Resolution Reduction

Figure B.5: Class average bit rate ratio and overhead transcoding performance with RAM configuration


ISO/IEC-11172. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s. 1993.


I Kim, Junghye Min, W Han, and J Park. Block partitioning structure in the hevc standard. 2012.


J. Lainema, F. Bossen, W. Han, J. Min, and K. Ugur. Intra coding of the hevc standard. 2012.


