FAST SCREEN CONTENT CODING AND TWO TIER 360 DEGREE VIDEO STREAMING

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Fanyi Duanmu

May 2018

FAST SCREEN CONTENT CODING AND TWO TIER 360 DEGREE VIDEO STREAMING

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Fanyi Duanmu

May 2018

Approved:

____

Department Chair Signature

____

Date

University ID: N13364196

Net ID: fd471

Approved by the Guidance Committee:

Major: Electrical Engineering

Yao Wang
Professor of
Electrical and Computer Engineering

Date

Yong Liu
Associate Professor of
Electrical and Computer Engineering

Date

Ivan Selesnick
Professor of
Electrical and Computer Engineering

Date

Edward Wong
Associate Professor of
Computer Science and Engineering

Date

Microfilm or other copies of this dissertation are obtainable from

UMI Dissertation Publishing
ProQuest CSA
789 E. Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Vita

Fanyi Duanmu was born in China on May 26, 1986. He received the B.S. and M.S. degree both in Electrical Engineering from Beijing Institute of Technology, Beijing, China and Polytechnic Institute of New York University, Brooklyn, NY, in 2009 and 2011, respectively. Since January 2013, he has been a PhD student at Electrical and Computer Engineering Department at New York University, Brooklyn, NY under the supervision of Professor Yao Wang.

In summer 2014 and summer 2015, he worked as a video codec intern at Huawei Technologies, Santa Clara, CA, on the standardization of HEVC Screen Content Coding (SCC) extension. In summer 2017, he worked as a video codec intern at InterDigital Communications, San Diego, CA, on the standardization of Next Generation Video Coding on SDR and 360-degree videos. During his doctorate study, his research area includes video coding and transcoding, screen content compression and virtual reality streaming.

Acknowledgment

Firstly, I would like to present my appreciation and gratitude to my PhD advisor, Professor Yao Wang, for her invaluable guidance, insight and support through my entire PhD program. Her knowledge, research passion and diligence not only inspires me during my PhD study but also my future career. I would like to thank my guidance committee members, Professor Yong Liu, Professor Ivan Selesnick and Prof. Edward Wong for their precious time in reviewing this work and their constructive suggestions to improve this work from different aspects.

Secondly, I would like to thank my labmates: Dr. Eymen Kurdoglu, Dr. Jenwei Kuo, Mr. Shervin Minaee, Mr. Andy Chiang, Dr. Yilin Song, Mr. Yuan Wang, Ms. Chenge Li, Mr. Ran Wang for the inspirational discussions and collaborations during my PhD program. I also feel lucky and grateful to collaborate with Dr. Amir Hosseini, Mr. Liyang Sun, Mr. Yixiang Mao and Mr. Shuai Liu in my previous projects, who gave me valuable inputs and supports along the way.

Thirdly, I would like to thank my industrial collaborators during my previous internships and paper co-publications. Particularly, I would like to thank Dr. Haoping Yu, Mr. Wei Wang from Huawei, Dr. Yuwen He, Dr. Yan Ye, Dr. Xiaoyu Xiu, Dr. Philippe Hanhart and Mr. Yan Zhang from InterDigital, Dr. Zhan Ma from Nanjing University, Dr. Xiaozhong Xu from Tencent, Dr. Meng Xu from Ubilinx. Their inspirations and suggestions significantly benefit my research and skillset development from an industrial perspective.

Additionally, I also wish to thank everybody not mentioned here, but who contributed in one way or another towards the success of this thesis.

The last but not the least, I would like to thank my parents and my wife, for their unconditional love, support and encouragement throughout my life.

To my parents Qingduo Duanmu, Yihua Wang and my wife Chuhan Ran

ABSTRACT

FAST SCREEN CONTENT CODING AND TWO TIER 360 DEGREE VIDEO STREAMING

by

Fanyi Duanmu

Advisor: Prof. Yao Wang, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Electrical Engineering)

May 2018

In this thesis, efficient video delivery solutions are designed to address screen content coding (SCC) and 360-degree video streaming. Towards this goal, multiple sub-problems and applications are addressed.

The first problem is to accelerate screen content encoding and transcoding, while simultaneously preserving the coding efficiency. For encoding acceleration, machine learning based fast algorithms are proposed, using High Efficiency Video Coding (HEVC) coding unit (CU) features for fast block mode and partition decisions. For transcoding acceleration, additional side information from the decoded bitstream is further incorporated for fast block mode and partition decisions, using machine learning and mode mapping techniques for forward HEVC to HEVC-SCC transcoding and backward HEVC-SCC to HEVC transcoding, respectively. Significant encoding and transcoding complexity reductions have been achieved with negligible losses in coding efficiency.

The second problem is to efficiently stream on-demand 360-degree videos or virtual reality (VR) contents. A novel two-tier streaming framework is proposed to simultaneously address the dynamics in bandwidth variation and user viewing direction changes. The proposed solution formulates the 360-degree video streaming as a dynamic video segment scheduling problem and utilizes prioritized buffer control solution to effectively determine the pre-fetching action and bitrate, based on the current estimated bandwidth, buffer status and view prediction accuracy. The system achieves a significant performance improvement in the delivered video quality compared with benchmark solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation of Fast Screen Content Coding

Screen content (SC) videos have become popular in recent years due to the development and technical advances in mobile technologies and cloud applications, such as shared screen collaboration, remote desktop interfacing, cloud gaming, wireless display, animation streaming, online education, etc. These emerging applications create an urgent need for better compression technologies and low-latency delivery solutions for screen content videos, especially to support the incoming HD requirement of 4K or even higher resolutions.

To explore the unique signal characteristics of screen content and develop efficient SC compression solutions, the ISO/IEC Moving Picture Expert Group and the ITU-T Video Coding Experts Group, also referred as Joint Collaborative Team on Video Coding (JCT-VC), has launched the standardization of SCC extension [60] on top of the latest video standard - High Efficiency Video Coding (HEVC) [61] since January 2014 and this extension is concluded in 2016 with significant research efforts involved from both academia and industry.

The official JCTVC Screen Content Model software (SCM) [28] is reported to provide over 50% BD-Rate saving over HEVC Range Extension (RExt) [60] for computer-generated contents. Four major coding tools were introduced and adopted during the standardization, known as "Intra Block Copy" (IBC) [49][5], "Palette Coding Mode" (PLT) [22], "Adaptive Color Transform" (ACT) [80] and "Adaptive Motion Compensation Precision" (AMCP) [40] [37], respectively.

Recognizing the market need and SCC efficiency, many industrial companies are currently following this new extension and most likely may include these new coding techniques into their future products and services. However, the intrinsic flexibility of HEVC partitioning scheme and the newly-introduced screen content coding tools impose a significant computational burden on the encoder, primarily during the seeking of optimal combinations of coding unit (CU) partitions and modes, as summarized into the following two fundamental problems:

Problem 1 - Mode Decision: given the current CU, which mode (among 35 intra modes, inter mode, palette mode and intra block copy mode) should be chosen to minimize Rate-Distortion (RD) cost?

Problem 2 - Partition Decision: given the current CU, should it be further partitioned into smaller sub-CUs for a reduced RD-cost?

In HEVC model software (HM) [47] and SCM [28], an exhaustive search method is employed to solve these two problems by examining all possible modes for the current CU and comparing the RD-cost of the current CU using the best mode against the sum of RD-costs from its sub-CUs, each using the best mode and partition recursively. However, for time-critical applications, such exhaustive mode and partition decision strategy is not practical. Therefore, it is important to develop fast algorithms to accelerate the CU partition and mode decisions while simultaneously preserving the coding efficiency.

On the other hand, how to efficiently bridge the existing HEVC bitstream and the incoming SCC bitstream through software-based video transcoding (VTC) solution becomes desirable and useful, especially during the phase when baseline HEVC bitstream and the novel SCC bitstream coexist, as summarized into the following two scenarios:

Scenario 1 - Forward Transcoding: given the HEVC bitstream, how to convert it into more efficient SCC bitstream to accommodate bandwidth-critical applications?

Scenario 2 - Backward Transcoding: given the novel SCC bitstream, how to convert it back to HEVC bitstream to provide backward compatibility over the legacy HEVC devices?

VTC is a useful and mature technology to realize video adaptation. It converts the incoming bitstream from one version to another. During the conversion, many properties from the source video may change, such as video format, video bitrate, frame rate, spatial resolution and coding standards used, etc. In the literature, the conversion within the same standard (e.g., the spatial re-scaling in H.264) is referred as the "homogeneous transcoding", while the conversion between different standards (e.g., between H.264/AVC and HEVC) is referred as the "heterogeneous transcoding". Beyond that, even additional information could be inserted during transcoding, such as video watermarking, error resilience, etc. In practice, a transcoding server can be set up to periodically examine the client's constraints (e.g., bandwidth, power limit, display resolution, etc.) and accordingly "tailor" the suitable bitstream to the clients.

Though it is absolutely possible to use the "trivial" solution, which firstly decodes the source bitstream and then completely re-encodes into the target bitstream, however, such approach proves inefficient from the complexity point of view. A reasonable solution should maximally utilize the decoded side information from the source bitstream to facilitate the re-encoding such that both the coding efficiency is well-preserved and the re-encoding speed is significantly improved.

1.2 Previous Work on Video Codec Acceleration

There are a huge amount of prior works for video encoding accelerations. These prior works can be categorized into the following categories:

Category 1 - Mode Reduction. A gradient-based fast mode decision framework was proposed in [83], which is based on CU directional histogram analysis to reduce the number of Intra candidates before mode selection. A 20% complexity reduction over HM-4.0 is reported under this scheme with negligible coding performance loss for Intra-frame coding. Another fast intra mode decision algorithm was proposed in [27], which exploits the directional information of neighboring blocks to reduce the Intra candidates of the current CU. Up to 28% complexity reduction is reported over HM-1.0 with insignificant coding performance loss for Intra-frame coding. The HM test model software adopted [54] to reduce Intra-mode candidates. Firstly, a rough mode decision (RMD) is performed using Hadamard cost to choose fewer candidates out of 35. Then the extra most probable modes (MPMs) derived from spatial neighbors will be added to the previous candidate set if they are not yet included.

Category 2 - Cost Replacement. An entropy-based fast Coding Tree Unit partition algorithm was proposed in [82], which replaces heavy Rate-Distortion optimization (RDO) calculation by Shannon entropy calculation. A 60% complexity reduction is reported using this algorithm with a BD-rate loss of 3.8% for Intra-frame coding. In [54], Hadamard cost is used for Intra RMD without fully formulating the RD cost. This approach significantly reduces the Intra-frame coding complexity.

Category 3: Fast Partition Termination. A fast CU splitting decision scheme was proposed in [43], using weighted SVM decision for early CU partition termination for both Intra-frame and Inter-frame coding. A complexity reduction of >40% is reported over HM-6.0. Another fast termination algorithm was proposed in [26], using texture complexity of neighboring blocks to eliminate unnecessary partition of the current CU. A 23% encoder speed-up on average is reported over HM-9.0 for Intra-frame coding. Another work by Zhang and Ma [77] includes a set of early termination criteria for HEVC intra coding based on experimental observation and simulation results. To determine the splitting decision, encoder will do a 1-level RD evaluation by comparing current CU Hadamard cost with the combined Hardmard cost of 4 sub-CUs without further splitting. Zhang and Ma further proposed an improved 3-step fast HEVC Intra coding algorithm in [78]. At the RMD step, a 2:1 down-sampled Hadamard transform is used to approximate the encoding cost followed by a progressive mode refinement and early termination verification. It reports a 38% average complexity reduction over HM-6.0 with 2.9% BD-rate loss for Intra-frame coding.

Category 4: Fast Search Algorithm. A number of fast motion estimation (ME) algorithms have been proposed in the past years, including multi-step search [39][55], diamond search [88], cross-diamond search [6], hexagon search [87], etc. These algorithms follow different search patterns to reduce the number of search points for Inter-frame coding. In HEVC Test Model software (HM), Enhanced Predictive Zonal Search (EPZS) [65] is incorporated to reduce encoder complexity, in which prediction is continuously refined within a local search using a small diamond or square pattern and the updated best vector becomes the new search center.

These prior works were mainly proposed for natural video coding without considering the unique signal characteristics of screen contents, which typically contain limited distinct colors, sharper edges, repetitive graphical patterns, less complicated textures and irregular temporal motions. Besides, these works did not take into account the newly-introduced SCC modes (e.g., IBC or PLT). Therefore, these conventional fast algorithms cannot be directly applied onto screen content coding.

There are a few recent works on SC fast encoding. For instance, Li and Xu presented a fast algorithm for AMCP [36] to quickly determine the frame type (namely, SC image or Natural image), based on the percentage analysis of smooth blocks, collocated blocks, matched blocks and other blocks (i.e., the blocks that do not belong to the previous three categories). For similar encoder complexity, a 7.7% average BD-Rate improvement is reported over SCM-2.0 for Inter-frame coding. Kwon and Budagavi proposed a fast IBC search algorithm [33], by imposing restrictions on IBC search range, search directions and motion compensation precision. There are also several works on hash-based fast search algorithms for IBC mode and Inter mode coding [57] [35] [38]. Tsang, Chan and Siu proposed a Simple Intra Prediction (SIP) scheme [66] to bypass RMD and RDO processing for smooth SCC regions, whose CU boundary samples are exactly the same. Up to 26.7% peak complexity reduction is reported over SCM-2.0 with marginal video quality degradation for Intra-frame coding. Lee et al. proposed a fast transform-skip mode decision scheme for SCC [34], by enforcing IBC block with zero coded block flag (CBF) to be encoded with transform-skip mode. A 3% encoding speed-up is reported over SCM-2.0. Zhang, Guo and Bai proposed a fast Intra partition algorithm [81] for SCC, using CU entropy and CU coding bits to determine CU partition decision for Intra-frame coding. In a recent work [79] proposed by Zhang and Ma, temporal CU depth correlations are exploited and adaptive block search step size is incorporated. Average complexity reductions of 39% and 35% are reported for lossy and lossless Intra-frame coding, respectively.

Beyond these fast encoding solutions, video transcoding fast algorithms benefit greatly from re-utilizing the decoded video side information, including block partition decision, coding mode distribution, residuals, transform coefficients, etc. In [67], a high-level video transcoding overview is presented from a system architectural perspective and introduces the spatial and temporal resolution reduction, DCT-domain down-conversion. When HEVC is finalized in 2013, a huge amount of VTC studies were redirected to "H.264-HEVC" conversion area. For instance, Peixoto, et al. proposed several machine learning and statistics based schemes (e.g., [51] [53] [52]) to improve HEVC re-encoding speed. In their papers, H.264/AVC Macroblocks (MBs) are mapped into HEVC coding units (CU) based on the motion vectors (MVs) distribution through online or offline training. Incorporated with statistics-based fast termination criteria, the proposed schemes could introduce a >3x encoder speedup with a 4% BD-Rate loss compared with trivial transcoder. Diaz-Honrubia, et al. also proposed a series of fast VTC schemes (e.g., [8] [25]) to exploit H.264/AVC decoded side information for HEVC CU partition decision based on a Naive-Bayes (NB) classifier, specifically for CUs with size 32x32 and 64x64, whereas for the smaller CUs, the proposed transcoder simply mimics the

H.264/AVC coding behaviors. A quantitative speed-up of 2.5x is reported with only 5% BD-Rate penalty. In [84], a HEVC fast transcoder is proposed based on block homogeneity prediction. Residuals and MV consistencies are populated to represent the homogeneity of the target region and decide the CU partition. Another similar work [85] proposed by Zheng uses residual mean absolute deviations (MAD) and sum of absolute residual (SAR) as the homogeneity indicator to early terminate CU partition. A 57% complexity reduction is achieved with only 2.2% BD-Rate loss. In [48], a mode merging and mapping solution is presented using H.264/AVC motion vector (MV) variance and mode conditional probabilities to predict HEVC merge decisions. A 50% complexity reduction is reported with negligible BD-Rate loss.

1.3 Contributions to Screen Content Coding

Even though there have been substantial prior research efforts in video coding and transcoding acceleration, to my best knowledge, there are very limited prior works on fast SC encoding and there is no prior work on SC transcoding. In this thesis, fast compression frameworks and algorithms are designed and implemented to accelerate SC encoding and transcoding. The major contributions in this work can be summarized as follows.

Firstly, extensive statistical investigations and complexity studies are conducted, to analyze HEVC and SCC encoding decisions over different screen contents. Such information enables us to better understand the relationships between screen content characteristics and the codec behaviors (e.g., mode preferences) to facilitate our algorithm development.

Secondly, a machine learning based fast screen content encoding framework is proposed to accelerate CU mode and partition decisions. This framework includes three major classifiers designed to either reduce the mode candidates to be examined or to make fast partition decisions. Our experimental results demonstrate that the proposed algorithm can achieve a significant complexity reduction while simultaneously preserving the coding efficiency.

Thirdly, a machine learning based HEVC-SCC forward transcoding framework is proposed for accelerating CU mode and partition decisions, using CU low level features and the decoded residual signal sparsity to accurately classify the incoming CUs into either “natural video” or “screen content” and directly trigger the corresponding mode candidate(s). To our best knowledge, this is the first work in this area and it achieves a significant transcoding speedup. The framework is designed to facilitate some bandwidth-critical screen content applications, such as remote desktop interfacing, wireless display, etc.

Fourthly, a fast SCC-HEVC backward transcoding framework based on statistical mode mapping technique is proposed. Fully exploiting the side information from the decoded SCC bitstream, fast mode and partition decisions are designed to accurately determine the mode mapping relationship between the novel SCC modes and the conventional HEVC modes. To my best knowledge, this is the first work in this area and it achieves remarkable transcoding complexity reduction with negligible coding efficiency loss. The framework supports the backward bitstream-compatibility over the legacy HEVC devices and can be further extended into the single-input-multiple-out (SIMO) architecture to support adaptive screen content video streaming over the edge cloud.

1.4 Motivation of Two-tier 360 Video Streaming

In recent years, Virtual Reality (VR) and Augmented Reality (AR) technologies have become popular and rapidly commercialized. A variety of applications have been developed continuously to meet the market demands and consumer expectations, such as immersive cinema, gaming, education and training, tele-presence, social media, and healthcare, etc. The main differentiator is to provide the end consumers with omni-directional viewing flexibility and immersive video experience. Some preliminary 360 video services are already made available on several major video platforms, such as YouTube, Facebook, etc. However, the current delivery solutions still treat 360 videos as regular 2D videos and stream the entire 360 degree view scope to the end users regardless of their view directions. Compared with the traditional video streaming, 360 degree video streaming confronts unique new challenges. Firstly, to deliver an immersive VR experience, 360 video has much higher bandwidth requirement. For example, a premium quality 360 video with 60 frames per second, 4K resolution can consume bandwidth up to multiple Gigabits-per-second (Gbps). Secondly, user view direction dynamics is a new dimension of freedom in 360 degree video streaming. A user may freely change or navigate her viewing direction during the video playback and expect to see the scene in the new viewing direction immediately. Since only a small portion of the entire 360 degree video is watched at any time, therefore, streaming the entire 360 degree video representation is unnecessary and bandwidth-consuming. On the other hand, only streaming the video in the predicted viewing direction will introduce streaming discontinuity whenever the view prediction is inaccurate. How to stream 360-degree video and robustly adapt to network variation and the users' view direction dynamics is a challenging problem and a critical requirement for the wide adoption of VR/AR.

1.5 Previous Work on 360 Video Streaming

In recent years, a few 360-degree video streaming solutions have been proposed, as summarized into the following categories.

Category 1: 360 Video Source Representation. A typical 360 degree video compression and delivery workflow is illustrated in Figure 1.1. Firstly, videos captured from multiple cameras are stitched together into a native projection format, e.g., equirectangular projection (ERP) format. The native projection format can be converted into another projection format, e.g., cubemap (CMP), and frame-packed before being fed into modern video codecs, such as H.264/AVC, HEVC, VP9, etc. In such a framework, the selection of the intermediate video projection format is important and would potentially improve 360 video coding efficiency. Facebook proposed the cubemap [31] and pyramid [32] projection and encoding solutions in 2016, to specifically address the on-demand 360 video streaming application, with 25% and 80% compression improvements reported, respectively. The Joint Video Exploration Team (JVET) also proposed several projection solutions, including Icosahedral projection (ISP) [86], Segmented Sphere Projection (SSP) [76], Truncated Square Pyramid Projection (TSP) [9], Octahedron Projection (OHP) [41], Hybrid Cubemap (HCP) [23] [11] [24], etc.

Category 2: Source Bit Allocation. Different view regions have different perceptual quality implications, consequently deserve different numbers of coding bits. In [4], a region-adaptive smoothing scheme is proposed to reduce the bitrate spent within the polar regions of equi-rectangular 360 videos through Gaussian

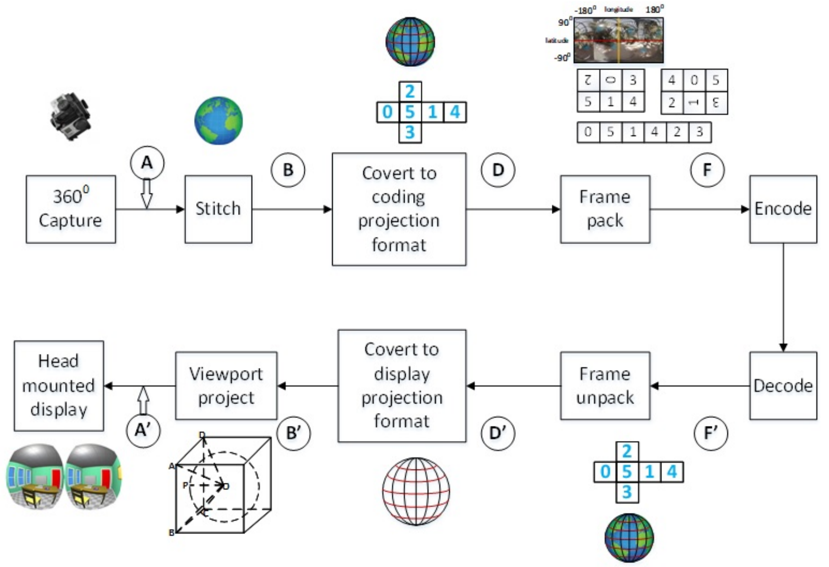


Figure 1.1: Workflow of 360 Video Compression and Delivery System

filtering. A 20% bitrate reduction is reported with unnoticeable perceptual quality degradation.

Category 3: View-oriented Streaming. In [70], a few tile-based encoding and streaming solutions are proposed, including scalable coding scheme and simulcast coding scheme. Video tiles that cover the whole 360 scene are coded in multiple rates. Depending on the Field of View (FOV), tiles within or close to the predicted FOV are fetched with higher bitrate while tiles far away from the predicted FOV are fetched with lower bitrate. In [56], a view prediction based framework is proposed by only fetching the video portions desirable to the end user to reduce the bandwidth consumption. A dynamic video chunk adaptation scheme is implemented to adjust the tile coverage based on the view prediction accuracy. An estimated 80% maximum rate reduction is reported without considering the coding efficiency loss due to video tiling and bandwidth variations.

1.6 Contribution to 360 Video Streaming

Inspired by the prior works, in this thesis, a two-tier 360-degree video streaming framework with prioritized buffer control is proposed. In this framework, 360 videos are encoded into two tiers and adaptively streamed to better accommodate the dynamics in both network throughput and user viewing direction. This framework aims at maximally utilizing the available bandwidth to enhance the end viewers' quality of experience (QoE). Unlike the previous solutions, the proposed framework is source-representation-independent. Any aforementioned projection solution (e.g., Cube-map, Icosahedral, etc.) and source bit-allocation approach (e.g., region-adaptive smoothing) can be easily incorporated into our framework for additional performance improvement. Through extensive simulations driven by real network bandwidth traces and user view traces, the proposed framework demonstrates significant performance improvement against the benchmark 360-degree video streaming solutions.

1.7 Outline of the Thesis

This thesis consists of two major research topics.

Topic 1 on fast screen content coding spans from Chapter 2 to Chapter 6 and is organized as follows.

Chapter 2 briefly reviews screen content model (SCM) quad-tree coding structure, new compression tools and discusses about the technical challenges.

Chapter 3 provides statistical studies and coding behavior analyses and complexity distribution of SCC and HEVC encoders over typical screen contents.

Chapter 4 presents the proposed fast screen content encoding framework based on machine learning techniques, including feature selection, classifier selection, parameter training methodologies and the classifiers design. Experimental results are presented to demonstrate the achieved encoder acceleration.

Chapter 5 presents the proposed HEVC-SCC forward transcoding framework based on machine learning techniques, using neural network to design classifiers to make fast coding decisions. Experimental results are presented to demonstrate the achieved transcoding speedup.

Chapter 6 presents the proposed SCC-HEVC backward transcoding framework based on statistical mode mapping techniques. The proposed fast algorithms are integrated into single-input-multiple-output (SIMO) framework for adaptive screen content streaming applications over the edge cloud. Experimental results are presented to demonstrate the achieved transcoding speedup.

Topic 2 on two-tier 360-degree video streaming is introduced in Chapter 7. A prioritized buffer control algorithm is proposed to adaptively determine the video tier and video rate to be requested. Experimental results demonstrate the potential performance gain compared with the benchmark solutions.

Finally, Chapter 8 concludes this thesis with some future work summarized.

Chapter 2

Screen Content Model (SCM) - Brief Review

SCM [28] is the JCTVC official test model software for SCC extension development. This software is developed upon HEVC-RExt [60] codebase and supports YUV-4:4:4, YUV-4:2:0 and RGB-4:4:4 sampling formats. Beyond HEVC, new coding tools are introduced to improve SC coding efficiency. Within the scope of this thesis, we are working on SCM-4.0 software and the proposed algorithms can be easily generalized onto other SCM releases.

2.1 SCM Mode and Partition Decision

SCM inherits the same flexible quadtree block partitioning scheme from HEVC, which enables the use of CUs, Prediction Units (PU) and Transform Units (TU) to adapt to diverse picture contents. CU is the square basic unit for mode decision. The Coding Tree Unit (CTU) is the largest CU, of 64x64 pixels by default.

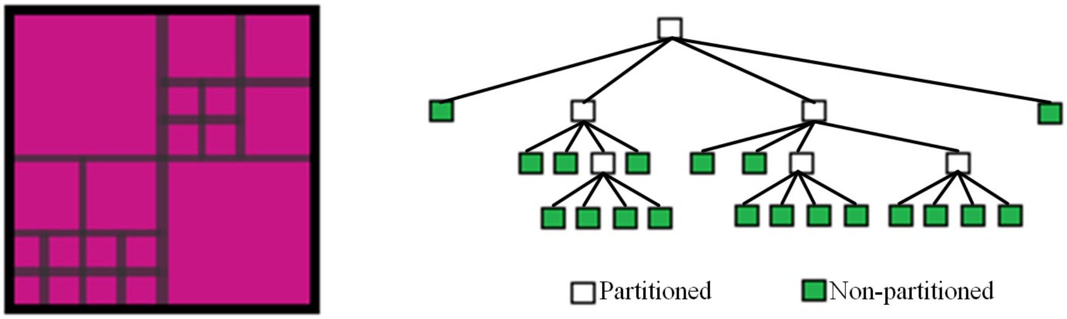


Figure 2.1: SCM Hierachitical Quadtree Partitioning Structure

At encoder, pictures are divided into non-overlapping CTUs and each CTU can be recursively divided into four equal-sized smaller CUs, until the maximum hierarchical depth is reached, as shown in Figure 2.1. At each CU-level, to determine the optimal encoding parameters (e.g., partition decision, mode decision, tool usage, etc.), an exhaustive search method is employed by comparing the RD costs using different coding modes and comparing the minimum RD cost at the current CU level against the sum of RD costs of its sub-CUs (each using the best mode and partition recursively). For the rest of this thesis, we will use “CU64” (i.e., CTU), “CU32”, “CU16” and “CU8” to denote the CUs at different hierarchical depths.

2.2 SCM New Coding Tools beyond HEVC

Beyond HEVC, SCM adopted four major coding tools to compress screen contents more efficiently, known as “Intra Block Copy”, “Palette Mode”, “Adaptive Color Transform”, “Adaptive Motion Compensation Precision”, respectively.

Intra Block Copy (IBC) [49] [5] is an Intra-frame version of the motion estimation and motion compensation scheme. To compress the current CU, the encoder will look back into the previously coded areas (either in restricted area or globally) in the same frame and look for the best matching block. If chosen, a “Block Vector” (BV) will be signaled, either explicitly or implicitly, to indicate the relative spatial offset between the best matching block and the current PU location.

Palette Mode [22] encodes the current CU as a combination of a color table and the corresponding index map [73]. Color table stores representative color “triplets” of RGB or YUV. Then the original pixel block is translated into a corresponding index map indicating which color entry in the color table is used for each pixel location.

Adaptive Color Transform [80] converts residual signal from original RGB or YUV color space onto YCoCg color space. It de-correlates the color components, reduces the residual signal energy and therefore improves the coding performance.

Adaptive Motion Compensation Precision [40] [37] analyzes Inter-frame characteristics and categorizes the current frame as either a natural video frame (NVF) or screen content frame (SCF). For SCF, integer-pixel precision is applied for motion estimation. For NVF, sub-pixel precision is applied.

2.3 SCM-4.0 IBC and Inter Unification

Beyond the previous SCM releases, IBC mode and Inter mode are unified in SCM-4.0. Namely, IBC mode is treated as a special Inter mode with the reference frame restricted to the current frame and the reference area restricted to the previously-encoded area in the current frame. This unification improves SCM encoding from the following perspectives.

1. Workflow harmonization for hardware re-utilization.

2. Inter-frame CU encoding flexibility, i.e., Inter-frame CUs can have its PUs copied from both the current frame and the temporal reference frame simultaneously.

3. Inter-frame coding schemes generalization over IBC blocks, such as "Advanced Motion Vector Prediction" (AMVP) [42], "Merge/Skip mode" [29], etc. Henceforward, we use "IBC-Merge", "IBC-Skip" and "IBC-Inter" to address the difference against conventional HEVC terminology.

4. Inter-frame fast algorithms can be transfered onto Intra-frames. For instance, when IBC finds a "Skip"-coded candidate, the encoder can safely terminate all the following rate-distortion optimizations (RDO).

2.4 SCM Fast Coding Decision Challenges

Different from the conventional HEVC modes, new SCC modes are highly dependent on the repetitive graphical patterns and image colors that previously appeared. This "historical dependency" makes CU partition and mode decision problems much more complicated and challenging. For example, in IBC blocks, depending on whether similar pattern appeared previously, encoding costs of the same CU pattern but at different locations may vary significantly. Similarly, for PLT coding mode, two color tables are maintained. One is used for the current CU coding and the other table (also referred as "palette predictor") is used as a dynamic lookup table caching the historical colors previously used. Depending on whether similar colors appeared before and how frequent these colors are, the PLT coding costs of the same CU pattern but at different locations may vary significantly. Furthermore, the newly-introduced SCC modes enable "inhomogeneous"

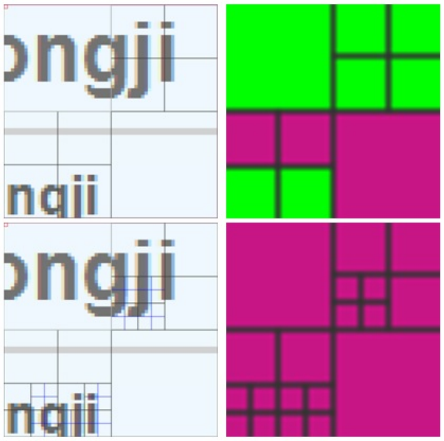


Figure 2.2: SC Block Coding Decision Comparison between SCC and HEVC. Green blocks are PLT-coded. Red blocks are Intra-coded.

blocks to be encoded into a larger block without splitting. As shown in Figure 2.2, 16×16 textual CUs in the top row are encoded directly using PLT mode (marked in green) without splitting into smaller 8×8 Intra CUs (marked in red) in the bottom row.

To conclude, due to the unique signal characteristics of screen contents and the designs of PLT and IBC algorithms, traditional HEVC fast mode and partition algorithms cannot be directly applied to SCC. How to efficiently and accurately determine the SCC decisions for fast encoding and how to accurately correlate modes and partitions between HEVC and SCC for fast transcoding are challenging problems, even for human analytical judgment.

Chapter 3

Screen Content Coding Statistical Study

In this chapter, we conduct extensive statistical studies on SC mode and partition distribution, to better understand the unique screen content signal characteristics and HEVC and SCC encoding complexity distribution and mode selection behaviors.

3.1 Dataset Preparation

Our statistical studies are based on HM-16.4 and SCM-4.0 encoding data over the standard SC sequences jointly selected by the experts from JCTVC community. These sequences cover the most typical screen contents, such as “Desktop”, “Console”, “Map”, “SlideShow”, “WebBrowsing”, etc., as shown in Figure 3.1. The corresponding sequences and coding parameter configurations are described in JCTVC SCC Common Testing Conditions (CTC) document [75].

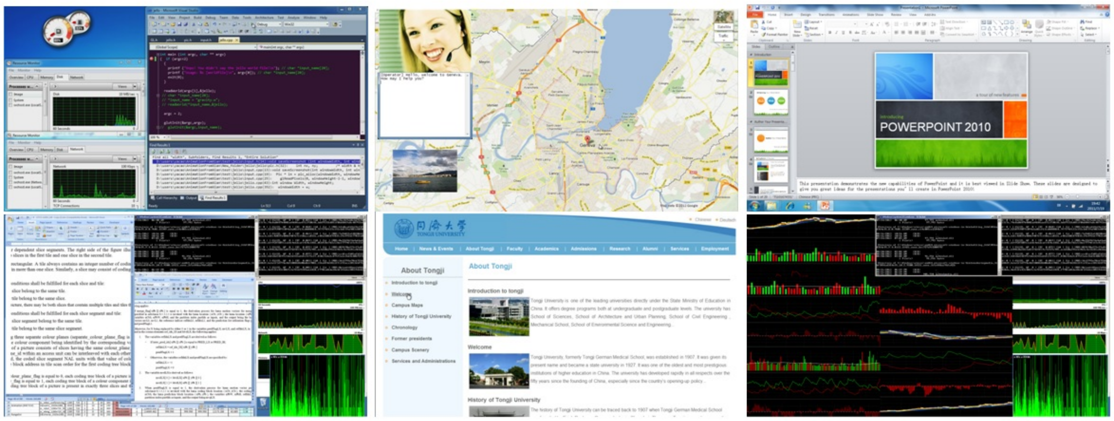


Figure 3.1: Sample Frames from SCC Standard Sequences

For Intra-frame coding study, to avoid duplicate CU samples in the training set, we extract 10 sample frames from each sequence whose pixel-wise temporal difference against the previous frames are the largest over luma (i.e., Y-Component). The selected frames (in Table 3.1) are coded using All-Intra (AI) configurations under QP=22, QP=27, QP=32 and QP=37, respectively. The mode selection and partition decision labels derived from SCM-4.0 and HM-16.4 encoder are used as the ground-truth data for the following statistical studies, encoding module machine learning training and transcoding heuristic derivation.

For Inter-frame coding statistical study, the first 10 frames from each sequence are encoded using Low-Delay with P-frame (LDP) configuration. Only data from the Inter-frames are collected. We assume that the mode distribution can be generalized to the new SC videos. Since the mode and partition decisions are quantization-dependent, the simulation data for QP=22 and QP=37 cases are provided for comparison purpose.

Table 3.1: Intra-frame Coding Sample Frame Selection

Sequence	Sample Frame Index
Map	0, 484, 73, 151, 112, 100, 61, 65, 102, 63
WebBrowsing	0, 286, 100, 67, 254, 8, 34, 144, 120, 188
Programming	0, 73, 568, 598, 43, 70, 45, 38, 244, 48
SlideShow	0, 3, 27, 474, 170, 169, 171, 167, 168, 266
FlyingGraphics	0, 278, 144, 111, 141, 84, 100, 73, 171, 240
Desktop	0, 463, 501, 422, 480, 479, 508, 128, 30, 31
Console	0, 418, 130, 599, 439, 137, 509, 465, 419, 429
Basketball	482, 540, 538, 376, 534, 504, 409, 479, 493, 439
MissionControlClip2	209, 255, 211, 154, 158, 206, 259, 161, 260, 151
MissionControlClip3	0, 478, 146, 481, 479, 475, 362, 483, 474, 365

3.2 Intra-Frame Mode and Partition Statistics

The Intra-frame partition distribution using HM-16.4 and SCM-4.0 are summarized in Table 3.2. The statistics demonstrate that the total non-partitioned block percentage of SCM-4.0 is greater than the percentage using HM-16.4. It coincides with the reasoning as illustrated in Figure 2.2, that the inhomogeneous SC block can be efficiently coded using IBC or PLT mode without further splitting. Besides, larger CUs are more likely to be further partitioned. Till CU16 level, the partition decision using SCC becomes almost unpredictable.

Table 3.2: Intra-frame CU Partition Statistics

CU Width	QP	HM Split	HM Unsplit	SCM Split	SCM Unsplit
64	22	91.27%	8.73%	90.37%	9.63%
64	37	86.14%	13.86%	84.80%	15.20%
32	22	79.88%	20.12%	82.75%	17.25%
32	37	73.24%	26.76%	78.50%	21.50%
16	22	69.77%	30.23%	49.06%	50.94%
16	37	65.14%	34.86%	44.84%	55.06%

The Intra-frame mode distribution statistics using SCM-4.0 is provided in Table 3.3. Statistically, SCM-4.0 uses a large proportion of SCC modes to compress screen contents. At CU64 level, PLT mode and IBC-Inter mode are disabled by default for complexity consideration. The IBC mode utilization increases as CU size decreases, since that smaller blocks have higher likelihoods to find perfect or good matches. PLT mode does not have a significant utilization percentage at all CU levels but consumes a large bitrate consumption percentage, as studied in [72].

Table 3.3: Intra-frame CU Mode Statistics

CU Width	QP	IBC Merge	IBC Skip	IBC Inter	Intra	PLT
64	22	0.46%	46.01%	N/A	53.53%	N/A
64	37	0.43%	29.44%	N/A	70.13%	N/A
32	22	0.49%	37.49%	4.29	29.01%	28.74%
32	37	0.91%	29.70%	3.01	33.31%	33.06%
16	22	1.39%	34.75%	10.52%	26.74%	26.61%
16	37	1.40%	34.37%	10.70%	27.04%	26.49%
8	22	5.41%	31.60%	18.03%	22.63%	22.33%
8	37	3.96%	32.49%	19.88%	22.02%	21.65%

The Intra sub-mode selection distribution using HM-16.4 is provided in Table 3.4. Different from natural videos, this distribution implies that screen contents are dominated by purely horizontal and purely vertical graphical patterns. The four major Intra sub-modes, i.e., Intra-Planar, Intra-DC, Intra-Horizontal and Intra-Vertical consume a large percentage of Intra sub-mode usage (i.e., from >65% up to >90%).

Additionally, visual analysis and CU coding bitrate is evaluated. If SC block pattern is simple (e.g., a bicolor CU containing two horizontal stripes, etc.), the RD decision becomes unpredictable among PLT, IBC and Intra modes. However, the CU bit-consumptions using all these modes are relatively close. Smoothly-varying CUs (mostly from natural video region) will be coded primarily using DC or Planar without further partitioning. For SC blocks, if a child sub-CU can find a perfect match, SCM encoder will mostly choose to partition. If all the sub-CUs in

Table 3.4: Intra-Submode Statistics

CU Width	QP	Planar	DC	Horizontal	Vertical	Others
64	22	7.79%	7.04%	27.39%	47.99%	9.80%
64	37	14.40%	9.02%	27.85%	32.59%	16.14%
32	22	5.83%	10.17%	45.86%	25.22%	12.92%
32	37	9.49%	9.81%	43.38%	21.74%	15.59%
16	22	8.37%	5.92%	43.52%	21.30%	20.89%
16	37	10.71%	6.37%	37.91%	21.07%	23.95%
8	22	8.05%	5.39%	30.73%	25.61%	30.23%
8	37	10.26%	6.20%	26.08%	23.07%	34.39%

the current block cannot find good matches, SCM encoder will preferably choose PLT mode for the whole block without further splitting.

3.3 Inter-Frame Mode and Partition Statistics

The Inter-frame partition distribution using HM-16.4 and SCM-4.0 are provided in Table 3.5. The Inter-frame mode distribution using HM-16.4 and SCM-4.0 are provided in Table 3.7 and Table 3.6, respectively. The statistics reveals that “Merge” and “Skip” modes cover a large proportion. Since the computer-generated contents are mostly noise-free, therefore, the stationary areas within SC videos are more likely to find perfect temporal matches than natural videos and therefore coded in Skip mode at larger block sizes.

Table 3.5: Inter-frame CU Partition Statistics

CU Width	QP	HM Split	HM Unsplit	SCM Split	SCM Unsplit
64	22	19.06%	80.94%	19.09%	80.91%
64	37	13.41%	86.59%	14.99%	85.01%
32	22	47.10%	52.90%	47.82%	52.18%
32	37	44.95%	55.05%	44.49%	55.51%
16	22	51.38%	48.62%	33.76%	66.24%
16	37	49.18%	50.82%	31.74%	68.26%

Table 3.6: SCM-4.0 Inter-frame CU Mode Statistics

CU Width	QP	Intra	PLT	IBC	Merge/Skip	Inter
64	22	3.58%	0.00%	0.84%	94.29%	1.29%
64	37	5.56%	0.00%	0.69%	91.67%	2.09%
32	22	10.40%	2.03%	8.82%	65.26%	13.48%
32	37	6.27%	0.90%	8.48%	69.46%	14.89%
16	22	8.43%	1.45%	13.62%	63.48%	13.03%
16	37	2.36%	0.39%	9.61%	69.39%	18.25%
8	22	3.02%	1.70%	16.12%	58.42%	20.75%
8	37	0.89%	0.13%	12.55%	66.76%	19.68%

Table 3.7: HM-16.4 Inter-frame CU Mode Statistics

CU Width	QP	Intra	Skip	Merge	Inter
64	22	1.74%	96.20%	1.67%	0.39%
64	37	2.07%	95.76%	1.46%	0.71%
32	22	5.83%	80.27%	8.85%	5.05%
32	37	4.73%	86.55%	4.96%	3.76%
16	22	9.19%	76.18%	8.65%	5.98%
16	37	6.74%	84.65%	3.82%	4.79%
8	22	8.69%	64.21%	18.53%	8.56%
8	37	5.32%	72.27%	14.92%	7.49%

3.4 SCM Complexity Distribution Statistics

We also conduct a survey on SCM-4.0 encoder complexity distribution, using CPU tick counter to document the clock cycles consumed by the target encoding mode. Even though this complexity profiling result may differ from platform to platform, we assume the percentage of each mode will not vary significantly and should reflect the encoder complexity distribution. The profiling is conducted at each CU size, over different coding modes. The distribution for Intra-frame coding and Inter-frame coding are provided in Table 3.8 and Table 3.9 respectively. Please note this result only reflects the average complexity across sequences. The per-sequence result may vary slightly, depending on the contents.

Table 3.8: SCM-4.0 Intra-frame Complexity Statistics

CU Width	Intra	IBC Merge/Skip	IBC Inter	PLT	Total
64	6.93%	3.98%	0.00%	0.00%	10.92%
32	7.72%	6.44%	0.02%	2.34%	16.52%
16	7.44%	8.33%	6.65%	2.06%	24.48%
8	20.83%	4.87%	20.27%	2.11%	48.08%

Table 3.9: SCM-4.0 Inter-frame Complexity Statistics

CU Width	Intra	IBC	PLT	Merge/Skip	Inter	Total
64	2.80%	0.37%	0.16%	6.12%	13.11%	22.56%
32	2.16%	0.40%	0.40%	4.02%	14.72%	21.70%
16	1.42%	1.89%	0.29%	5.99%	17.71%	27.30%
8	1.77%	3.51%	0.25%	2.99%	19.72%	28.24%

Intra-frame complexity statistics from Table 3.8 coincides with SCM-4.0 implementation. The IBC complexity at CU64 and CU32 levels are negligible since IBC-Inter mode is disabled at CU64 level and IBC-Inter mode only checks 64 previous block vectors (BV) at CU32 level. At CU16 level, IBC complexity increases due to the additional 1-D search within horizontal and vertical CTU line. At CU8 level, IBC complexity increases significantly due to the global hash-based search. The major complexity is consumed by Intra and IBC-Inter modes. The CU8 Intra mode and IBC-Inter mode together consume >40% of the total encoder complexity.

As shown in Table 3.9, the major Inter-frame complexity is consumed by Inter modes (i.e., approximately 80%). This aligns with HM and SCM codec implementation that the Intra-frame modes can be fast-bypassed when temporally a perfect matching block is found for the current CU. The major complexity is introduced during the motion estimation (ME) stage, in which the motion vectors (MV) is refined progressively until a convergence at sub-pixel precision.

Chapter 4

Fast Screen Content Encoding Using Machine Learning

In this chapter, the proposed fast screen content encoding framework is presented, based on three pre-trained decision tree classifiers. The classifiers are designed to either reduce the SCC mode candidates to be examined or to make fast partition decisions, without changing the implementations of low-level Intra, IBC and PLT modes implementation. Experimental results demonstrate significant Intra-frame encoding speedup beyond anchor SCM software with a marginal coding performance loss.

4.1 Fast SCC Encoder Workflow Overview

As illustrated in Figure 4.1, the proposed encoder includes three major classification processes: In the first process, the input image block is classified into either a natural image block (NIB) or a screen content block (SCB). For NIBs, only Intra modes will be considered at the current CU level, while for SCBs, only SCC modes (i.e., IBC and PLT) will be checked at the current CU level. In the second process, NIBs are classified into either "Partitioned Blocks" (P-Blocks) or "Non-Partitioned Blocks" (NP-Blocks). P-Blocks will bypass the current level Intra mode and enter the next-level CU processing directly. For NP-Blocks, the encoder will only evaluate the current level Intra sub-modes and immediately terminate further splitting. In the last process, NP-Blocks are further classified into either "Directional Blocks" (D-Blocks) or "Non-Directional Blocks" (ND-Blocks). For ND-Blocks, the encoder checks both Intra-DC mode and Intra-Planar mode. For D-Blocks, the encoder determines the dominant edge direction and then triggers the corresponding Intra directional sub-mode. For simplicity and parallel CU processing consideration, our classifiers make fast decisions based on the features computed inside the current CU. Therefore, it is impossible for the classifier to predict whether a block has a closely-matched block in the previously-coded area using these features. Therefore, the block type classifier may erroneously classify a "Natural Image alike Block" as an NIB, when this block happens to match a previously-coded region and is coded more efficiently using an SCC mode. Because such situation mainly happens at CU8 level, therefore, even for those CU8 blocks classified as a NIB, we still check the IBC-Skip mode, because this mode only checks a small number of candidates and can often find the best match without going through full IBC search.

For each classification, we use a "soft-decision" classifier that outputs a decision confidence level. When the decision confidence for a CU is below a preset threshold, this CU is defined as a "Controversial Block" (CVB). Throughout this work, the CVBs in the first classification process (i.e., "NIB vs. SCB Classification")

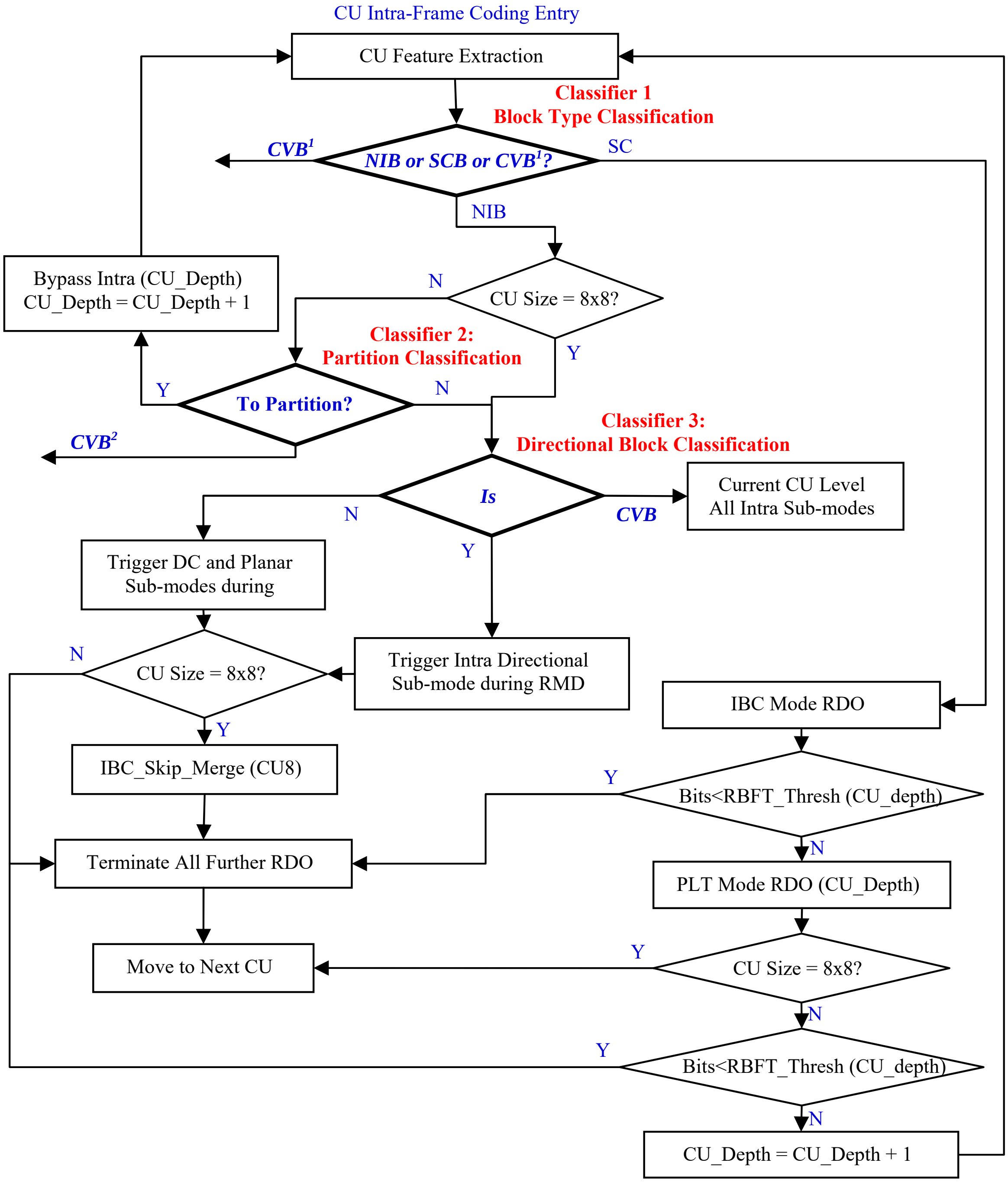


Figure 4.1: Fast Screen Content Encoding Workflow Diagram

are denoted as CVB^1. The CVBs in the second classification process (i.e., "Partition vs. Non-Partition Classification") are denoted as CVB^2. The CVBs in the third classification process (i.e., "Directional vs. Non-Directional Classification") are denoted as CVB^3. Two encoding configurations with different CVB handling strategies are proposed. Under "Rate-Distortion Preserving" (RDP) setting, for CVB^{1}s, the encoder will evaluate both Intra and SCC modes at the current level and then examine the RD performance with CU partition, where each sub-CU will be processed according to the same workflow in Figure 4.1. For CVB^{2}s, the encoder will evaluate all the Intra modes for the current CU level and then examine the RD cost with CU partition. Under "Complexity Reduction Boosting" (CRB) setting, rather than going through Intra mode full-RD evaluations, we allow fast Intra partition and sub-mode decisions: For CVB^{1}s, the encoder will firstly follow the NIB branch in Figure 4.1 to evaluate Intra modes and then check SCC modes. For CVB^{2}s, the encoder will evaluate Intra mode at the current level with fast directional block classification invoked and then proceed with CU partition and sub-CU processing. For CVB^{3}s, the encoder will evaluate all the Intra sub-modes for both RDP and CRB settings. The confidence thresholds are chosen according to the desired trade-off between the encoder Rate-Distortion(RD) performance and complexity reduction. A higher confidence threshold configuration will preserve coding efficiency better while a lower confidence threshold configuration will reduce the encoder complexity more. Different threshold settings are allowed at different CU levels, because larger CU mis-classifications may affect the coding performance more than the smaller CU mis-classifications.

At CU64 level, by default, SCM already disables PLT and IBC-Inter modes due to complexity consideration. According to the simulations, by further disabling

IBC-Merge and IBC-Skip modes at CU64 level, a 4.5% complexity reduction is achieved with almost no BD-Rate loss, since the small amount of CU64 IBC-Merge or IBC-Skip blocks can be also efficiently coded by CU32 level IBC-Merge or IBC-Skip mode. Therefore, at CU64 level, only Intra mode is enabled and block type classifier at this level is not designed.

When SCM finishes the RD-calculation for a specific mode, the optimal RD cost will be updated and the corresponding best mode will be documented if the current mode outperforms the previous best coding mode. Therefore, in addition to reducing the mode candidates using the three classifiers, we further incorporate a rate-based fast termination (RBFT) process in a similar formulation as described in [3]. Basically, when the current mode bit-consumption is lower than our statistics-based threshold (i.e., 8 bit in our configuration), we assume that the current coding mode is sufficiently efficient and the remaining modes are terminated. RBFT is invoked over SCB, CVB^1s and CVB^2s. RBFT also trades off the complexity reduction and the coding efficiency. A larger RBFT threshold configuration will terminate more CU mode checking and CU splitting and therefore promote the encoding speedup while a smaller RBFT threshold configuration will preserve the coding efficiency better.

4.2 Feature Extraction and Classifier Design

In this section, we describe the features used to drive these classifiers and the criteria used to select the classifier type, the training methodologies and the final trained decision tree structures.

4.2.1 Feature Selection

Statistically we have some prior knowledge on SC videos and the relationships between SC image patterns and mode selection. For instance, homogeneous regions are more likely to be encoded in larger CUs with Intra mode. PLT-coded CUs are more likely to contain fewer distinct colors and sharp edges. Discontinuous-tone SC areas are more likely to be encoded using IBC or PLT modes. Larger CUs are more likely to be further partitioned than smaller CUs, especially under low QP settings, and so on.

With such prior knowledge, we directly derive features for our fast mode and partition decision tasks and then apply the supervised-learning approach, rather than learning the features from the raw image blocks using deep learning techniques. The features used to train different classifiers are summarized as follows.

Feature 1: Sub-CU Horizontal and Vertical DC Difference (HVDD), as formulated in (4.1), (4.2) and (4.3), where HDD and VDD are intermediate horizontal and vertical components of sub-CU DC value difference between horizontally and vertically adjacent sub-CUs. The sub-index indicates the corresponding sub-CU location. For instance, DC_1 corresponds to the DC value of the first sub-CU located at the upper-left corner of the current CU. A CU with a smaller $HVDD$ value has a stronger horizontal or vertical directionality.

$$HDD = |DC_1 - DC_2| + |DC_3 - DC_4| \tag{4.1}$$

$$VDD = |DC_1 - DC_3| + |DC_2 - DC_4| \tag{4.2}$$

$$HVDD = min\left(HDD, VDD\right) \tag{4.3}$$

Feature 2: CU Variance as defined in (4.4), where $Y(x, y)$ is the luminance value at pixel location (x, y), and $\bar{Y}$ is the average luminance value over the current CU and N is the CU width. Variance is a good indicator of block smoothness. A CU with smaller variance is less likely to be further partitioned.

$$\sigma^2 = \frac{1}{N^2} \sum_{(x,y) \in CU} (Y(x, y) - \bar{Y})^2 \tag{4.4}$$

Feature 3: CU Gradient Kurtosis (GK). In order to evaluate whether a block has a dominant direction, we compute the histogram of the gradient orientation and then compute the orientation histogram kurtosis, which measures the histogram peakiness. To calculate this feature, the CU gradient maps are firstly derived by convolving the input image with "Sobel" masks. Let $Y_H(x, y)$ and $Y_V(x, y)$ denote the horizontal and vertical gradient at pixel location (x, y) and $Mag(x, y)$ and $Ang(x, y)$ denote the magnitude and orientation of the gradient computed based on $Y_H(x, y)$ and $Y_V(x, y)$. We firstly apply a threshold (with a value of 30 for CU64 and CU32 and 10 for CU16 and CU8) on gradient magnitude map to filter away small local texture variation. From these thresholded gradient maps, we compute the gradient angle histogram $G(\theta)$, which denotes the sum of gradient magnitudes of all pixels for each gradient angle θ between 0° and 180 ° (exclusive) with a stepsize of 1°. Finally, GK is derived as in (4.5), where $\bar{G}$ is the average gradient magnitude over the entire gradient direction histogram.

$$GK = N^2 \frac{\sum_{\theta \in [0,180)} \left[G(\theta) - \bar{G}\right]^4}{\left(\sum_{\theta \in [0,180)} \left[G(\theta) - \bar{G}\right]^2\right)^2} \tag{4.5}$$

Feature 4: CU Gradient Magnitude Peak (GMP), which is the gradient magnitude that achieves the peak over the gradient magnitude histogram (excluding zero-gradient). GMP indicates the most frequent nonzero gradient magnitude in a CU. SCBs usually consist of sharper edges and therefore have larger GMP values. The reason that we exclude zero-gradient is because such gradient is most likely to be the peak value in the majority of the blocks and does not facilitate the differentiation between NIBs and SCBs.

Feature 5: Zero Gradient Percentage (ZGP), defined as the ratio between the pixel number with zero gradient magnitude and the CU area. SCBs mostly contain a large area of constant background color and therefore have larger ZGP than smoothly-varying NIBs.

Feature 6: CU Color Number (CN). To calculate this feature, RGB or YUV components are firstly combined into a 24-bit “color triplet”. The number of distinct “color triplets” inside the current CU is counted to derive CN.

4.2.2 Classifier Selection

In recent years, several groups have considered the application of machine learning techniques for fast mode decisions in video coding and transcoding applications (e.g., [43][17][51]). In this work, several popular supervised learning methods are evaluated, including “Supported Vector Machine” (SVM), “Neural Networks” (NN) and “Decision Tree” (DT). All these three classifiers provide a similar accuracy for our classification task when the same feature set is used. For SVM, because our data samples are not linearly separable, “Gaussian Radial Basis Function Kernel” (RBF) is used to implicitly map input features onto high-dimensional space. Therefore, all the supporting vectors (SVs) derived during the training stage have to be stored within the encoder memory for future prediction and the number of SVs tends to grow larger when training samples are increased. For instance, the number of SVs is over 1600 on average when we include 10,000 random training samples. This will not only consume valuable encoder on-chip memory but also make the resultant classifier less adapted to new data. For the NN, we used a single hidden layer structure and optimized the number of hidden nodes and other network parameters through a validation process. Both NN and DT involve minimal processing complexity and can be incorporated into the encoder easily. However, the derived NN classifier structure (including weighting factors, bias terms, etc.) does not provide a good interpretation. DT training model can be easily implemented as a set of “if-then” rules. These derived conditions typically coincide with human observations and analytical reasoning, which provide valuable insights about the features and what ranges of the features are typically associated for different classes. Taken into account the prediction accuracy, model simplicity and interpretability and memory consumption, decision tree is adopted as our learning model.

4.2.3 Classifier Training

All the classifiers are trained separately for each CU size. Our training data initially contains totally 191200 CU64 samples, 764800 CU32 samples, 3059200 CU16 samples and 12236800 CU8 samples from the sample frames introduced in Table 3.1 for QP=22, QP=27, QP=32 and QP=37, respectively. If the higher CU level ground-truth decision is “Not to Partition”, all its sub-CU data samples will be deactivated and removed from our training set. Since we have sufficient samples, cross-validations are assumed unnecessary and thus not used. For simplicity, over each CU level, we randomly select half the samples to form the Training Set (TRS) and the other half samples to form the Validation Set (VLS). TRS is used to derive the DT model and parameters (e.g, the decision variables and the decision thresholds at each tree node) and VLS is used to evaluate the generalization error and prune the derived trees. Experimental results demonstrate that the division setting between TRS and VLS will not change the VLS prediction accuracy significantly since we have sufficient training and validation samples.

The ground-truth mode and partition decision labels for CU samples are obtained by encoding the sample frames in Table 3.1 using SCM-4.0 encoder using AI encoding configuration and default settings as in [75]. For each CU sample, if the entire CU (including its sub-CUs, if partitioned) is encoded using Intra mode, the block is labeled as an “NIB”. If the entire CU is encoded using an SCC mode (either IBC or PLT), or if at least one sub-CU is coded using an SCC mode, the CU is labeled as an “SCB”. The training is implemented using MATLAB built-in Classification and Regression Tree (CART) module in “Statistics Toolbox” (Version 8.3) [46].

The feature priority is determined based on maximum deviance reduction (or equivalently, the cross entropy). Namely, the feature that provides larger deviance reduction is given a higher priority and appears earlier in the decision tree. We use the sample number in the DT node as the stopping criterion and stop splitting a node as soon as the total number of training samples in the node is less than or equal to 1% of the TRS size. Then the tree structure is progressively pruned node by node backward while we track the VLS prediction accuracy until the maximum is acquired.

4.2.4 Block Type Classification

In this process, a decision tree is trained using all the features to classify the incoming CUs as either an NIB or an SCB. Training blocks for NIBs include blocks coded fully using Intra-mode. Training blocks for SCBs include blocks coded using IBC and PLT modes only, or a mixture of Intra and SCC modes. The derived decision trees for different CU levels after pruning are provided in Figure 4.2, Figure 4.3 and Figure 4.4, respectively. Please note that in this design, at CU64 level, only Intra mode is enabled, as discussed in Sec. 4.1, so no block type classifier is designed at this level.

For each decision tree leaf node, we show two numbers (in percentages) obtained during the training stage. The top number (i.e., "node probability") indicates the percentage of samples going to this node among all the training samples. The bottom number (i.e., "node accuracy") reveals the percentage of samples correctly classified among all training samples in that node. A low node accuracy indicates that the samples landing in that node cannot be classified with high confidence. In our work, we consider the node accuracy as the decision confidence. Any block landing in a leaf node with confidence lower than a preset threshold in this process will be defined as a CVB^1. Recall that CVB^1 will go through both the SCB and NIB workflows for the mode and partition decisions.

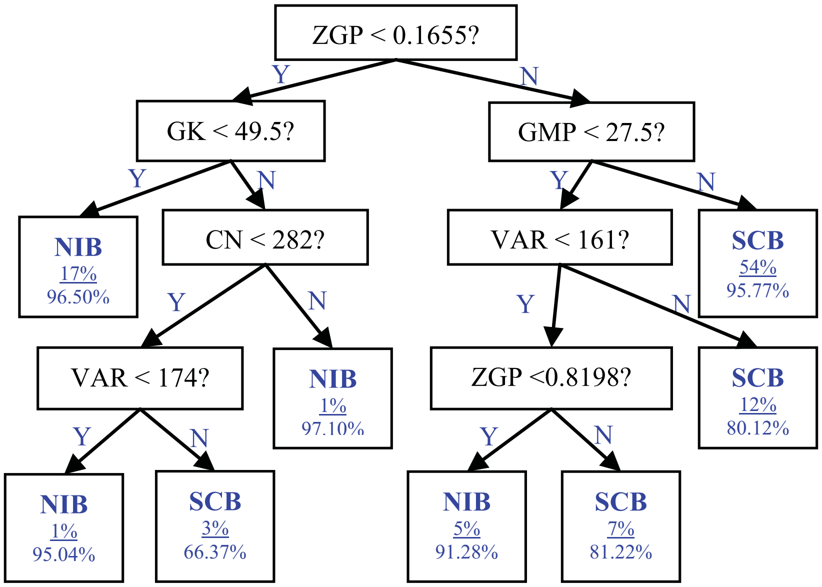


Figure 4.2: CU32 NIB-SCB Classification Decision Tree

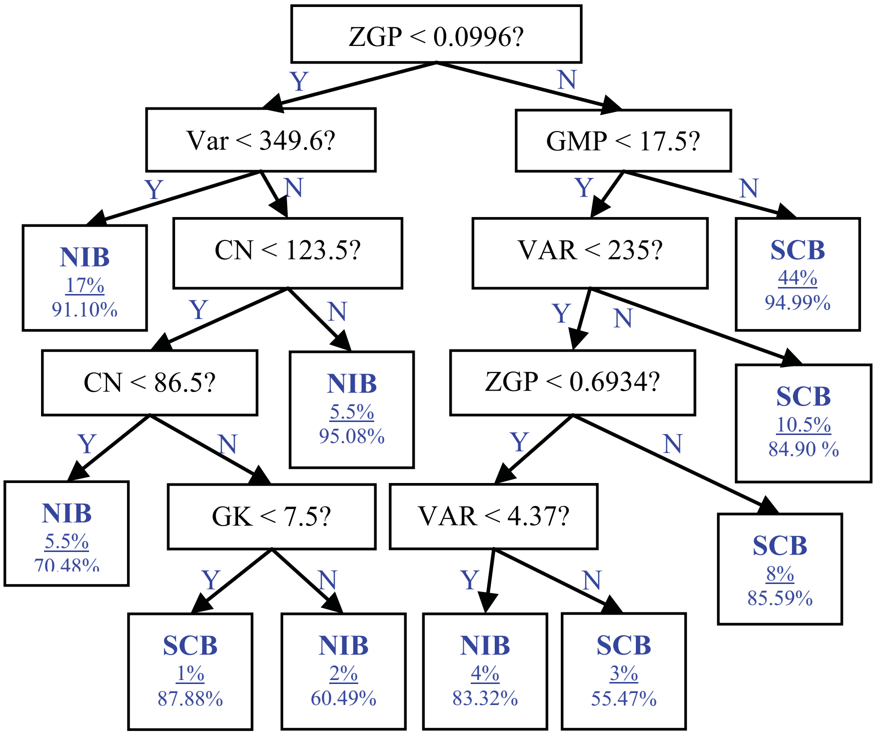


Figure 4.3: CU16 NIB-SCB Classification Decision Tree

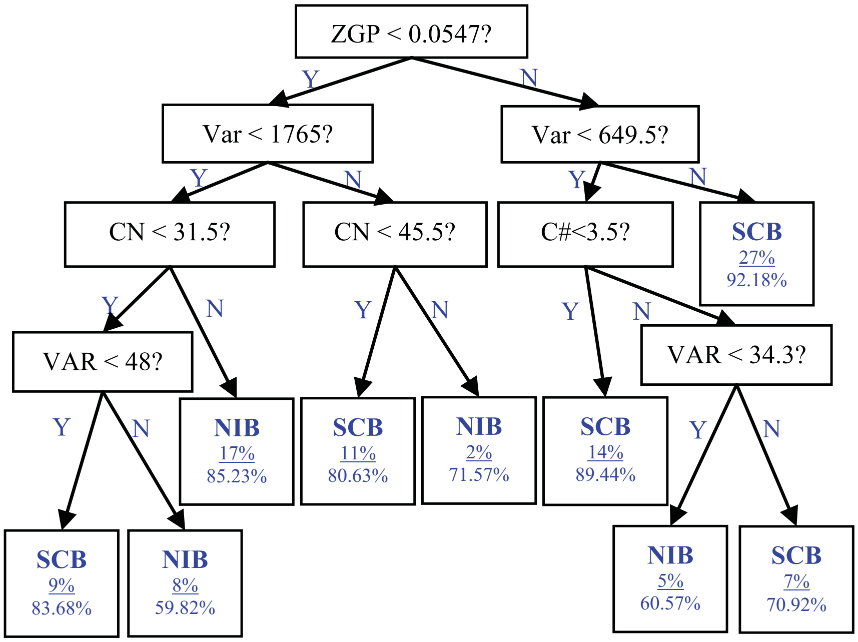


Figure 4.4: CU8 NIB-SCB Classification Decision Tree

4.2.5 Partition Decision

In this process, decision trees are trained to classify NIBs into either “Partitioned Blocks” (P-Blocks) or “Non-Partitioned Blocks” (NP-Blocks) based on the CU homogeneity. Statistically, NIBs with larger variance, sub-CU mismatch and smaller gradient kurtosis are more likely to be further partitioned. Therefore, we use features 1, 2, 3 to train this classifier over each CU level. HVDD and Variance describe the Sub-CU homogeneity, while GK reflects CU orientation homogeneity. P-Blocks with high confidence level will directly bypass current level Intra mode selection and directly enter the next-level CU processing. NP-Blocks with high confidence level will only examine the current level Intra mode and immediately terminate further splitting. The trained decision trees for different CU levels are provided in Figure 4.5, Figure 4.6 and Figure 4.7, respectively.

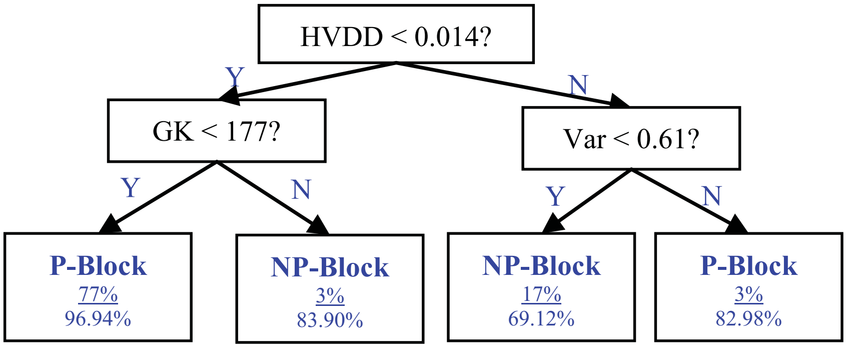


Figure 4.5: CU64 NP/P-Block Classification Decision Tree

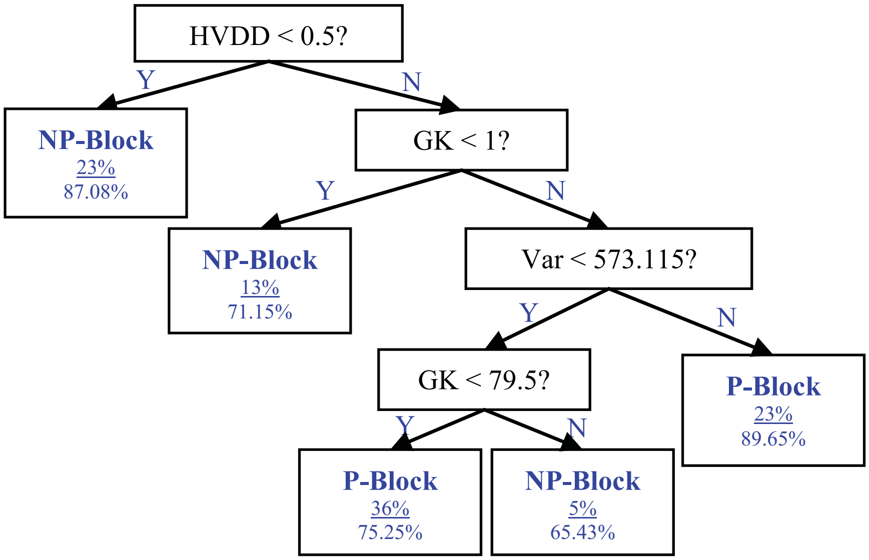


Figure 4.6: CU32 NP/P-Block Classification Decision Tree

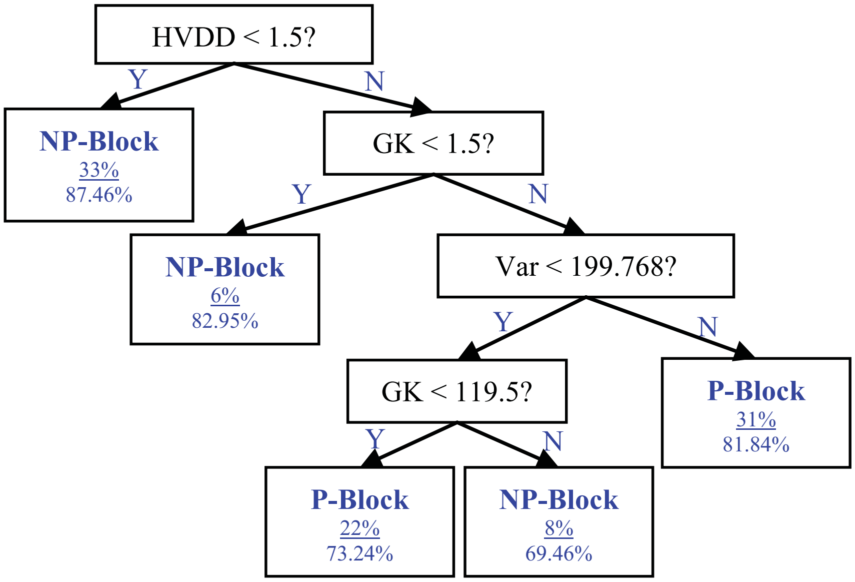


Figure 4.7: CU16 NP/P-Block Classification Decision Tree

4.2.6 Intra Sub-mode Decision

To classify NP-Blocks into "Directional Blocks" (D-Blocks) or "Non-Directional Blocks" (ND-Blocks), features 1, 2, 3 are used for training because directional CU usually has a dominant direction that can be reflected by a larger GK value, while non-directional CU is usually smooth with smaller HVDD, Variance and GK. The resultant classifiers are provided in Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11, respectively.

4.3 Experimental Results and Evaluation

The proposed machine learning based fast Intra-frame coding framework is evaluated and compared with SCM-4.0 anchor software, following the CTC as defined in [75]. 13 standard SCC sequences are evaluated under four QP settings (i.e., 22, 27, 32 and 37) under All-Intra (AI) configurations. To verify that our

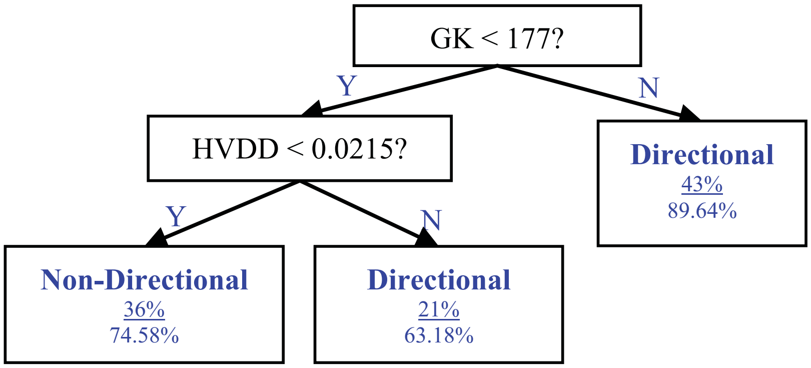


Figure 4.8: CU64 ND/D-Block Classification Decision Tree

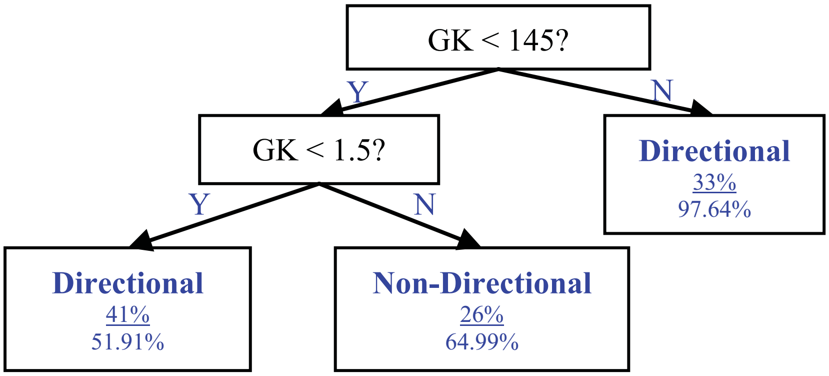


Figure 4.9: CU32 ND/D-Block Classification Decision Tree

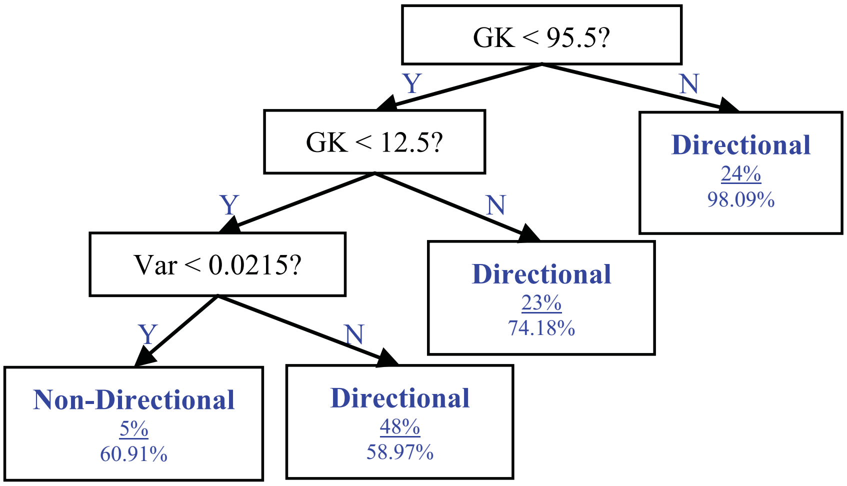


Figure 4.10: CU16 ND/D-Block Classification Decision Tree

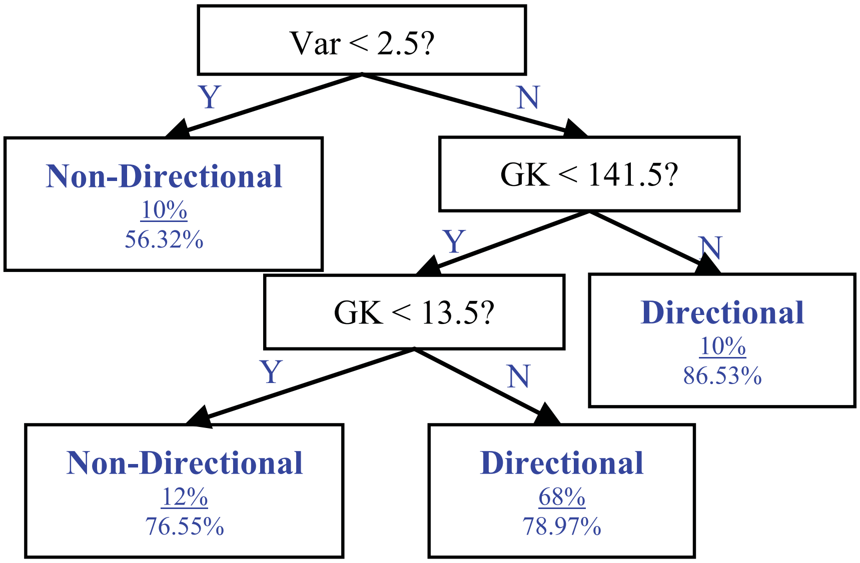


Figure 4.11: CU8 ND/D-Block Classification Decision Tree

proposed classifiers are generalizable over new SC videos, we further evaluate our proposed framework over unseen SC testing sequences (from [74], [69], [10] and [7]) proposed in the previous JCTVC meetings. The coding performances are evaluated using homogeneous Windows 7 (64-bit) desktops with Intel-i5 CPU (2.67 GHz dual-cores) and 4GB RAM.

The coding performance is measured using BD-Rate [2] against SCM-4.0 encoder. The complexity saving is measured directly using the relative reduction of the encoding time, as defined in (4.6), where T_{anchor} is the encoding time of SCM-4.0 encoder and $T_{proposed}$ is the encoding time of the proposed encoder scheme over the same sequence.

$$\Delta C = \frac{T_{anchor} - T_{proposed}}{T_{anchor}} \times 100\% \tag{4.6}$$

Two sets of simulation results are provided with different encoding settings.

The “Rate-Distortion Preserving” (RDP) configuration focuses on coding efficiency preservation, while the “Complexity Reduction Boosting” (CRB) configuration focuses on encoder acceleration. The summarized experimental results are provided in Table 4.1 and Table 4.2. ΔR and ΔC represent the BD-rate increment and encoding time reduction in percentages, respectively. The sequences marked with * are those videos whose 10 sample frames were selected for training. The results reported are the full-sequence encoding statistics. The sequences marked with $^+$ are not used during training and reported to validate the machine learning model generalization. The detailed per-sequence per-QP encoding results can be found in [18].

Table 4.1: Proposed Framework Coding Efficiency and Complexity Reduction Evaluation using “Rate-Distortion Preserving” (RDP) Configuration

Sequence	Resolution	Category	ΔR	ΔC
FlyingGraphics*	1920×1080	Text & Graphics	+1.02%	-40.40%
Desktop*	1920×1080	Text & Graphics	+1.93%	-46.15%
Console*	1920×1080	Text & Graphics	+1.78%	-46.53%
WebBrowsing*	1280×720	Text & Graphics	+1.51%	-48.05%
Map*	1280×720	Text & Graphics	+1.21%	-34.73%
Programming*	1280×720	Text & Graphics	+1.33%	-36.46%
SlideShow*	1280×720	Text & Graphics	+2.52%	-52.99%
BasketballScreen*	2560×1440	Mixed Content	+1.36%	-33.53%
MissionControlClip2*	2560×1440	Mixed Content	+2.57%	-42.86%
MissionControlClip3*	1920×1080	Mixed Content	+1.80%	-38.55%
Robot*	1280×720	Animation	+1.19%	-32.02%
EBURainFruits$^+$	1920×1080	Camera-Captured	+1.19%	-30.15%
Kimono1$^+$	1920×1080	Camera-Captured	+1.46%	-31.62%
Doc$^+$	1280×720	Text & Graphics	+1.34%	-41.69%
PptDocXls$^+$	1280×720	Text & Graphics	+1.54%	-39.95%
TwistTunnel$^+$	1280×720	Text & Graphics	+0.45%	-33.92%
VideoConfDocSharing$^+$	1280×720	Text & Graphics	+1.66%	-42.63%
Viking$^+$	1280×720	Animation	+1.42%	-29.02%
Web$^+$	1280×720	Text & Graphics	+0.91%	-49.64%
WordEditing$^+$	1280×720	Text & Graphics	+1.09%	-43.87%

Table 4.2: Proposed Framework Coding Efficiency and Complexity Reduction Evaluation using "Complexity-Reduction Boosting" (CRB) Configuration

Sequence	Resolution	Category	ΔR	ΔC
FlyingGraphics*	1920×1080	Text & Graphics	+4.84%	-46.33%
Desktop*	1920×1080	Text & Graphics	+3.67%	-50.63%
Console*	1920×1080	Text & Graphics	+4.32%	-51.01%
WebBrowsing*	1280×720	Text & Graphics	+3.93%	-57.13%
Map*	1280×720	Text & Graphics	+4.15%	-46.86%
Programming*	1280×720	Text & Graphics	+5.09%	-47.50%
SlideShow*	1280×720	Text & Graphics	+3.82%	-62.01%
BasketballScreen*	2560×1440	Mixed Content	+4.18%	-49.27%
MissionControlClip2*	2560×1440	Mixed Content	+4.46%	-57.50%
MissionControlClip3*	1920×1080	Mixed Content	+4.29%	-52.30%
Robot*	1280×720	Animation	+3.26%	-55.39%
EBURainFruits+	1920×1080	Camera-Captured	+1.54%	-52.20%
Kimono1+	1920×1080	Camera-Captured	+1.38%	-49.45%
Doc+	1280×720	Text & Graphics	+3.59%	-53.32%
PptDocXls+	1280×720	Text & Graphics	+2.66%	-47.76%
TwistTunnel+	1280×720	Text & Graphics	+4.27%	-44.24%
VideoConfDocSharing+	1280×720	Text & Graphics	+3.69%	-52.84%
Viking+	1280×720	Animation	+3.59%	-55.30%
Web+	1280×720	Text & Graphics	+2.47%	-51.79%
WordEditing+	1280×720	Text & Graphics	+3.71%	-49.77%

Compared with the anchor SCM-4.0 software with default Full-Frame IBC (FF-IBC) configuration, our proposed fast Intra-coding framework can achieve a complexity reduction of 40% on average under RDP setting with 1.46% negligible BD-loss. Under CRB setting, we achieve 52% complexity reduction on average with 3.65% acceptable BD-Rate loss. Given the space limit, only YUV-444 results are provided. However, the proposed framework and methodologies can be easily generalized onto RGB-4:4:4 sequences and YUV-4:2:0 sampling format.

From the simulation results, the following conclusions can be drawn.

Firstly, the proposed framework has less gain over the sequences coded in smaller CU sizes, because fewer blocks can be fast-terminated. For instance, "Map"

Table 4.3: Per-Classifier Contribution Analysis

Category	T&G		MC		ANM		CC	
Beyond SCM	ΔR	ΔT	ΔR	ΔT	ΔR	ΔT	ΔR	ΔT
C1	+0.9%	-31%	+0.5%	-36%	+0.3%	-23%	+0.3%	-21%
C1+C2	+1.1%	-34%	+1.0%	-40%	+1.2%	-28%	+1.2%	-27%
C1+C2+C3	+1.1%	-35%	+1.1%	-41%	+1.5%	-30%	+1.3%	-29%

and “Robot” sequences are dominated by small CU8 blocks and have the minimum complexity reductions among all the sequences. The proposed framework has larger complexity reductions over sequences coded in higher QP values, because more SCBs can find a perfect or good match under higher QP settings during IBC search.

Secondly, the derived machine learning model is generalizable to unseen SC videos. We observe comparable BD-rate increase and similar complexity reduction over the unseen SC sequences. Besides, the derived machine learning model does not degrade the coding performances over camera-captured videos.

Thirdly, the proposed framework is scalable and controllable. By adjusting classifier confidence thresholds and RBFT thresholds, desired trade-off can be achieved between the encoder efficiency and complexity for various applications. Additionally, the per-classifier results over different contents, including “Text & Graphics”(T&G), “Mixed Content”(MC), “Animation”(ANM) and “Camera-Captured” (CC) are summarized in Table 4.3, in which “C1”, “C2” and “C3” represent the first, second and third classifiers, respectively. From this result, we can see that the complexity is primarily saved from the block type classifier. “Animation” and “Camera-Captured” videos behave similarly in the proposed framework. The contribution from Classifier 3 is relatively small, given that SCM-4.0 has already embedded fast Intra mode candidate reduction algorithm [54].

Finally, the proposed machine learning based framework is compared and evaluated against some previous SCC fast encoding solutions. Table 4.4 demonstrate that our proposed framework achieves substantially more complexity reductions than the methods of [33], [66] and [81], with only slightly higher BD-rate increase. Compared with [79], the proposed framework achieved a slightly higher complexity reduction, but noticeably higher BD-rate loss. However, the method proposed in [79] relies on stationary region detection in the current frame and reuses the partition side information from the stationary regions in the previous frame. Our proposed framework is self-contained and does not require temporal information from other frames. When the access to the previous frame mode and partition decisions is feasible, the proposed approach can be applied to the non-stationary SC regions, to achieve more complexity savings than both the current approach and [79].

Table 4.4: Coding Efficiency and Complexity Comparisons with Prior Work

Prior Work	Codebase	Complexity Reduction	BD-Rate Loss
Kwon and Budagavi [33]	HM-12.0	22%-31%	0.5%
Tsang, Chan and Siu [66]	SCM-2.0	8%-29.2%	0.1%-0.7%
Zhang, Guo and Bai [81]	HM-12.1	32%	0.8%
Zhang and Ma [79]	SCM-3.0	39%	1.1%
Duanmu, Ma, Wang [17]	SCM-4.0	37%	3.0%
Proposed (RDP)	SCM-4.0	40%	1.4%
Proposed (CRB)	SCM-4.0	52%	3.6%

Chapter 5

Fast HEVC-SCC Transcoding Using Machine Learning

In this chapter, a fast heterogeneous transcoding framework is proposed to convert baseline HEVC bitstream to HEVC-SCC bitstream for Intra-frame coding, as illustrated in Figure 5.1. Specifically, a block type classifier is trained to accurately classify the incoming CUs into either "Natural Image Block" (NIB) or "Screen Content Block" (SCB). The NIBs will directly inherit HEVC intra mode and partition decisions, while the SCBs will be directly coded using SCC modes (i.e., IBC or PLT). The proposed framework is implemented as a "pre-processing" module in SCM software. Both CU statistical features (such as CU color quantity, CU pixel variance, CU edge directionality distribution, etc.) and the decoded HEVC side information (such as CU partitions, modes, residual, etc.) are jointly analyzed to derive fast CU partition and mode decisions. Experimental results demonstrate significant Intra-frame transcoding speedup beyond the full-encoding solution with only marginal coding efficiency loss.

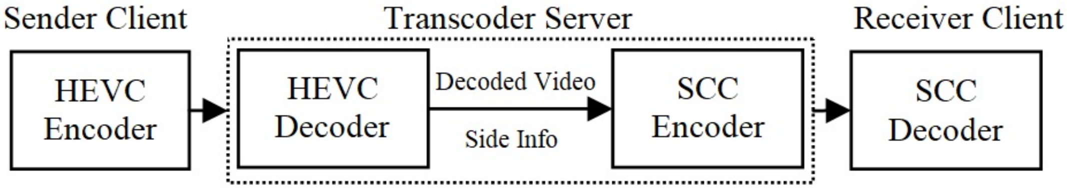


Figure 5.1: HEVC-SCC Transcoding Framework

5.1 Fast HEVC-SCC Transcoder Design

The proposed fast HEVC-SCC transcoder is designed and implemented as a pre-processing module before Intra-frame mode selection, as compared in Figure 5.2. To summarize, the non-partitioned block mode and partition decisions will be directly inherited during transcoding. Over HEVC-partitioned blocks, a block classifier is designed and embedded in the encoder to categorize the incoming blocks into either a natural image block (NIB) or screen content block (SCB). The NIBs will directly bypass the current level CU processing, while the SCBs will be directly coded using only SCC modes (i.e., IBC or PLT). This framework design arises from the following observations.

Firstly, over flat, smoothly-varying or directional blocks illustrated in Figure 5.3, HEVC and SCC encoders will both use Intra-mode without further partitions. Therefore, the HEVC-SCC transcoder may simply inherit the Intra sub-mode from the HEVC bitstream and directly apply to SCC bitstream.

Secondly, SCC modes (e.g., IBC or PLT) enable “inhomogeneous” blocks to be encoded in larger CU sizes. As illustrated in Figure 2.2, compared with Intra-mode, PLT and IBC modes are mostly chosen at a larger CU size. Therefore, an intuitive yet safe transcoding heuristic is that CU coding depth using SCC should be shallower than the coding depth using HEVC. For example, in Figure 2.2, the

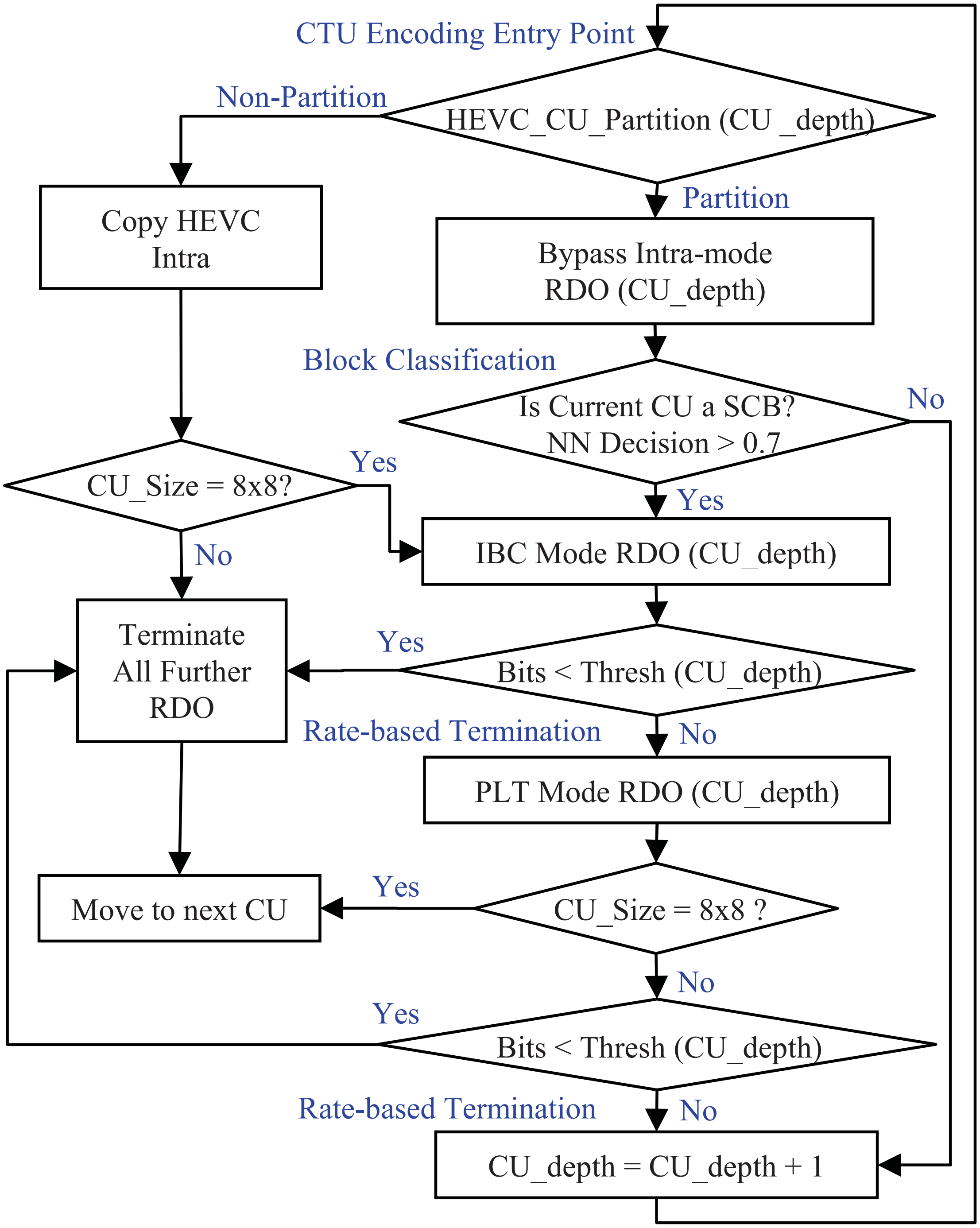


Figure 5.2: HEVC-SCC Transcoding Workflow

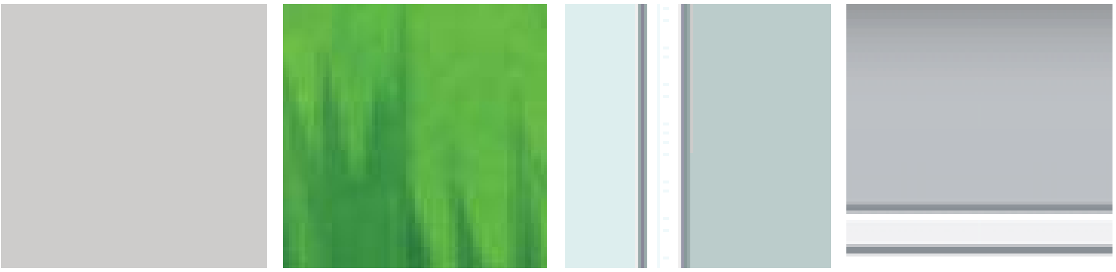


Figure 5.3: Sample Intra-coded Blocks in HEVC and SCC

RD-optimal coding depth is 3 using SCC modes but 4 using conventional HEVC modes.

Thirdly, computer-generated areas (such as icon, graphics) are usually coded using PLT mode or IBC mode. Camera-captured areas (such as natural picture) are mostly coded using Intra-mode. Therefore, a fast and accurate classification between Screen Content Block (SCB) and Natural Image Block (NIB) can effectively reduce the mode candidates from {Intra, PLT, IBC} to either {Intra} or {PLT, IBC}. To design such a classifier, both decoded side information and CU block-level features are jointly analyzed, as detailed in Section 5.2.

5.2 Block Type Classifier Design

Based on the statistical prior knowledge, a block type classifier is trained using Neural Network (NN) to efficiently categorize the incoming CU into either an “SCB” or “NIB”. Four features are selected as follows.

Feature 1: CU Variance as defined in Eq. (4.4).

Feature 2: CU Color Number (CN), defined as the number of distinct “color triplets” inside the current CU.

Feature 3: CU Gradient Kurtosis (GK), as defined in Eq. (4.5).

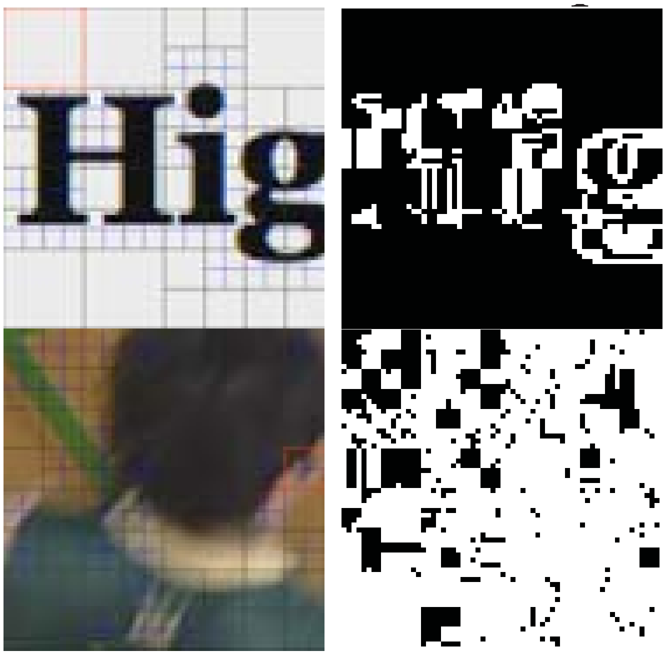


Figure 5.4: SCB and NIB Intra-coding Residual Analysis (Top: SCB sample; Bottom: NIB sample; Left: Image Pattern; Right: Residual Map using white pixels to indicate nonzero entries)

Additionally, based on the encoder behavior, even though HEVC splits large SCBs into smaller Intra CUs, the final residual image after splitting is mostly sparse, whereas for NIBs, the residual image is less sparse, as shown in Figure 5.4 in the right column. In this example, the SCB residual (after Intra prediction) has only 711 nonzero pixels while the NIB residual has 3273 nonzero pixels. Accordingly, an additional residual feature is introduced, as follows.

Feature 4: Residual sparsity (RS), defined as the $L0$ norm of residual image block.

Our training and validation set (TVS) contains totally 5,940 CU64 samples, 23,760 CU32 samples, 96,480 CU16 samples from three SCC standard sequences (i.e., “Console”, “Desktop”, “FlyingGraphics”) in CTC [75]. Since that our preliminary study shows that IBC mode utilization at CU8 level is determined by global search rather than local statistics, we do not apply block classification at this CU depth to preserve the coding efficiency. For simplicity and without loss of generosity, we randomly choose half the samples as the training set (TRS) and the other half as the validation set (VLS).

Our ground-truth block labels are obtained by re-encoding the decoded HEVC videos according to the SCC CTC [75] using SCM-4.0 encoder. If the entire block is coded using the Intra mode, regardless of the Sub-CU partitions, the block is labeled as an "NIB". If the block is coded using SCC mode, either purely by SCC or a mixture of Intra and SCC modes, the block label is labeled as an "SCB".

Block classifiers over each CU size are trained with a 2-layer NN structure with two "sigmoid" transfer functions between the input and the hidden layer and between the hidden layer and the output, respectively. Considering that we have sufficient samples, cross-validation is therefore assumed unnecessary and not used. We tune the optimal hidden layer node numbers directly according to the prediction accuracies over the VLS. In our models, the optimal hidden node numbers are 2, 3 and 3 for CU64, CU32 and CU16, respectively. The NN training is implemented using MATLAB NN Toolbox (Ver. 8.1)[45].

The NN classifiers output a soft-decision between 0 and 1. A block with a decision value closer to 1 has a higher probability to be an "NIB". On the contrary, a block with a decision value closer to 0 has a higher probability to be an "SCB". To preserve the coding performance, we apply a biased decision boundary value of 0.7 empirically. Namely, we classify a block as an "NIB" only if the NN decision is greater than 0.7 and bypass the SCC modes at the corresponding CU depths. The boundary selection provides a tunable trade-off between the coding efficiency and the complexity saving. A reduced decision boundary configuration will classify more blocks into the “NIB” category and thus increase the complexity reduction but will simultaneously degrade the coding efficiency.

The proposed framework is further incorporated with Rate-based Fast Termination (RBFT) as introduced in Section 4.1. The rate thresholds are determined by the encoding bits consumed at each CU depth and are conservatively chosen to preserve the coding performance. The rate thresholds used are 20, 12 and 4 for CU32, CU16 and CU8, respectively.

5.3 Experimental Results

Our proposed HEVC-SCC fast transcoding framework is evaluated as follows: Firstly, 7 standard SCC sequences (with sample frames shown in Figure 3.1) are encoded using anchor HEVC encoder (i.e., HM-16.4) according to common testing conditions (CTC) with 4 QPs (i.e., 22, 27, 32 and 37, respectively). Afterward, the HEVC bitstream is decoded into distorted YUV videos and the side information (e.g., mode, partition, residual, etc.) is retrieved and cached. Finally, our proposed transcoding system will load and analyze distorted videos and the side information and re-encode decoded YUV videos using same QP configurations.

To verify the model generalization accuracies, the trained classifiers are applied over four unseen SCC sequences (i.e., “Programming”, “SlideShow”, “WebBrowsing” and “BasketballScreen”) in the testing set (TSS) for performance validation.

The screen content re-encoding performances are evaluated using homogeneous Windows 7 (64-bit) desktops with Intel-i5 CPU (2.67 GHz dual cores) and 4GB RAM. The complexity reduction is directly measured by the encoding time reduction. Compared with the SCM-4.0, our proposed framework can achieve a 48%

re-encoding complexity reduction on average with only 2.0% BD-Rate loss. The detailed simulation results are summarized in Table 5.1.

Table 5.1: Fast HEVC-SCC Transcoding Framework Performance

Sequence	Anchor		Proposed		Performance	
	Rate	PSNR	Rate	PSNR	ΔR	ΔT
Programming	542536	49.62	549720	49.64	+1.05%	-43%
	386224	45.42	389408	45.31		
	267264	40.79	268360	40.85		
	192944	36.44	197568	36.45		
SlideShow	373184	51.78	378696	51.66	+2.21%	-51%
	268352	47.95	270584	47.80		
	195680	44.46	197320	44.30		
	134056	40.48	134576	40.36		
WebBrowsing	286456	52.53	290520	52.58	+3.08%	-48%
	232328	48.03	236392	47.80		
	169048	44.23	171344	43.93		
	127440	38.05	130504	37.77		
BasketballScreen	127013	52.14	131481	51.93	+3.13%	-46%
	85529	48.27	86322	48.19		
	59184	45.26	60575	45.13		
	39551	41.23	39939	41.11		
Console	627400	51.79	640960	51.95	+1.85%	-49%
	558648	47.25	569968	47.17		
	465992	42.97	473880	43.01		
	358304	38.47	363968	38.48		
Desktop	708400	50.59	719056	50.57	+1.72%	-48%
	637944	46.13	646584	46.27		
	599872	40.30	611928	40.43		
	522744	35.40	539160	35.34		
FlyingGraphics	1431048	48.75	1467784	48.74	+1.94%	-50%
	1105376	44.30	1130208	44.31		
	824000	40.32	829944	40.24		
	550840	35.95	559640	35.90		

Chapter 6

Fast SCC-HEVC Transcoding Using Statistical Mode Mapping

In this chapter, we propose a fast SCC-HEVC transcoding solution based on statistical mode mapping techniques, as illustrated in Figure 6.1. This is the first work addressing both Intra-frame and Inter-frame SCC-HEVC transcoding and enables the backward bitstream-compatibility over the legacy HEVC devices. Experimental results demonstrate that the proposed solution achieves a significant transcoding speedup.

Furthermore, we generalize the proposed framework to support Single-Input-Multiple-Output (SIMO) transcoding for adaptive streaming over the edge clouds, to accommodate heterogeneous end users with different networks (e.g., WiFi, LTE, etc.) and device constraints, such as battery life, display resolution, etc. Experimental results demonstrate that the proposed framework realizes the parallel screen content transcoding and achieves a significant transcoding speedup.

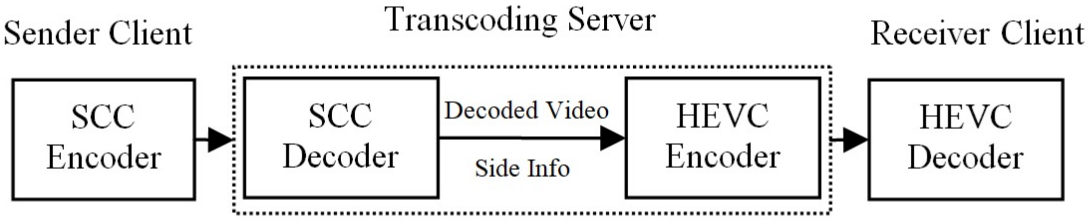


Figure 6.1: SCC-HEVC Transcoding Framework

6.1 Fast SCC-HEVC Transcoder Design

The fast SCC-HEVC transcoding workflow is designed as shown in Figure 6.2. The design philosophy of the proposed framework is summarized as follows.

Firstly, over the flat, smooth or directional SC blocks, SCC and HEVC encoders will both use Intra mode without further partitions. Therefore, the transcoder may directly copy the Intra sub-mode from SCC bitstream and apply to HEVC bitstream.

Secondly, for temporally-predictable blocks, HEVC and SCC encoders will both use Inter mode (e.g., Merge, Skip or Inter). Therefore, the transcoder may directly reuse the motion information (including reference frame index, motion vector, etc.) decoded from the SCC bitstream. Even though Merge, Skip and Inter modes are neighbor-dependent (i.e., the same block may choose a different motion vector when its AMVP or Merge candidates get updated), the motion vector (MV) derivation process can be mostly bypassed and therefore the major complexity can be saved from the motion estimation (ME) and sub-pixel interpolation processes.

Thirdly, over PLT-coded blocks, the decoded index map (IM) reflects the CU structure and texture directionality, as shown in Figure 6.3. When the index map structure is purely flat, horizontal or vertical, during transcoding, the HEVC encoder can directly trigger the corresponding Intra sub-mode and terminate CU

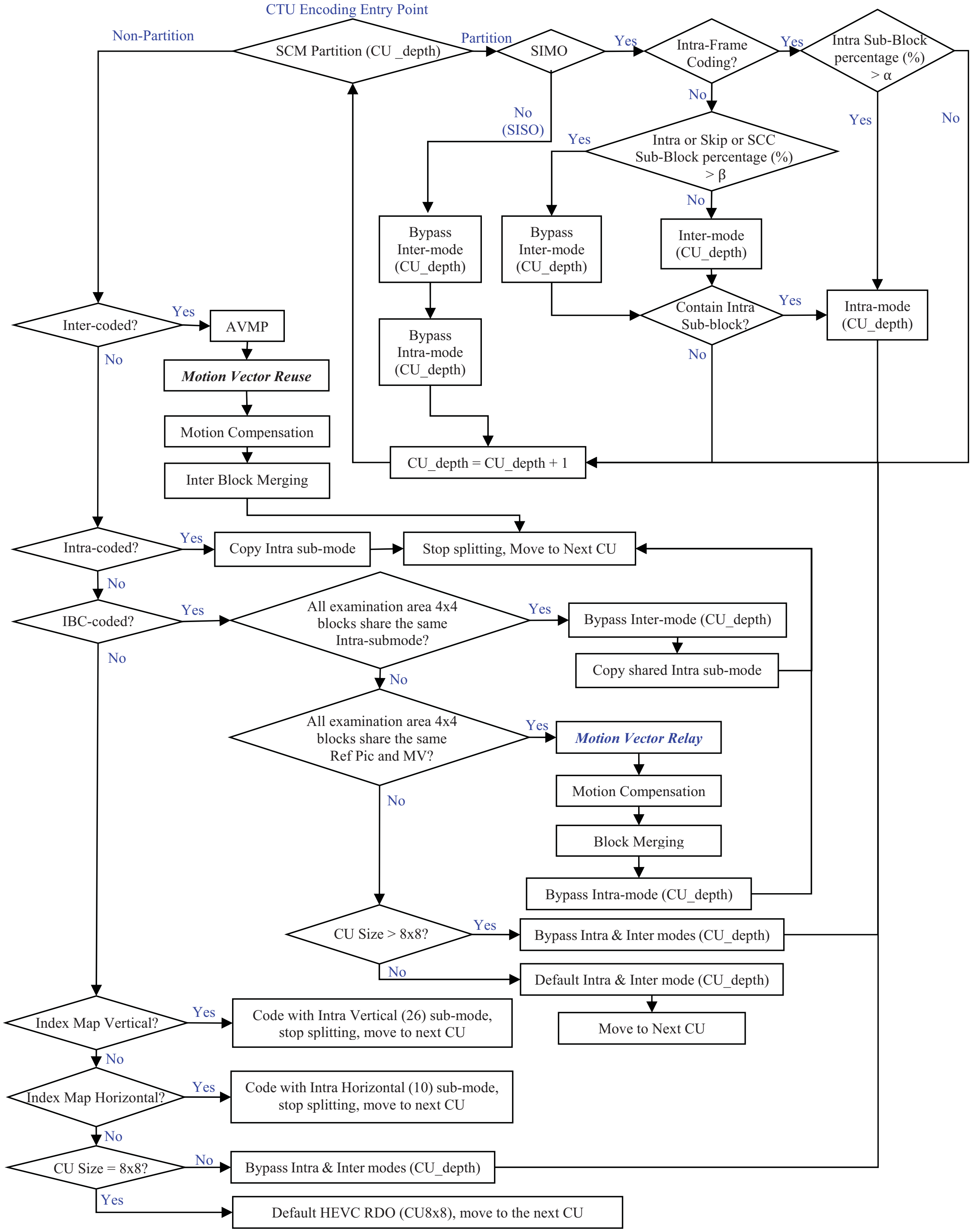
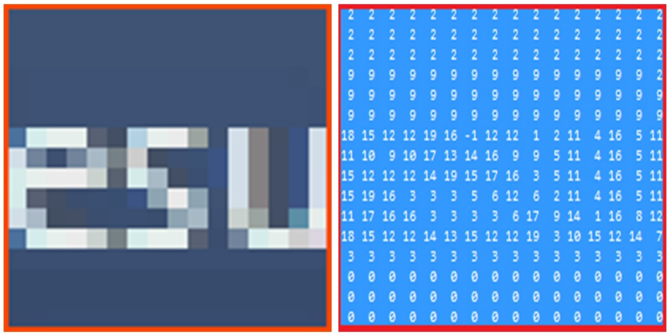

CTU Encoding Entry Point
Non-Partition
SCM Partition (CU _depth)
Partition
SIMO
Yes
Intra-Frame Coding?
Yes
Intra Sub-Block percentage (%) > α
No
(SISO)
No
Yes
No
Intra or Skip or SCC Sub-Block percentage (%) > β
Yes
No
Bypass Inter-mode (CU_depth)
Bypass Inter-mode (CU_depth)
Inter-mode (CU_depth)
Bypass Intra-mode (CU_depth)
Contain Intra Sub-block?
Yes
Intra-mode (CU_depth)
No
Inter-coded?
Yes
AVMP
No
Motion Vector Reuse
Motion Compensation
Inter Block Merging
CU_depth = CU_depth + 1
Intra-coded?
Yes
Copy Intra sub-mode
Stop splitting, Move to Next CU
No
IBC-coded?
Yes
All examination area 4x4 blocks share the same Intra-submode?
Yes
Bypass Inter-mode (CU_depth)
Copy shared Intra sub-mode
No
No
All examination area 4x4 blocks share the same Ref Pic and MV?
Yes
Motion Vector Relay
Motion Compensation
Block Merging
No
Bypass Intra-mode (CU_depth)
CU Size > 8x8?
Yes
Bypass Intra & Inter modes (CU_depth)
No
Default Intra & Inter mode (CU_depth)
Move to Next CU
Index Map Vertical?
Yes
Code with Intra Vertical (26) sub-mode, stop splitting, move to next CU
No
Index Map Horizontal?
Yes
Code with Intra Horizontal (10) sub-mode, stop splitting, move to next CU
No
CU Size = 8x8?
No
Bypass Intra & Inter modes (CU_depth)
Yes
Default HEVC RDO (CU8x8), move to the next CU

Figure 6.2: SCC-HEVC Transcoding Workflow



2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	2
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
18	15	12	12	19	16	-1	12	12	1	2	11	4	16	5	11
11	10	9	10	17	13	14	16	9	9	5	11	4	16	5	11
15	12	12	12	14	19	15	17	16	3	5	11	4	16	5	11
15	19	16	3	3	3	5	6	12	6	2	11	4	16	5	11
11	17	16	16	3	3	3	3	6	17	9	14	1	16	8	12
18	15	12	12	14	13	15	12	12	19	3	10	15	12	14	7
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6.3: Sample PLT Block and Corresponding Index Map

splitting. Otherwise, the encoder can safely bypass the Intra-mode coding at the current CU depth.

Fourthly, over IBC-coded blocks, the decoded block vectors (BV) may be used to locate the matching block in the previously-coded area or in the reference frame. Considering that the matching block is the same as or similar to the current CU, therefore, the transcoder can infer the mode and partition from the matching block and its neighbors. If all the 4x4 blocks covered by the matching block (marked inside the yellow "examination area") are coded using the same Intra sub-mode, as illustrated in Figure 6.4, the current CU (marked as a green box) can directly copy the same Intra sub-mode and terminate CU splitting. If all the 4x4 blocks are Inter-coded and share the same MV and the same reference picture list and reference frame index, as illustrated in Figure 6.5, the current CU may directly reuse the MV and the corresponding reference picture. The final motion vector mv_{final} with respect to the reference frame can be calculated as in Eq. (6.1), where bv denotes the block vector from the current CU to its IBC matching block (IBC-MB) inside the current frame and $mv_{matching}$ denotes the motion vector from the IBC-MB to its inter-frame matching block (Inter-MB). The sum of these two terms indicates the final motion vector from the current CU to its temporal matching

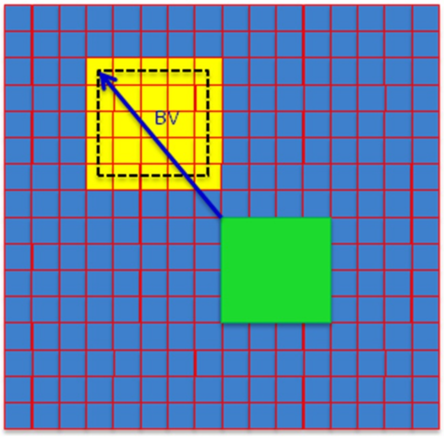


Figure 6.4: Block Vector Reuse for IBC-Intra Mode Mapping. Green Block: the current CU; Black Block in dashed line: the matching Block; Blue Arrow: IBC Block Vector; Yellow Blocks: Mode Examination Area; Each small blue square represents a 4x4 image block.

block (i.e., the spatial-temporal relayed translational offset). Otherwise, if the 4x4 blocks do not share the same coding mode or the same motion (requiring same motion vector, reference frame index), the current CU will be directly partitioned without going through the current level CU processing.

$$mv_{final} = bv + mv_{matching} \tag{6.1}$$

Finally, as illustrated in Figure 2.2, PLT and IBC modes enable "inhomogeneous" blocks to be encoded at a larger CU size. Therefore, an intuitive yet safe transcoding heuristic is that the coding depth in HEVC should be greater than the depth in SCC over the same block. For the block in Figure 2.2, the optimal coding depth is 3 using SCC modes but 4 using HEVC Intra mode. Such relation holds for both Intra-frame and Inter-frame coding.

To sum up, in our proposed framework, the Intra-mode and Inter-mode mode decisions are directly inherited from the SCC bitstream. To be specific, HEVC

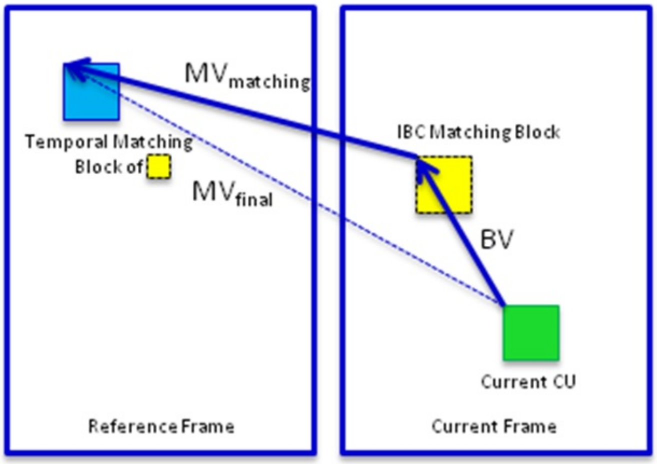


Figure 6.5: Block Vector Reuse for IBC-Inter Mode Mapping. Green Box: the current CU; Yellow Box: IBC Matching Block; Blue Box: Inter-frame matching block of IBC matching block; Blue Dashed Line: the final “relayed” motion vector from the current CU to its temporal matching block.

Intra coding will directly copy SCC Intra sub-mode. HEVC Inter coding directly reuses the decoded motion vector. Even though “Advanced Motion Vector Prediction” (AMVP) and “Block Merging” are both dependent on the spatial and temporal candidates, however, the most computationally-expensive motion estimation (ME) stage (consisting of Enhanced Predictive Zonal Search (EPZS) and sub-pixel interpolation) can be avoided. To preserve rate-distortion (RD) performance, merge/skip mode is always evaluated since it is computationally-light and efficient.

For the blocks coded in PLT mode, for simplicity, fast termination is implemented only over blocks with horizontal and vertical patterns, which are dominantly distributed in SC videos, whereas the non-directional blocks (e.g., text, icon, etc.) have to be split into very small Intra blocks to be homogeneous, and therefore can be safely fast-bypassed at larger CU sizes. Please note that sometimes flat blocks are sporadically coded using PLT mode. Such blocks can be treated as a special horizontal or vertical block in our framework.

For the blocks coded in IBC mode, three scenarios are considered. If (a) the blocks in "examination area" are all coded in the same Intra mode, the current CU can directly copy this Intra sub-mode without further partitioning. If (b) the blocks in the "examination area" are coded using the same motion (i.e., requiring the same reference list, reference picture index and motion vector), then the "relayed" motion vector (as illustrated in Figure 6.5) can be used to derive the final MV from the current CU to its temporal matching block. Since IBC has been used frequently in both Intra-frame and Inter-frame coding, such design achieves a significant speedup. If neither condition (a) nor (b) is met, the IBC-coded block can be directly partitioned without going through the current CU level processing. Please note that in SCM-4.0, IBC and Inter are unified for hardware re-utilization. Namely, IBC mode is treated as a special Inter mode with the reference frame restricted to the current frame and the reference area restricted to the previously-encoded area in the current frame. Therefore, for the "mixed" blocks, i.e., blocks partially coded in Inter and partially coded in IBC, we still treat them as "partitioned block" and will bypass the RDO of the current CU depth.

6.2 Single-Input-Multiple-Output Transcoding

To accommodate heterogeneous end users over different networks (e.g., WiFi, LTE, etc.), as illustrate in Figure 6.6, SC contents are typically generated with multiple copies with different quality levels (e.g., spatial resolution, frame rate, bitrate, etc.) so as to support the adaptive SC video streaming services, in which subscribers could request the most suitable version given the network and device

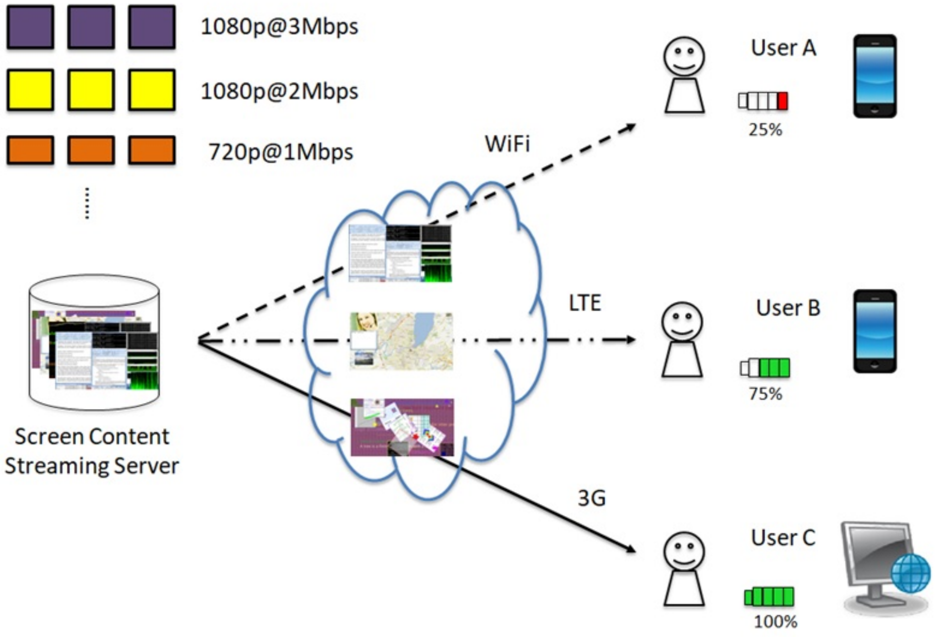


Figure 6.6: Illustration of on-demand SC video streaming

conditions, such as the bandwidth, display resolution, battery life, computing capacity, etc.

Screen content transcoding is one of the most straightforward solutions, where a high-quality bitstream can be utilized to produce multiple bitstreams with reduced quality levels using different combinations of spatial and temporal resolutions and bitrates. The generated bitstreams can be chunked into video fragments to feed in adaptive stream frameworks, such as the HTTP Live Streaming (HLS) [50], the Dynamic Adaptive Streaming over HTTP (DASH) [59], etc.

Intuitively, the overall SC transcoding complexity is roughly $n\times$ the complexity of transcoding a single SC bitstream, where n is the total number of different quality levels required to be achieved at the content server. (Please note n is simply a loose approximation, e.g., the transcoding may introduce spatial and temporal resolution reduction.) Single-Input-Single-Output (SISO) scheme is inefficient since it imposes a significant system complexity, buffer storage, processing delay, and backbone bandwidth (i.e., all SC video copies are likely to be requested by the end users.) As an alternative, leveraging on our previous work [68], we propose an SC Single-Input-Multiple-Output (SIMO) framework, to convert one single high-quality SC video stream into multiple HEVC bitstreams in different qualities. Exploiting the side information from the incoming bitstream and the correlations among the output videos, the re-encoding complexity is significantly reduced, while simultaneously the Rate-Distortion (RD) performances are preserved. Besides the processing complexity, the proposed framework also benefits the system processing delay and potentially would decrease the backbone network traffic from the central SC content server to the edge tower, where the SIMO transcoder is deployed to respond to different user requests.

Different from [68], in this work, SIMO is implemented between two heterogeneous bitstreams (i.e., SCC and HEVC). The SC content characteristics and codec behaviors are taken into account when we customize the SC transcoding algorithms. For simplicity, in this work, we only consider the quality-based SCC-HEVC transcoding, in which a high-quality SC bitstream is transcoded into multiple HEVC bitstreams with reduced qualities. Specifically, according to CTC [75], in our configurations, a high-bitrate SCC bitstream (i.e., coded with QP=22) is transcoded into four HEVC bitstreams (i.e., coded with QP=22, 27, 32, 37, respectively). In [68], a simple yet effective heuristic is used to relate the HEVC depth decisions among coding bitrates, as summarized in Eq. (6.2), where $d(L)$, $d(H)$ and $d(I)$ represents the coding depths of the low bitrate, high bitrate and input bitrate, respectively.

$$d(L) \leq d(H) \leq d(I) \tag{6.2}$$

In our configuration, the same relationship holds among the generated HEVC bitstreams, as shown in 6.3, where $d(QP_{37})$, $d(QP_{32})$, $d(QP_{27})$, $d(QP_{22})$ denote the coding depths for QP=37, 32, 27 and 22, respectively.

$$d(QP_{37}) \leq d(QP_{32}) \leq d(QP_{27}) \leq d(QP_{22}) \tag{6.3}$$

A similar approach as in [68] could be used to accelerate transcoding. Firstly the transcoder decides and caches the coding depth information of QP=22. To encode QP=37 bitstream, the maximum coding depths can be narrowed down based on Eq. (6.3). Finally, to encode QP=27 and QP=32, the coding depth upper bounds, i.e., $d(QP_{22})$, and lower bounds, i.e., $d(QP_{37})$, can be implicitly retrieved from the previous coding decisions from QP=22 and QP=37 bitstreams. Such approach is general and can be used safely regardless of video contents. However, such framework imposes sequential dependencies among generated bitstreams. For example, to compress QP=27 bitstream, depth decisions of both QP=22 and QP=37 are needed. Therefore, this approach is more useful for sequential transcoding and the coding decisions need to be stored for future lookup during transcoding.

In this work, a parallel transcoding scheme is proposed to directly convert SCC bitstream into multiple HEVC bitstreams. Based on our simulation statistics, the SC blocks are relatively insensitive to QP settings. On one hand the partition decisions of SC blocks coded with different QP settings are very similar. On the other hand, a minor SC block partition mismatch will not introduce visible BD-Rate difference. As demonstrated in Figure 6.7, SC videos usually contain

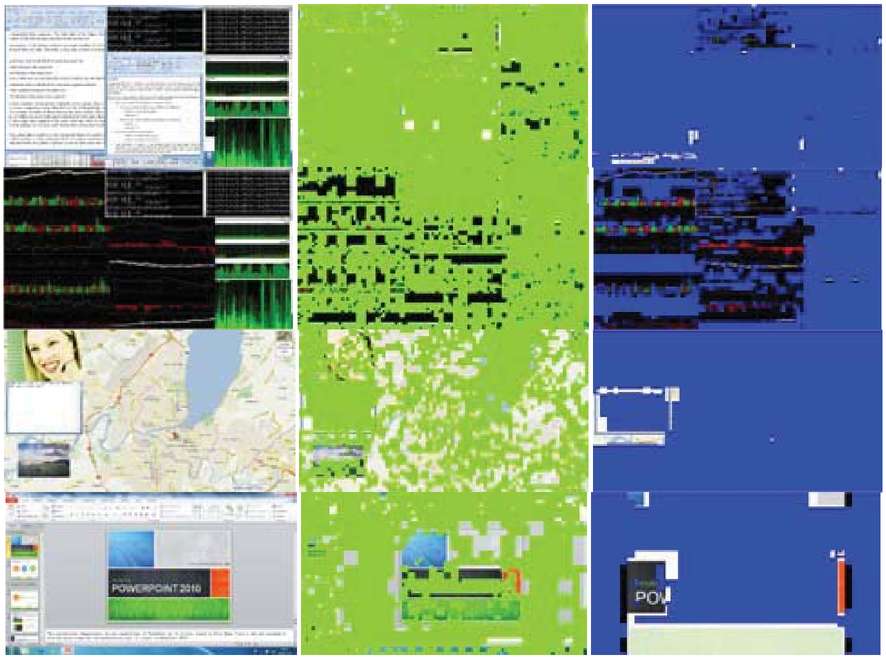


Figure 6.7: HEVC Coding Decision Sensitivity Illustration (QP=22 vs QP=37). Left column: Sample Frames from “Desktop”, “Console”, “Map”, “SlideShow”; Middle column: Intra-frame Sensitivity Map using green blocks indicating consistent mode and partition decisions; Right column: Inter-frame Sensitivity Map using blue blocks indicating consistent mode and partition decisions

a large proportion of “QP-insensitive areas” (QPIA). For Intra-frame coding (in the central column), in “Desktop”, the QPIA percentage is the highest (94.31%). In “SlideShow”, the QPIA percentage is the lowest but still significant (67.06%). Over the “QP-sensitive area” (QPSA), most blocks are coded in Intra-mode. For Inter-frame coding (in the right column), the QPIA percentage varies depending on the contents. The QPIA percentages are 92% and 56% for “Desktop” and “Console” sequences, respectively.

Accordingly, in our SIMO transcoding configuration, the QP=22 HEVC bitstream can be directly transcoded using the default SISO algorithm as illustrated in Figure 6.2 (namely, QP=22 SIMO and SISO cases are identical). For the other bitstreams (i.e., QP=27, 32 and 37), additional mode checking are applied over the partitioned blocks in the SCC bitstream (QP=22), particularly those blocks consisting of sensitive sub-blocks, as illustrated in Figure 6.2 under the SIMO workflow.

For Intra-frame coding, if the current block is mostly coded with SCC-coded sub-blocks, the current block is more likely to be a “QP-insensitive” SC block and can be directly partitioned. Otherwise, if the current block is mostly coded with Intra sub-blocks, this block is more likely a “QP-sensitive” natural image block and therefore goes through the current level Intra mode. For Inter-frame coding, if the current block contains a large proportion of Intra sub-blocks, indicating there is no good temporal matching block available, this block can safely bypass the Inter mode at the current CU depth. If the current block contains a large proportion of Skip or SCC mode, such block is more likely to be a QP-insensitive SC block and therefore can be directly partitioned.

For simplicity, we introduce two tuning parameters α and β, as shown in Figure 6.2 under the SIMO workflow. α and β are pre-defined percentage thresholds used to examine the current block quantization sensitivity for Intra-frame and Inter-frame, respectively. The two parameters can be tuned to trade off the additional mode checking complexity and the coding efficiency. Larger α or β configurations lead to additional Intra or Inter mode-bypass and therefore boost the complexity-saving, but simultaneously degrades the coding efficiency. Smaller α and β configurations lead to additional mode checking at larger block sizes and thus preserve the coding performance better but simultaneously compromise the complexity reduction. In our configuration, α and β are chosen empirically with values of 0.9 and 0.5, respectively.

Besides, our proposed SCC-HEVC transcoding algorithm is self-adaptive. For example, if the matching block region coding depths are adjusted due to the QP change, the current IBC block coding depth and structure will be updated automatically, as demonstrated in Figure 6.4 and Figure 6.5.

6.3 Experimental Results

Our proposed SCC-HEVC fast transcoding framework is evaluated and compared with HM-16.4 anchor software for re-encoding performance evaluation.

For SISO configuration, 9 JCTVC standard SC sequences are coded using SCM-4.0 following the CTC [75], with 4 QPs (i.e., 22, 27, 32 and 37). At the transcoder, the SCC bitstreams are decoded into individual YUV videos with side information cached. Finally, our proposed transcoder will load the cached side information and re-encode the decoded videos into HEVC bitstreams in the same QP for All-Intra (AI) and Low-Delay (LD) configurations.

For SIMO configuration, 9 JCTVC standard SC sequences are coded using SCM-4.0 with QP=22. At the transcoder, the SCC bitstream is decoded into a YUV video (corresponding to QP=22) with side information cached. Finally, our proposed transcoder will load the side information from the decoded SCC bitstream and re-encode the decoded videos (corresponding to QP=22) into HEVC using QP=22, QP=27, QP=32 and QP=37, respectively, for AI and LD configurations.

The coding performances are evaluated using homogeneous Windows 7 (64-bit) desktops with Intel-i5 CPU (2.67 GHz dual cores) and 4GB RAM. The coding efficiency is measured using BD-Rate [2]. The complexity saving is measured directly using the relative reduction of the re-encoding times, as defined in (6.4), where T_{Anchor} is the re-encoding time using HM-16.4 encoder and $T_{Proposed}$ is the re-encoding time of our proposed framework.

$$\Delta C = \frac{T_{Anchor} - T_{Proposed}}{T_{Anchor}} \times 100\% \tag{6.4}$$

Compared with HM-16.4 anchor re-encoding, our proposed fast transcoding framework can achieve complexity reductions of 51% and 49% on average for AI re-encoding and 82% and 76% on average for LD re-encoding using SISO and SIMO frameworks, respectively. Given the space limitation, only the YUV-444 results are provided. However, the proposed framework can be easily generalized to RGB-4:4:4 color space and YUV-4:2:0 sampling format. The All-Intra (AI) transcoding performances are summarized in Table 6.1 and Table 6.2 for SISO and SIMO configurations, respectively. The Low-Delay (LD) transcoding performances are summarized in Table 6.3 and Table 6.4 for SISO and SIMO configurations, respectively. Based on the experimental results, the following conclusions are drawn:

Firstly, the proposed fast SCC-HEVC transcoding framework achieves a remarkable transcoding speedup. Please note that the proposed framework is purely software-based and therefore can be further improved with hardware acceleration. Besides, in this work, only the basic framework is presented without specifically optimizing each individual encoding module. Therefore, other fast video encoding or transcoding algorithms can be easily incorporated into our framework for an additional speedup.

Secondly, the Intra-frame coding acceleration mainly comes from BV-based fast mode and partition reuse over the previously Intra-coded blocks. Besides, the fast directional Intra mode selection over the PLT-coded blocks also provides a visible speedup (i.e., ranging from 1%-5%). The speedup ratio also depends on

Table 6.1: Fast SCC-HEVC Transcoding Framework Performance (AI-SISO)

Sequence	Anchor		Proposed		Performance	
	Rate	PSNR	Rate	PSNR	ΔR	ΔT
Desktop	187871	49.40	188545	49.35	+0.62%	-44%
	153474	44.69	154063	44.63		
	124247	39.56	124647	39.51		
	89900	34.30	90135	34.24		
Console	92030	50.64	92559	50.54	+1.03%	-50%
	75228	45.68	75652	45.62		
	60140	40.60	60503	40.53		
	44254	34.98	44502	34.74		
WebBrowsing	31361	50.71	31394	50.72	+0.20%	-51%
	24598	46.14	24626	46.13		
	17604	42.17	17632	42.19		
	9382	36.74	9405	36.70		
Map	68215	46.76	68324	46.78	+0.42%	-48%
	44655	42.41	44797	42.43		
	27596	38.92	27811	38.92		
	16962	35.90	17196	35.90		
Programming	62167	48.70	62379	48.66	+0.72%	-51%
	42291	44.62	42454	44.58		
	28621	40.05	28706	40.01		
	19591	35.77	19713	35.74		
SlideShow	4388	54.49	4412	54.50	+1.17%	-69%
	2903	50.43	2928	50.43		
	2005	46.21	2025	46.19		
	1370	41.94	1389	41.78		
BasketballScreen	212370	48.86	212897	48.86	+0.44%	-52%
	150110	44.83	150586	44.83		
	102654	40.71	103023	40.70		
	63978	36.55	64275	36.52		
MissionControlClip2	221700	50.18	222303	50.09	+0.43%	-53%
	164565	45.23	164903	45.21		
	114858	40.68	115189	40.67		
	70381	36.11	70635	36.10		
MissionControlClip3	165880	49.06	166182	49.05	+0.19%	-48%
	123646	44.27	124060	44.30		
	87469	39.64	87762	39.66		
	55290	35.22	55560	35.24		

Table 6.2: Fast SCC-HEVC Transcoding Framework Performance (AI-SIMO)

Sequence	Anchor		Proposed		Performance	
	Rate	PSNR	Rate	PSNR	ΔR	ΔT
Desktop	187871	49.40	188545	49.35	+0.64%	-45%
	153523	44.57	154108	44.52		
	124748	39.48	125202	39.43		
	90918	34.31	91230	34.24		
Console	92030	50.64	92559	50.54	+1.08%	-50%
	75239	45.64	75692	45.60		
	59814	40.56	60212	40.49		
	44349	34.51	44641	34.20		
WebBrowsing	31361	50.71	31394	50.72	+0.55%	-51%
	24637	45.18	24669	45.11		
	17684	41.15	17714	41.10		
	9516	35.75	9556	35.72		
Map	68215	46.76	68324	46.78	+1.03%	-43%
	45064	40.88	45311	40.88		
	28397	36.42	28770	36.40		
	17608	32.79	18076	32.71		
Programming	62167	48.70	62379	48.66	+1.00%	-48%
	43065	43.73	43255	43.67		
	29224	39.11	29419	39.06		
	19921	34.76	20157	34.72		
SlideShow	4388	54.49	4412	54.50	+1.26%	-67%
	2915	48.18	2936	48.15		
	2025	44.19	2052	44.15		
	1376	40.07	1409	40.04		
BasketballScreen	212370	48.86	212897	48.86	+0.68%	-47%
	151330	43.20	151976	43.17		
	104209	38.94	104736	38.91		
	66348	34.82	66979	34.79		
MissionControlClip2	221700	50.18	222303	50.09	+0.60%	-48%
	165079	43.76	165562	43.72		
	115740	39.34	116261	39.33		
	72000	34.91	72567	34.89		
MissionControlClip3	165880	49.06	166182	49.05	+0.43%	-45%
	124103	43.02	124638	43.02		
	87902	38.43	88358	38.43		
	55861	33.92	56320	33.95		

Table 6.3: Fast SCC-HEVC Transcoding Framework Performance (LD-SISO)

Sequence	Anchor		Proposed		Performance	
	Rate	PSNR	Rate	PSNR	ΔR	ΔT
Desktop	3956	50.28	3274	50.31	-18.16%	-81%
	3485	45.81	2869	45.81		
	3088	40.75	2516	40.80		
	2566	35.22	2095	35.31		
Console	9970	50.53	9005	50.36	-12.14%	-79%
	8085	45.51	7143	45.42		
	6392	39.83	5575	39.99		
	4620	34.73	4028	34.79		
WebBrowsing	756	50.19	396	50.44	-47.81%	-84%
	599	45.37	313	45.66		
	411	40.38	226	41.03		
	238	35.23	144	35.90		
Map	2508	45.82	2505	45.78	-0.23%	-82%
	1563	41.66	1564	41.62		
	922	38.25	921	38.30		
	535	35.30	536	35.43		
Programming	6813	48.04	6793	48.03	-1.02%	-83%
	3624	43.79	3618	43.84		
	1713	39.54	1713	39.67		
	805	35.61	817	35.58		
SlideShow	845	51.57	863	51.70	+1.36%	-82%
	488	47.64	502	47.79		
	286	43.63	294	43.69		
	172	39.53	177	39.59		
BasketballScreen	7220	48.92	7339	48.61	+1.96%	-82%
	3906	45.25	3930	45.25		
	2319	41.32	2338	41.31		
	1388	37.05	1427	36.90		
MissionControlClip2	3813	50.21	3828	50.11	+0.93%	-83%
	2545	45.69	2550	45.68		
	1700	41.55	1707	41.46		
	1048	37.02	1052	36.96		
MissionControlClip3	3268	48.25	2955	48.33	-12.76%	-82%
	2123	44.16	1876	44.24		
	1401	39.71	1219	39.79		
	890	35.18	779	35.25		

Table 6.4: Fast SCC-HEVC Transcoding Framework Performance (LD-SIMO)

Sequence	Anchor		Proposed		Performance	
	Rate	PSNR	Rate	PSNR	ΔR	ΔT
Desktop	3956	50.28	3274	50.31	-18.13%	-78%
	3482	45.67	2858	45.66		
	3084	40.65	2512	40.67		
	2561	35.25	2100	35.28		
Console	9970	50.53	9005	50.36	-10.23%	-72%
	7942	45.37	6992	44.97		
	6265	39.63	5499	39.23		
	4593	34.25	4016	33.81		
WebBrowsing	756	50.19	396	50.44	-47.90%	-83%
	590	44.59	303	44.69		
	409	39.72	212	39.83		
	240	34.68	138	34.52		
Map	2508	45.82	2505	45.78	+1.99%	-74%
	1589	40.06	1593	39.94		
	953	35.54	964	35.41		
	565	31.97	578	31.79		
Programming	6813	48.04	6793	48.03	+2.69%	-72%
	3800	42.85	3830	42.73		
	1846	38.22	1894	38.18		
	873	33.98	907	33.89		
SlideShow	845	51.57	853	51.70	+3.07%	-73%
	504	46.11	513	46.05		
	293	41.85	298	41.70		
	179	37.77	189	37.52		
BasketballScreen	7220	48.92	7339	48.61	+3.34%	-76%
	4059	43.34	4140	43.28		
	2405	39.22	2467	39.17		
	1452	35.13	1515	35.11		
MissionControlClip2	3813	50.21	3828	50.11	+1.93%	-81%
	2583	44.08	2615	44.04		
	1728	39.91	1765	39.89		
	1074	35.71	1110	35.67		
MissionControlClip3	3268	48.25	2955	48.33	-7.09%	-78%
	2156	42.93	1959	42.82		
	1393	38.50	1300	38.33		
	906	33.99	841	33.76		

the contents. For sequences mainly coded in larger blocks, e.g., “SlideShow”, the complexity reduction is more (≈70%), whereas over sequences mainly coded using smaller blocks, e.g., “Map”, the complexity reduction is less (≈50%).

Thirdly, the Inter-frame coding acceleration mainly comes from the MV reuse and the CU fast bypass/termination. Since SCC uses full-frame hash-based IBC search [15] over CU8 blocks and Inter-frame hash-based motion search, the inherited MVs from SCC bitstream sometimes outperform the local ME results in the anchor HEVC configuration (with a default search range of 64 over the AMVP candidates). Therefore, over particular sequences (e.g., “WebBrowsing”), we observe a significant BD-Rate gain even after the transcoding acceleration. Besides, the motion vector relay technique applied over IBC blocks provides a significant speedup (i.e., ranging from 6%-18%), depending on the IBC utilization in the Inter-frames.

Fourthly, the proposed framework outperforms anchor HEVC Inter-frame coding significantly over the screen content areas with dominant temporal changes (e.g., scene-cut). As shown in Table 6.4, for “WebBrowsing” sequence, a 48% BD-Rate saving is achieved after transcoding speedup. This significant gain is achieved from several transition frames, in which the inherited MVs from SCC bitstreams significantly outperform the MVs derived from HEVC restricted motion search. As illustrated in Figure 6.8, during the content transitions (e.g., from 8th frame to 9th frame) in the webpage bottom panel (as enclosed inside the red square), the Inter-frame mode and partition distribution have changed drastically. Since SCM uses hash-based IBC and Inter search, therefore the inherited motion vectors are more accurate than anchor HEVC ME candidates. Consequently, more blocks are skip-coded or merge-coded and the bit-consumption is much lower (as shown clearly from the heatmap). For this exemplar frame, anchor HEVC spends 566344 bits while our proposed transcoding algorithm only spends 164272 bits. Similar behaviors are observed over “Desktop”, “Console”, etc.

Fifthly, for “Mixed-Content” Inter-frame coding, e.g., “MissionControlClip2”, “MissionControlClip3” sequences, depending on the content temporal variation locations, different behaviors are observed. As illustrated in Figure 6.9, though the two SC sequences appear similar, in “MissionControlClip2”, mostly natural video region (e.g., the man enclosed in the red box) has temporal motion, while the text region on the right side is relatively static. In such case, the natural video region is dominated by Intra mode and Inter mode with local search. Therefore, after transcoding, we do not observe any BD-Rate saving. This also applies for the “BasketballScreen”, in which the temporal motion is mainly located in the natural video regions (e.g., the basketball player window). In “MissionControlClip3”, a large proportion of text regions have temporal motion instead. In such case, the inherited motion vectors from SCC bitstream mostly outperform the anchor HEVC local ME and therefore leads to a coding efficiency improvement. To conclude, when transcoding the SC regions (e.g., text, graphics, icon, etc.), the proposed MV inheritance and MV relay schemes using side information from SCC bitstream can outperform the anchor HEVC motion estimation results, whereas when transcoding the natural video regions (e.g., natural picture), the derived MV does not differ much from the anchor HEVC motion derivation and therefore does not introduce BD-Rate saving.

Finally, the proposed parallel screen content SIMO transcoding framework significantly reduces the transcoding complexity, calculated using the sum of the re-encoding times for 4 QP settings between the anchor HEVC encoder and our

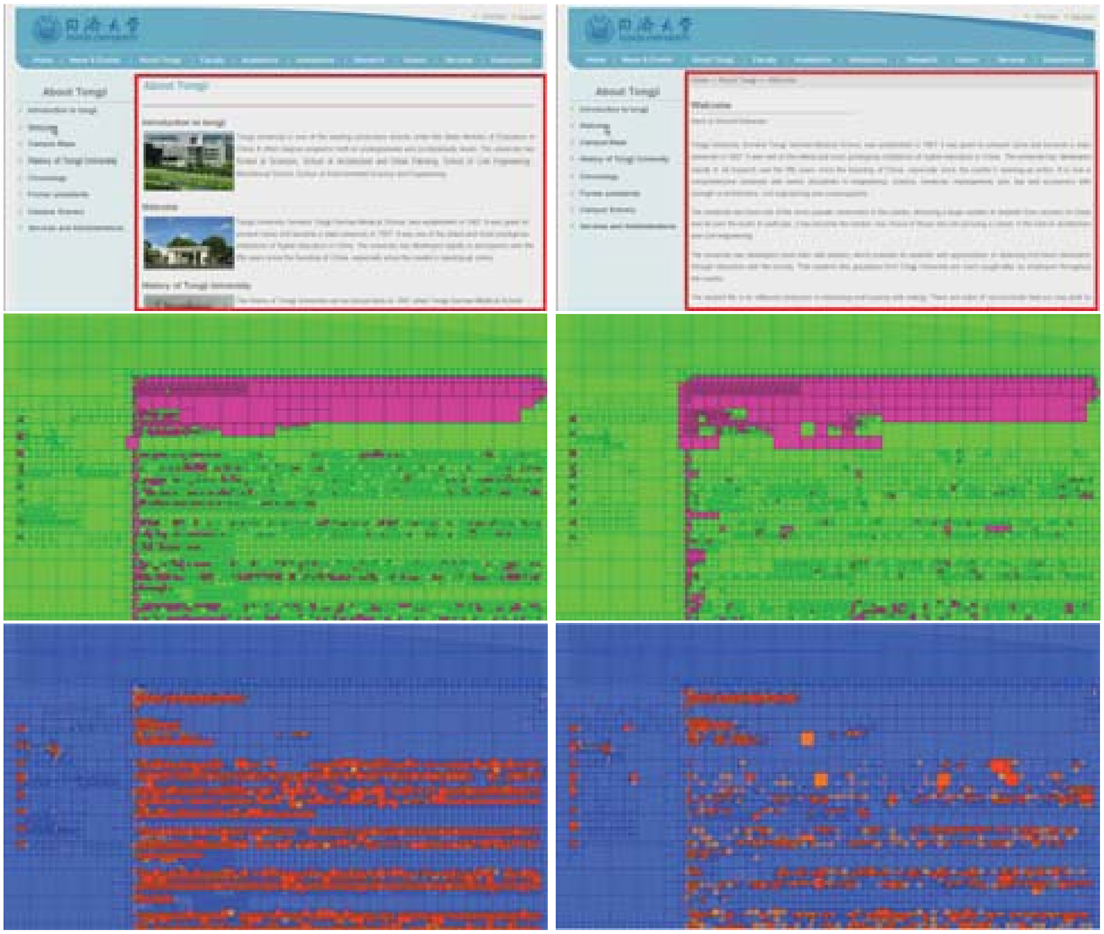


Figure 6.8: Transcoding Analysis of “WebBrowsing” (QP=22). Top Left: 8th Frame; Top Right: 9th Frame; Mid Left: Anchor Transcoding Mode and Partition Decisions; Mid Right: Proposed Transcoding Mode and Partition Decisions; Bottom Left: Anchor Transcoding Bit-Allocation. Bottom Right: Proposed Transcoding Bit-Allocation. Red box: Intra-coded blocks; Green box: Inter-coded blocks; Blue box: low-bit blocks; Orange box: high-bit blocks using color depth to indicate bit-consumption level.

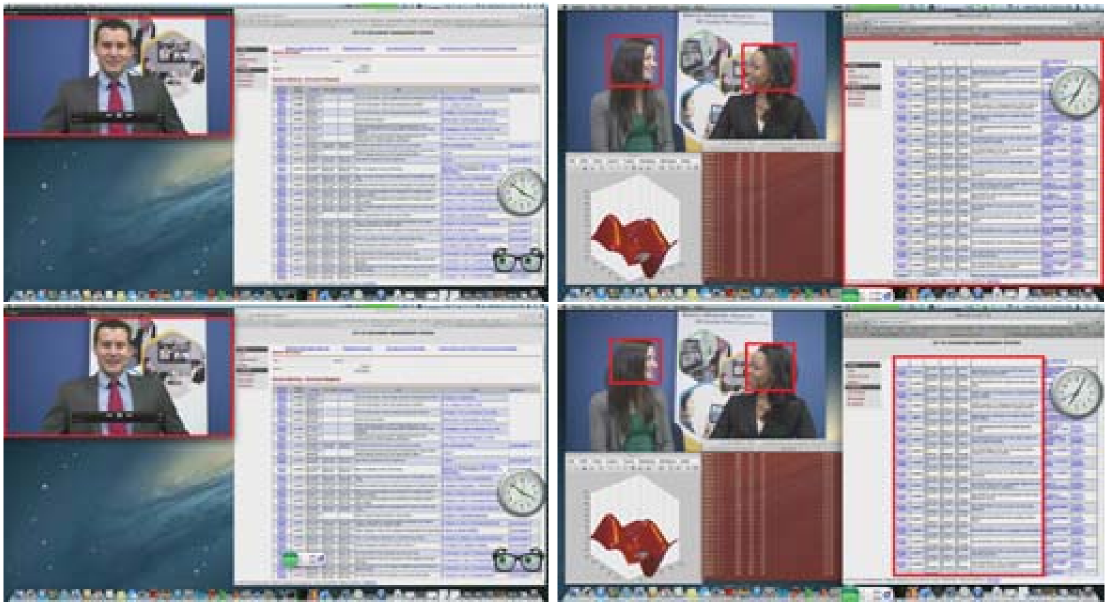


Figure 6.9: Sample frames in “MissionControlClip2” (left) and “MissionControlClip3” (right). Top and bottom: 1st and 11th frame, respectively. Red boxes indicate the regions with dominant temporal motions.

proposed framework. Even though the complexity saving is less than the SISO case due to the additional mode checking over the partitioned blocks in the SCC-bitstream, the margin is relatively small (i.e., within 6%). For the sequences dominated by temporal motion over SC region, e.g., “Desktop”, “Console”, the proposed framework preserves the coding performance better. For the sequences dominated by temporal motions over the natural video regions, the proposed framework introduces larger but still marginal BD-Rate losses, e.g., “SlideShow”, “Programming”, “BasketballScreen”, etc. Please note that the previous fast transcoding heuristic proposed in [68] can be incorporated into our SIMO framework for an improved coding performance when sequential transcoding is allowed. More details and additional simulation results are provided in [21] and [19]

Chapter 7

Two-Tier 360 Video Streaming with Prioritized Buffer Control

7.1 Overview of Two-Tier 360 Video Streaming System

In this section, a novel two-tier 360 degree video streaming framework is proposed, as illustrated in Figure 7.1. In this framework, a 360-degree video is partitioned into non-overlapping time segments, and each segment is encoded into a base-tier (BT) chunk and multiple enhancement-tier (ET) chunks. A BT chunk encodes the entire 360 view span ($360^\circ \times 180^\circ$) at a low bitrate to provide the basic quality. BT chunks for future time segments are pre-fetched in a long display buffer to cope with network bandwidth variations and user view changes and guarantee that any desired FOV can be rendered with minimum stalls at the client. Each ET chunk encodes 360-degree video within a view window with a certain view coverage (VC) (e.g., $120^\circ \times 90^\circ$) centered at a certain direction. To provide

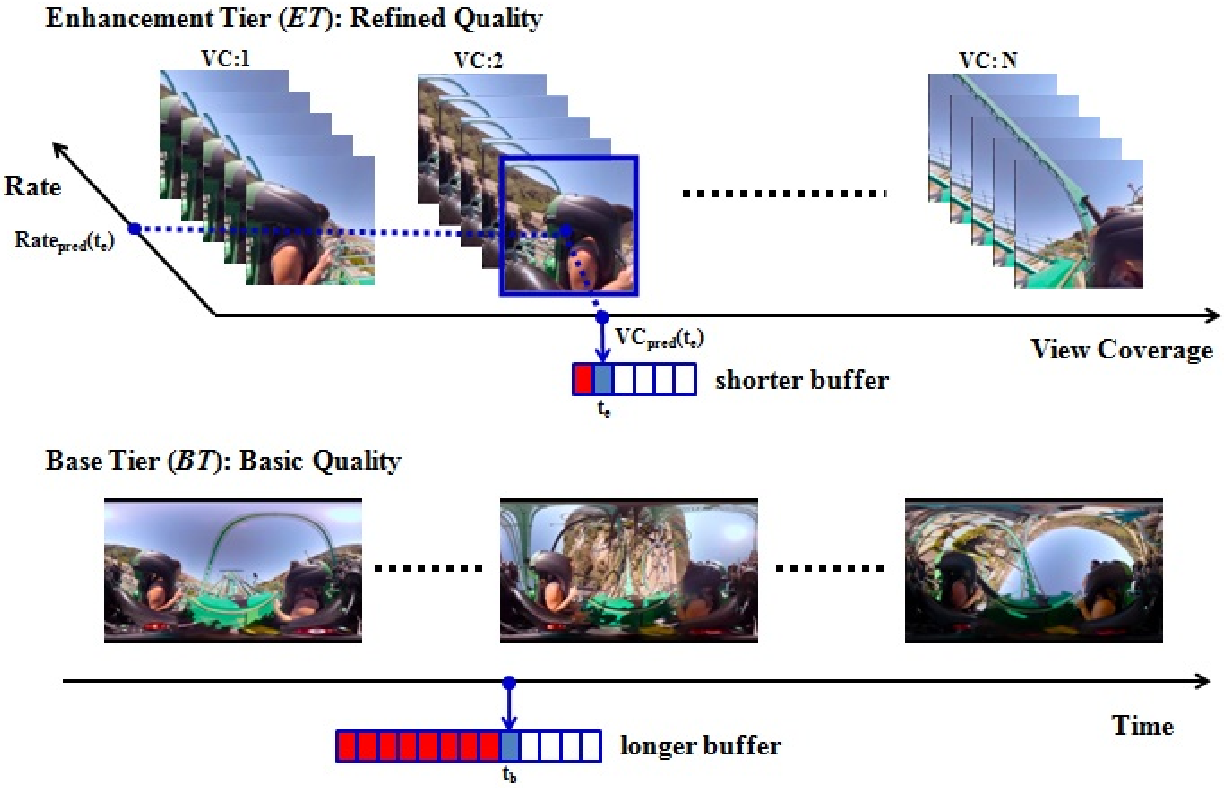

Enhancement Tier (ET): Refined Quality
VC:1
VC:2
VC:N
Rate
$\text{Rate}_{pred}(t_e)$
$VC_{pred}(t_e)$
View Coverage
shorter buffer
t_e
Base Tier (BT): Basic Quality
Time
longer buffer
t_b

Figure 7.1: Two-tier 360 Video Streaming System

the quality differentiation, multiple ET chunks can be generated for the same view window, but coded at different bitrates. For a complete coverage and a smooth transition, the view windows of ET chunks in the same time segment are *overlapping* and cover the whole 360 view span. All the pre-coded chunks are stored in the streaming server. The client will decide and request a particular rate version from a particular tier, according to the predicted view direction for the segment, the predicted download bandwidth in the next request interval, and the buffer status of each tier.

In our current implementation, the BT chunks are encoded at a basic rate, which is expected to be sustainable even when the network bandwidth is low.

Therefore, the bandwidth wasted on covering scenes outside of the user FOV is limited. Since a BT chunk encodes the entire 360-degree scene, it is always useful for rendering no matter how dynamically a user changes her view direction during the streaming session. In the extreme case of aggressive prefetching, as long as the average network bandwidth is above the average coding rate of the BT, the end user is guaranteed to receive a continuous 360 video experience with basic quality, regardless of how instantaneous network bandwidth varies, and regardless of how abruptly or frequently the user changes her view direction.

The ET chunks are designed to improve the streamed video quality whenever there is additional bandwidth available after the base tier chunks are delivered. Since both future available bandwidth and user viewing direction are generally unknown, ET chunks will be prefetched in an *opportunistic* fashion, with the help of available bandwidth estimation and view direction prediction. Specifically, at time t, when additional bandwidth is available, one can prefetch an ET chunk for $t + \Delta$ covering the predicted view window at that time. As studied in [56], the long-term head motion prediction is very difficult. Therefore, the ET chunks in our system are prefetched in a relatively shallow buffer (e.g., up to 5 second ahead) so that the delivered view window mostly coincides with the actual user FOV. When user view directions are predicted accurately and ET chunks are received successfully, the client video player can combine the ET chunk with the prefetched BT chunk for an enhanced quality. Even when the view prediction fails (e.g., due to unexpected head motion) or when the requested ET chunk does not arrive promptly before its deadline (e.g., due to sudden bandwidth decrease), the client can still render the desired view with basic quality from the prefetched BT.

7.2 Prioritized Buffer Control Based 360-degree Video Streaming

We formulate two-tier 360 video streaming as a dynamic scheduling problem. Similar to the Dynamic Adaptive Streaming over HTTP (DASH) [58] framework, we consider the scheduling as a discrete time process. At each time slot, a client strategically prefetches video chunks from both tiers based on view direction and available bandwidth predictions. The goal is to maximize the rendered video quality of the streaming session, while both the network bandwidth and user view direction may vary over time. We propose to set up strict priority between multiple design objectives and develop scheduling algorithms to achieve the desired priority. In our current design, we believe that playback continuity is the most important and therefore we give the highest priority to download the base tier chunks in near future so that we can always render a basic-quality version even if the high-quality version is not available at the desired FOV, either due to view prediction error or enhancement-tier buffer underflow. Similar to many DASH work, we measure the buffer length using the buffered video time. If the current base-tier buffer length is less than the target buffer length q^b_{ref}, one should always sequentially download the BT chunks until the BT buffer reaches q^b_{ref}.

After downloading enough BT chunks, the residual bandwidth will be used to download the ET chunks. In this work, we formulate the ET chunk scheduling problem as a buffer-based feedback control problem, leveraging on the previous study for buffer-based DASH [64]. Let $q^e(t)$ be the buffered video time for enhancement tier at time t. When a chunk k with future playback time is prefetched, the evolution of $q^e(t)$ can be approximated by the fluid model provided in Eq. (7.1), where $\mathbf{1}(\cdot)$ is the indicator function, and $\bar{b}(k)$ is the average bandwidth when downloading chunk k, τ is the video duration for each chunk, $s^e(k)$ is the size of chunk k, and $t_k^{(s)}$, $t_k^{(f)}$ is the starting and finishing time of downloading chunk k.

$$\frac{d}{dt}q^e(t) = \frac{\bar{b}(k)\tau}{s^e(k)} - \mathbf{1}(q^e(t) > 0), \quad t \in (t_k^{(s)}, t_k^{(f)}] \tag{7.1}$$

One can select the request rate version of an ET chunk by setting up a target buffer length q^e_{ref} for the ET. If the current buffer length is less than q^e_{ref}, one should be conservative and choose a chunk with size $s^e(k) < \hat{b}(k)\tau$, the estimated bandwidth budget, such that more video time can be accumulated; if the current buffer length is greater than q^e_{ref}, one can be more aggressive and choose a chunk with size $s^e(k) > \hat{b}(k)\tau$, so that the accumulated video time can be reduced to q^e_{ref}. As shown in [64], traditional feedback control algorithms, such as Proportional-Integral (PI) controllers, can maintain the target buffer length very well. The target rate $\hat{R}\,(k)$ of ET chunk k can be determined based on buffer evolution as

$$u(k) = K_P(q^e(t_k^{(s)}) - q^e_{ref}) + K_I \sum_{t=0}^{t_k^{(s)}} (q^e(t) - q^e_{ref}), \tag{7.2}$$

$$\hat{R}\,(k) = \frac{s^e(k)}{\tau} = min\left[(u(k)+1), \frac{\Delta(k)}{\tau}\right] \cdot \hat{b}(k), \tag{7.3}$$

where K_P and K_I are the proportional and integration gain control coefficients, respectively, $u(k)$ is the control signal, and $\Delta(k)$ is the remaining time till the display deadline of ET chunk k.

The target buffer length tuple $\langle q^b_{ref}, q^e_{ref} \rangle$ for base and enhancement tiers reflects the trade-off between robustness and quality. A large q^b_{ref} achieves high robustness against variations in both network bandwidth and user view direction changes, but at the cost of reduced likelihood to download the enhancement tier chunks, lowering the rendered video quality. A large q^e_{ref} also achieves high robustness against network bandwidth variation, but is *vulnerable* to user view direction changes, simply because it is more difficult to predict user's view direction into the far future, and a prefetched enhancement-tier chunk is useless if its view coverage does not cover the user's actual FOV for that segment. In this study, we encode the BT chunks at a basic rate expected to be sustainable even at low bandwidths. The bitrates of the ET chunks, on the other hand, are more significant compared to the BT. Target buffer length selection for ET chunks is a more interesting and important challenge in our two-tier streaming framework. Therefore, we present a formulation to determine the target enhancement tier buffer length q^e_{ref}. Ideally, we would like to maximize the rendered video quality. However, since the video quality is generally monotonically increasing with the average *video rendering rate* (in terms of bits per viewing area), we try to maximize the delivered video rate instead. This design obviates the prior dependency knowledge between the video quality and video rate, which is typically content dependent. Besides, it also leads to a simpler solution, because our design parameter (i.e., q^e_{ref}) directly impacts the rate.

Because the base-tier buffer length in our system is long, we assume that the base-tier chunks are mostly delivered in time for display, so that for each video segment, we either receive only the base-tier or both the base-tier and the enhancement-tier chunks. The base-tier chunks are coded to cover the entire area of 360 video with the total rate of R_b (in bits/second) and therefore the video rendering rate is R_b/A_b, where A_b is the viewing area of the 360 video. Let $\bar{R}_e$ and A_e denote the average enhancement-tier rate and the coverage area of each ET chunk, respectively. Since that the predicted view direction for a delivered video segment may not be the same as the actual user viewing direction, therefore, not all received chunks for the enhancement-tier are useful. In general, only a portion of each decoded frame in the delivered chunk may overlap with the user's FOV for that frame. Here we introduce α to denote the average ET View Prediction Accuracy (VPA), namely the average overlapping ratio between the predicted view coverage and user's actual FOV, and γ to denote the average ET Chunk Pass Rate (CPR), namely the likelihood that a requested ET chunk can be delivered successfully before its display deadline. Therefore, the expected Video Rendering Rate (VRR) can be expressed as

$$\begin{aligned} R_{VRR}(q^e_{ref}) &= \gamma \cdot (\alpha \cdot (\frac{R_b}{A_b} + \frac{\bar{R}_e}{A_e}) + (1-\alpha) \cdot \frac{R_b}{A_b}) + (1-\gamma) \cdot \frac{R_b}{A_b} \\ &= \frac{R_b}{A_b} + \alpha\gamma \cdot \frac{\bar{R}_e}{A_e}, \end{aligned} \tag{7.4}$$

where α and γ are both functions of q^e_{ref}. Intuitively, α decreases as q^e_{ref} increases because the view prediction into far future becomes less reliable. γ increases as q^e_{ref} increases because a longer ET buffer is more likely to absorb the temporary mismatch between the real network throughput and predicted bandwidth. Obviously, there is an intrinsic trade-off between increasing $\alpha(q^e_{ref})$ (i.e., view prediction accuracy) and increasing $\gamma(q^e_{ref})$ (i.e., ET chunk pass rate) when selecting q^e_{ref}. Assuming that one can predict the bandwidth for each chunk accurately, and that there are sufficient rate versions for the same time segment to match the available bandwidth, then we can approximate $\bar{R}_e = \bar{T} - R_b$, where $\bar{T}$ is the average available bandwidth. Eq. (7.4) implies that we should select the q^e_{ref} that maximizes the product of $\alpha(q^e_{ref})\gamma(q^e_{ref})$ to maximize the delivered video rate. In our implementation, we determine $\alpha(q^e_{ref})$ and $\gamma(q^e_{ref})$ experimentally by simulating our system using different q^e_{ref} with a variety of network traces and view traces. Please note that γ is determined by both network statistics and also video rate versions provided on the streaming server. In our system, four different rate versions are provided and used for simulations. α is mainly determined by view prediction methodologies and the number and size of VCs in the enhancement-tier.

7.3 Enhancement Tier Target Buffer Length Optimization

The q^e_{ref} in the proposed system is optimized off-line over the collected view and network traces. The optimal operation point is jointly determined by the average VPA (i.e., α) and the average CPR (i.e., γ). To illustrate, the VPA over our concatenated view trace is plotted in Figure 7.2 (red curve). For simplicity, we use the past 30 samples to predict future chunk viewing direction via a linear model, i.e., $y = At + b$, where t are sample timestamps and y are the corresponding viewing angles. The linear coefficients (i.e., A and b) are chosen to minimize the sum of the least-square error between the ground-true and predicted user viewing angles. Then we apply the derived linear model to predict the viewing angle for the incoming ET chunk at $t + \delta$, i.e., $\hat{y} = A(t + \delta) + b$, where δ is a positive time offset. The average CPR under our combined network trace is provided in Figure 7.2 (blue curve). As shown from Figure 7.2, the optimal operation point for our view trace and network trace is at $q^e_{ref} = 1$, corresponding to the peak on the purple curve. According to our formulation, the q^e_{ref} that maximizes the product of α and γ will maximize the VRR.

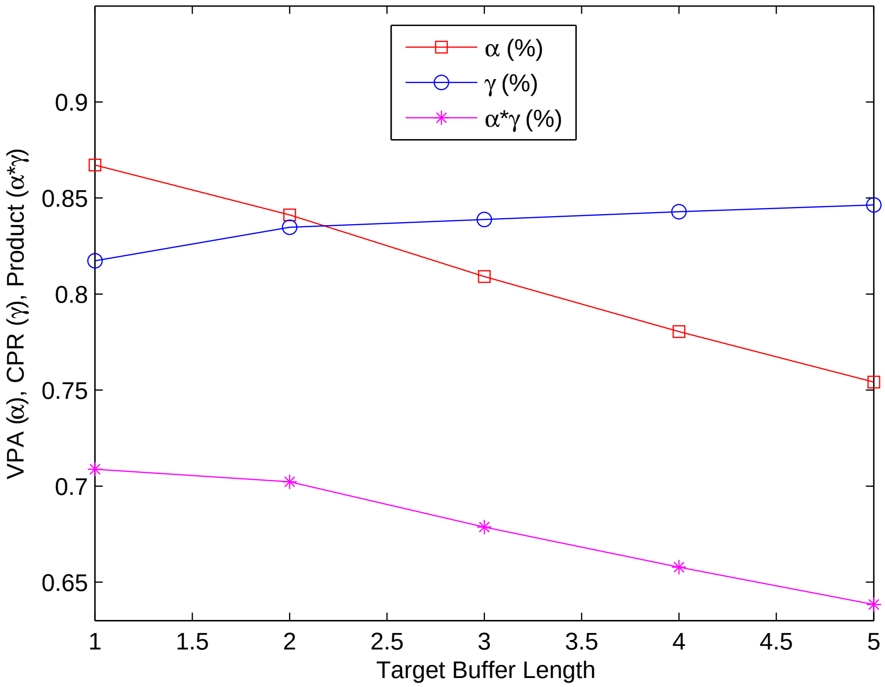


Figure 7.2: Target Buffer Length Selection. Red curve: average view prediction accuracy (α). Blue curve: average chunk pass rate (γ). Purple curve: product of α and γ. The optimal operation point locates at the peak of purple curve.

7.4 360 Video Inter-Tier Rate Allocation

One critical issue in the two-tier system is how to allocate the rates between the two tiers given the total video rate R_t (bits/second). One way to solve this problem is by maximizing the expected quality for a video segment. We provide a high-level formulation based on some simplistic assumptions. Because the base-tier buffer length in our system is long, we assume that the base-tier chunks are mostly delivered in time for display, so that for each video segment, we either receive only the base-tier or both the base-tier and the enhancement-tier chunks. The base-tier chunks are coded to cover the entire area of 360 video with the total rate of R_b (in bits/second) and therefore the video rendering rate is $\tilde{R}_b = R_b/A_b$ (bits/pixel), where A_b is the viewing area of the 360 video. Let R_e and A_e denote the average enhancement-tier rate and the coverage area of each ET chunk, respectively. Let us assume that the ET video is coded with layer coding based on the base-tier decoded video, so that the pixel bit rate for an enhancement tier coded pixel is $\tilde{R}_e = R_b/A_b + R_e/A_e$. Since that the predicted view direction for a delivered video segment may not be the same as the actual user viewing direction, therefore, not all received chunks for the enhancement-tier are useful. In general, only a portion of each decoded frame in the delivered chunk may overlap with the user's FOV for that frame. The perceived quality only depends on what is being rendered on the screen based on the user's FOV. The probability that a rendered pixel is covered by the delivered ET chunk is $\alpha \cdot \gamma$. Assuming the BT and the ET coders can be characterized by their respective rate-quality functions $Q_b(\tilde{R})$ and $Q_e(\tilde{R})$, where $\tilde{R}$ is the coding bits per pixel, the expected rendered video quality with the constraint $R_b + R_e = R_t$ can be expressed as

$$\begin{aligned} Q(R_b; \alpha, \gamma, R_t) &= \alpha\gamma Q_e(\tilde{R}_e) + (1-\alpha\gamma)Q_b(\tilde{R}_b) \\ &= \alpha\gamma Q_e\left(\frac{R_b}{A_b} + \frac{R_t - R_b}{A_e}\right) + (1-\alpha)Q_b\left(\frac{R_b}{A_b}\right) \end{aligned} \tag{7.5}$$

Note that α and γ are both dependent on the ET prefetching buffer length. Generally, γ also depends on R_e. Here we assume that we can estimate the average network bandwidth fairly accurately and we set R_t below the estimated bandwidth with a safe margin, so that average γ does not depend on R_e or R_b. Therefore, for a given prefetching buffer length, α and γ can be considered as constants. Therefore, the optimal R_b can be found by setting $\frac{\partial Q}{\partial R_b} = 0$, which yields

$$\left.\frac{\partial Q_e}{\partial R}\right|_{\tilde{R}_e^*} = \left(\frac{1-\alpha\gamma}{\alpha\gamma}\right)\frac{A_e}{A_b - A_e}\left.\frac{\partial Q_b}{\partial R}\right|_{\tilde{R}_b^*} = \beta\left.\frac{\partial Q_b}{\partial R}\right|_{\tilde{R}_b^*} \tag{7.6}$$

Equation (7.6) implies that R_b and R_e should be chosen such that the Quality-Rate (Q-R) slope at $\tilde{R}_e$ should be β times the slope at $\tilde{R}_b$. Figure 7.3 demonstrates the optimal $\tilde{R}_b^*$ and $\tilde{R}_e^*$ relations for two different β values for a hypothetical but typical Q-R curve: β_1=1/45 resulting from assuming $\alpha\gamma = 0.9$ and $A_b/A_e = 6$, and $\beta_2 = 3/35$ from assuming $\alpha\gamma = 0.7$. We see that if α and γ are both very close to 1, then β is very small, and the optimal allocation is to let $\tilde{R}_b$ to be very low. This corresponds to the case that view and bandwidth prediction are both very accurate, so that a rendered pixel can almost always be covered by a delivered ET chunk. In this circumstance, it is better not to waste bits to send entire 360 scope in the base tier. When view and/or bandwidth prediction is less accurate (α and/or β is lower), it is better to spend more bits on the base tier, to ensure that pixels that are rendered from BT chunks provide sufficient quality. In practice, we should set $\tilde{R}_b = \max(\tilde{R}_{b,min}, \tilde{R}_b^*)$, to make sure that any FOV region that are not covered by ET chunks due to either view prediction or delivery errors can be rendered with a basic satisfactory quality, with rate $\tilde{R}_{b,min}$. For a chosen encoding method, one can derive or model the operational Q-R relationship, and consequently determine R_b based in Eq. (7.6).

Following JVET common test conditions (CTC) and evaluation procedures for 360° video [1], the equirectangular encoding statistics for 4 Call-for-Evidence (CfE) 8K sequences [71] is used for Q-R model derivation. The rates R are HM-16.15 encoding bitrates (in kbps) with QP values of 22, 27, 32 and 37, respectively. Weighted-to-spherically-uniform peak-signal-to-noise ratio (WS-PSNR) is used as the objective quality metric. For simplicity, the QR relationship is modeled by a

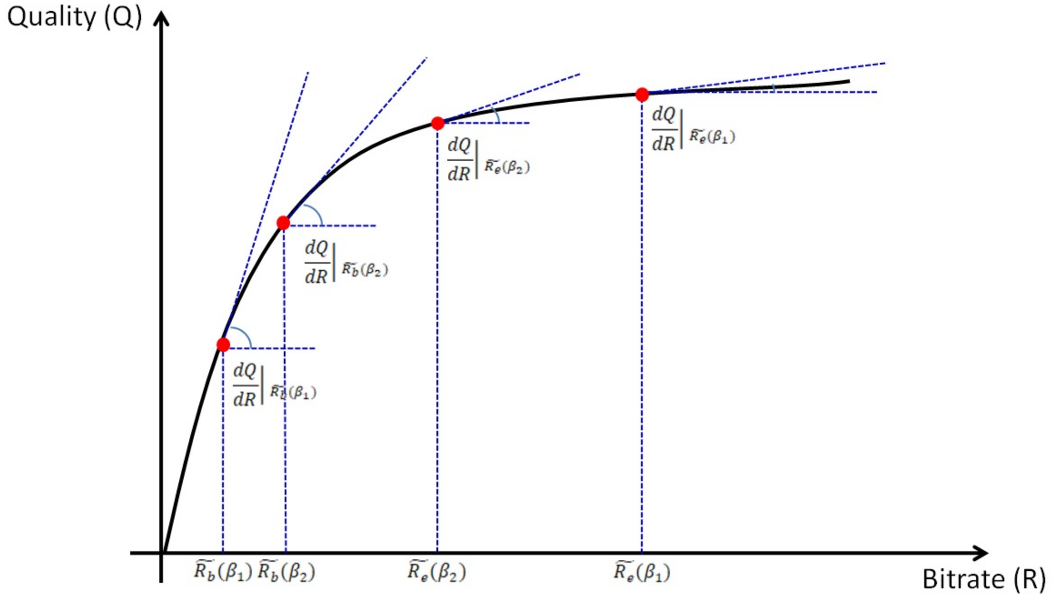

Quality (Q)
$\frac{dQ}{dR}\Big|_{\widetilde{R_e}(\beta_1)}$
$\frac{dQ}{dR}\Big|_{\widetilde{R_e}(\beta_2)}$
$\frac{dQ}{dR}\Big|_{\widetilde{R_b}(\beta_2)}$
$\frac{dQ}{dR}\Big|_{\widetilde{R_b}(\beta_1)}$
$\widetilde{R_b}(\beta_1)$
$\widetilde{R_b}(\beta_2)$
$\widetilde{R_e}(\beta_2)$
$\widetilde{R_e}(\beta_1)$
Bitrate (R)

Figure 7.3: Sample Rate-Distortion Operation Point Analysis

logarithmic function with two free parameters, i.e., $Q(R) = a + b \cdot \log R$, where a and b are content-dependent parameters, and are chosen to minimize the fitting mean-square-error (MSE) in our formulation. The sample R-D fitted curves are provided in Figure 7.4. Experimental results demonstrate that the b values derived from different sequences are pretty close (i.e., ranging from 2.6 to 3.9), indicating a similar curve trend across different 360 videos. For simplicity, we average the a and b values across videos to derive a universal model with $a = 31.74$ and $b = 3.3$. Please note that in our rate allocation formulation, the optimal rate allocation only depends on the sum and ratio between $\tilde{R}_b^*$ and $\tilde{R}_e^*$ and is therefore video-content independent, as long as the video Q-R relationship follows a logarithmic pattern (which is usually true, as validated in Figure 7.4). The closed-form solutions of $\tilde{R}_e^*$ and $\tilde{R}_b^*$ are provided in Eq. (7.7) and Eq. (7.8), where β is defined in Eq. (7.6) and is determined jointly by the view prediction accuracy (i.e., α), the chunk pass rate (i.e., γ) and the BT and ET coverage configurations (i.e., A_b and A_e). η is a network utilization control parameter defined in percentage.

$$\tilde{R}_e^* = \frac{R_t \cdot \eta}{1 + \beta} \tag{7.7}$$

$$\tilde{R}_b^* = \frac{\beta \cdot R_t \cdot \eta}{1 + \beta} \tag{7.8}$$

7.5 360 Video Streaming Experimental Settings

7.5.1 Video Selection

We downloaded two sample 360 videos[1] (1920x3840, 30Hz) from YouTube, as shown in Figure 7.5. For simplicity, we neglect the possible rate fluctuations caused by the content variation and assume the video rate control is perfect such that each ET video chunk is coded with a constant rate. Horizontally the 360 video is divided into 12 View Coverages (VC). Each VC spans 120° with a 30° stride. Vertically the video is divided into 3 VCs. Each VC spans 90° with a 45° stride. Therefore, for each BT chunk, there are totally 36 VCs in the ET to cover different viewing directions. For simplicity, we assume these independently-decodable video chunks are coded with a fixed group of picture (GOP) length (i.e., 1 second). We also assume that the user FOV has the same dimension as our VCs (i.e., 120° × 90°).

[1] Sample videos download links: https://www.youtube.com/watch?v=-xNN-bJQ4vI and https://www.youtube.com/watch?v=FzrkpXlRP1M.

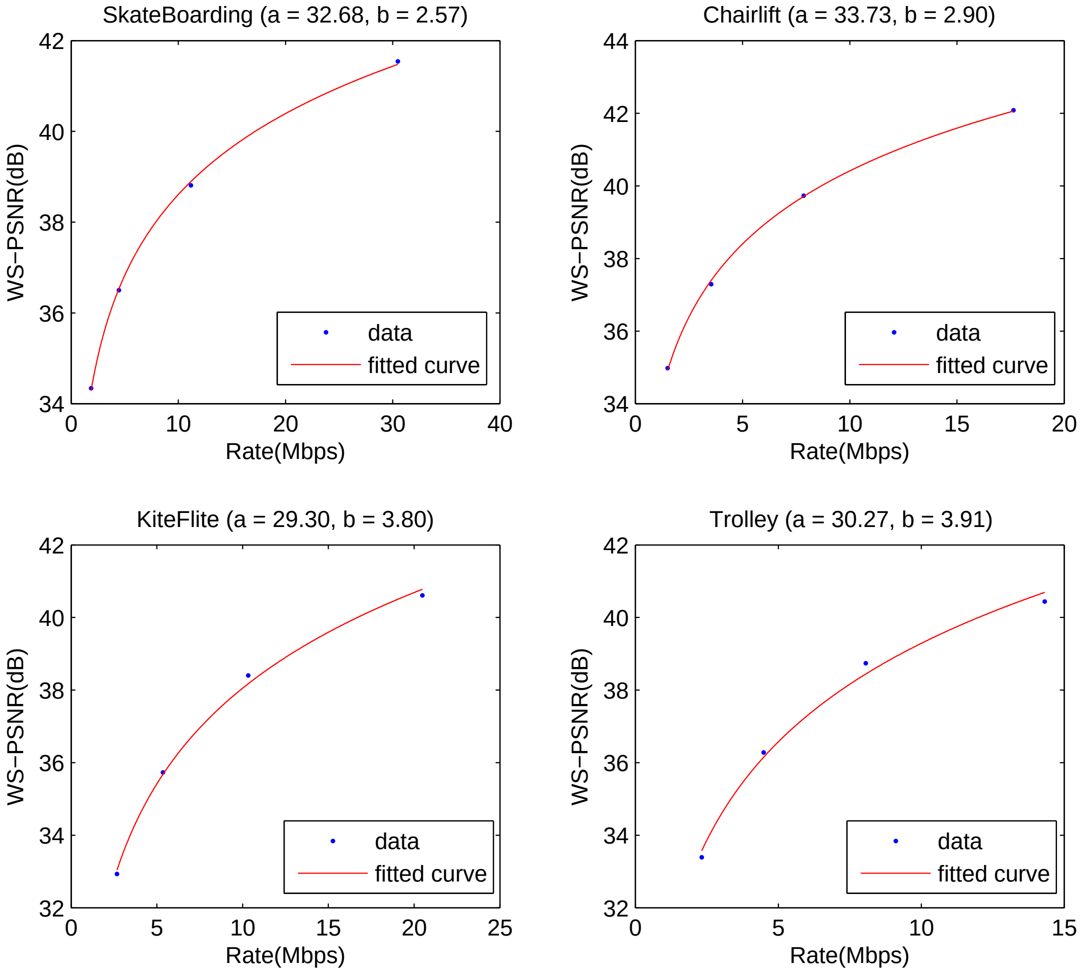


Figure 7.4: JVET CfE 360° Video Rate-Distortion Curves

Figure 7.5: Sample Frames in Test Videos “MegaCoaster”, “Amsterdam”

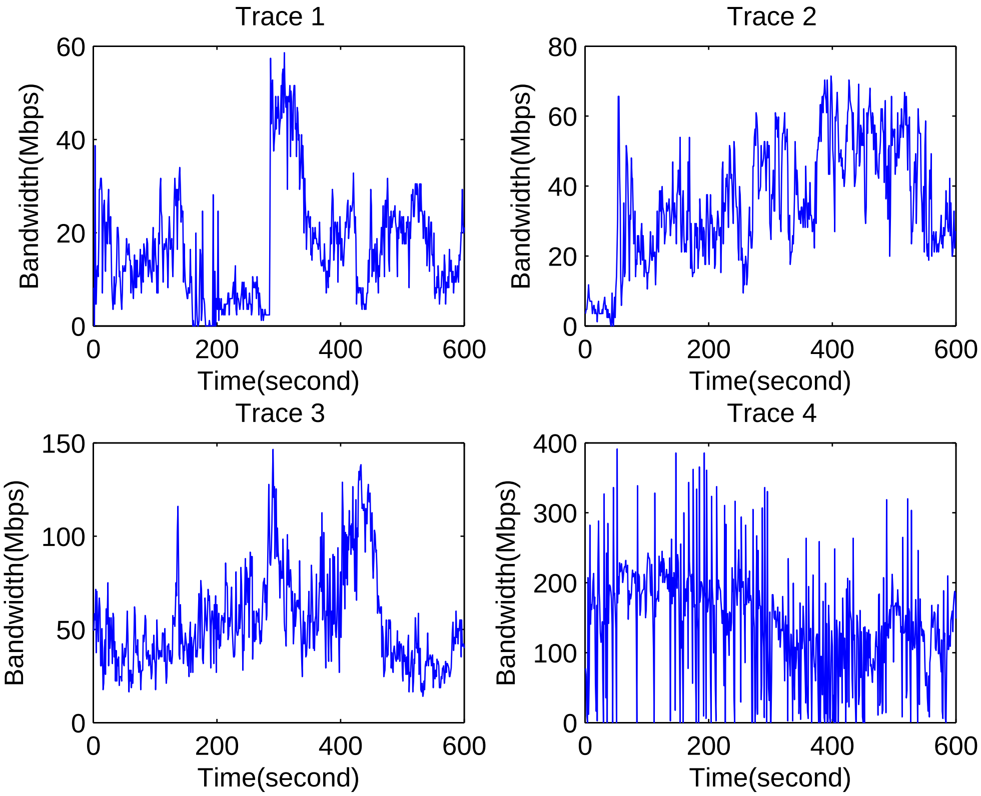


Figure 7.6: Sample Network Bandwidth Traces after Scaling.

7.5.2 Network Bandwidth Traces

To simulate dynamic networks with significant bandwidth variations, we use the traces collected over a 3.5G HSPA cellular network using the methodologies described in [30]. Sample traces are illustrated in Figure 7.6. These traces represent the most typical bandwidth variations in a cellular network. We further scale up the original bandwidth sample values to adapt to the 4K/30Hz 360 degree video bitrate range.

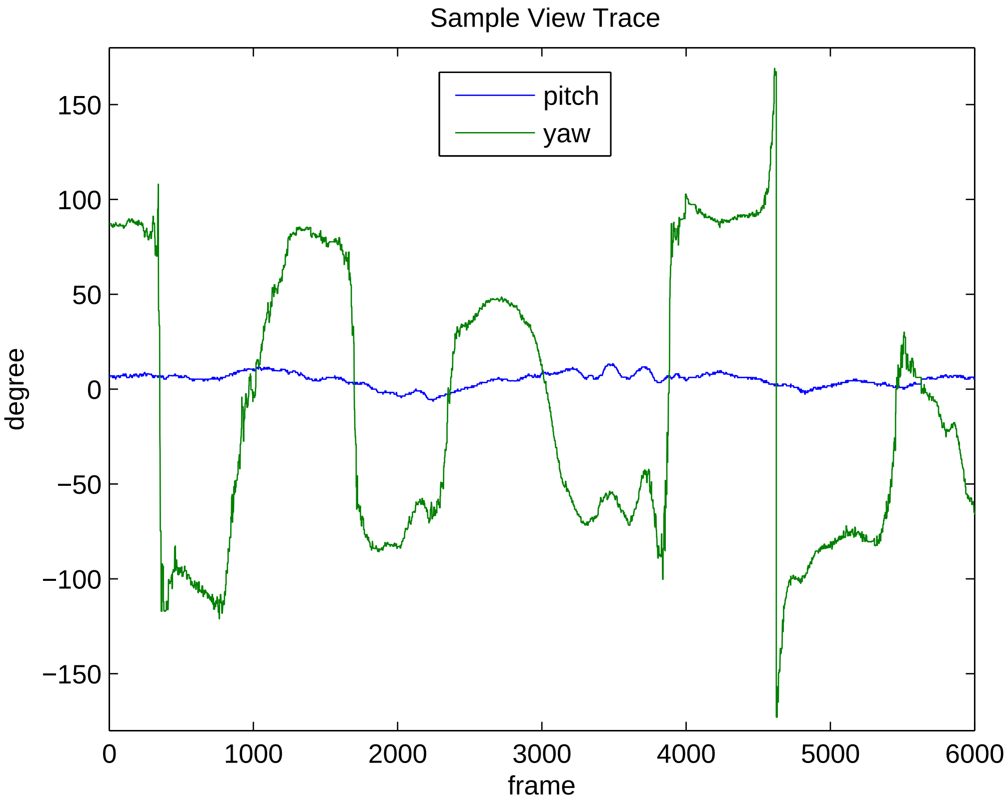


Figure 7.7: Sample view trace in yaw and pitch directions. Over yaw trace, the -180° coincides with +180°.

7.5.3 View Direction Traces

We collected the view direction traces from four users. Each user wears a Google Cardboard with a Motorola Nexus-6 smart-phone playing our test 360 videos. Simultaneously, a head tracker equipped on Cardboard dynamically transmits motion data (e.g., yaw, pitch, roll, etc.) to a nearby PC for data recording. Figure 7.7 illustrates a sample trace captured over the “RollerCoaster” test sequence.

7.5.4 Proposed Two-Tier Solution

In our proposed Two-Tier system (TTS), the BT stores the 360 degree view segments (each lasting 1 second) coded at a low bitrate R_0 =10 Mbps to provide basic quality. The ET stores all possible VCs ($120^\circ \times 90^\circ$). The VCs are coded at three different rates (R_1, R_2 or R_3). The initial buffer length for BT is set to 20 seconds. Upon complete reception of a chunk, the client estimates the available bandwidth for the next chunk to be equal to the average throughput for downloading the last chunk (from either BT or ET), i.e., $\hat{b}(k+1) = s(k)/T(k)$ where $s(k)$ is the size of the last received chunk k and $T(k)$ is the transmission time of chunk k. We select the target video rates based on the network bandwidth cumulative distribution function (CDF) in our traces. Specifically, R_0+R_1, R_0+R_2, R_0+R_3 are chosen at 40%, 60% and 80% percentile of the network bandwidth CDF, respectively. Furthermore, the client predicts the viewing direction for segment $n+1$ through linear regression based on the past 30 view samples and determine the VC to be fetched. When the BT queue length is less than q^b_{ref}, the client will always sequentially download the BT chunks. When the BT queue length is above q^b_{ref} (i.e., 10 second), the client will download ET chunks with target rate $\hat{R}\,(n+1)$ according to Eq. (7.2) and Eq. (7.3). Let $R_{n+1,L}$, $R_{n+1,M}$ and $R_{n+1,H}$ denote the actual rates of the low-quality, medium-quality and the high-quality versions of incoming chunk $n+1$ for the target VC, respectively. If $\hat{R}\,(n+1) \geq R_{n+1,H}$, the client will request the high-rate ET chunk. If $\hat{R}\,(n+1) < R_{n+1,H}$ and $\hat{R}\,(n+1) \geq R_{n+1,M}$, the client will request the medium-rate ET chunk. Otherwise, the client will request the low-rate ET chunk.

We use the received video rendering rate to quantify the system performance. Let $R^b(n)$ indicate the rate of the BT chunk for time segment n and $R^e(n)$ denote the rate of the ET chunk for a particular VC for time segment n. $R^e(n) = 0$ if the ET chunk is not available at the display time. We define w_f as the overlapping portion between the ground-truth FOV per frame (obtained from the ground-truth view trace) and the VC of the downloaded ET chunk, w_b the overlapping portion of the desired FOV and the 360 view decoded from the BT. Therefore, the VRR in Segment n is defined as $VRR(n) = w_b R^b(n) + w_f R^e(n)$. We assume that the rendered view covers $120° \times 90°$ so w_b=1/6. In the rare case when the BT buffer is empty at the display time, $VRR(n) = 0$.

7.5.5 Benchmark Solution 1 (BS1): Full-360 Streaming

BS1 simulates the typical $DASH$ streaming framework for 360 videos, in which the entire equirectangular videos are pre-encoded using multiple rates. For a fair comparison, we select the same rate setting as our proposed TTS (i.e., R_B=R_0=10 Mbps, R_L=R_0+R_1, R_M=R_0+R_2 and R_H=R_0+R_3). The initial buffer length is configured the same as TTS (i.e., 20 seconds) and the target buffer length is also 10 second. Similarly to TTS, the client estimates the sustainable transmission rate for the incoming segment $n + 1$ to be equal to the measured throughput for downloading the last received chunk n, and then accordingly chooses a rate to request over the next segment $n+1$ using PI-controller. The VRR for the desired view over each segment is therefore the rate of the downloaded chunk scaled by $w_b = 1/6$ and is 0 when the display buffer is empty.

7.5.6 Benchmark Solution 2 (BS2): VC-Streaming

In $BS2$, only VCs are pre-coded and stored on the server. Each VC covers $120° \times 90°$ view scope similarly as our TTS. Each ET chunk is encoded directly with four rates consistent with our proposed TTS (i.e., R_B=R_0=10 Mbps, R_L=R_0+R_1, R_M=R_0+R_2 and R_H=R_0+R_3). Similar to TTS, at time n, it predicts the view direction at $n+1$ using the linear regression based on the past 30 samples. If the requested segment does not arrive completely before its display deadline, VRR is set with 0 over that second. Otherwise, we use the portion of the downloaded VC that overlaps with the user desired FOV to calculate its VRR as $w_f \cdot R$, where $R \in \{R_B, R_L, R_M, R_H\}$. To be fair, we apply the same PI controller and the initial enhancement tier buffer length (i.e., 1 second) as in TTS configuration.

7.6 Experimental Results and Evaluation

Our proposed TTS is simulated and compared with the two benchmark solutions. We evaluate the performance directly using the delivered Video Rendering Rate (i.e, VRR) and Video Freeze Ratio (VFR). The delivered VRR is defined as the received bits per rendered area, averaged over all displayed frames. The VFR is the percentage of total time that video buffer underflows (i.e. no bits are available for the user FOV at the display time). Four different network traces (each of 600 seconds) and two view traces are used for simulation. For simplicity, the view traces are played in loops for a total duration of 600 seconds. After configuring the optimal q^e_{ref} as introduced in Section 7.3, The PI-controller parameters are chosen to maximize the ET chunk pass rate through an iterative search between K_P and K_I over the concatenated network trace, where K_P starts from 0.5 up to 1.0 with a stride of 0.1 and K_I starts from 0 up to 0.20 with a stride of 0.01.

Firstly, the VRR simulation data using different ET buffer length (i.e., q^e_{ref}) configurations are provided in Table 7.1. The result coincides with our conjecture in in Section 7.3 that the q^e_{ref} that maximizes the product of the view prediction accuracy (i.e., α) and the chunk pass rate (i.e., γ) will in turn maximize the received video rendering rate.

Table 7.1: Video Rendering Rate in Different ET Target Buffer Length

q^e_{ref}	1-second	2-second	3-second	4-second	5-second
VRR ($Mbps$)	49.80	49.15	47.10	46.24	45.13

The recipient video rendering rate (VRR) and chunk pass rate (CPR) using two benchmark solutions and our proposed solution are summarized in Table 7.2. Four network traces and two view traces are used for simulations, as visualized in Figure 7.6 and Figure 7.7, respectively.

Based on the simulation results, the following conclusions can be drawn:

In $BS1$, the long buffer setting effectively absorbs the network bandwidth variations and the available bandwidth is well-utilized. However, due to the ignorance of user FOV in streaming, the VRR is only 1/6 of the encoded rectangular video rate.

In $BS2$, when bandwidth is sufficient, the high-rate chunks can be successfully delivered. The delivered VRR is maximized when the view prediction is also accurate (particularly when the user viewing direction is relatively stable or changing smoothly). However, when the bandwidth suddenly decreases, the shallow enhancement-tier buffer may occasionally underflow, resulting in annoying frequent video freezes and severely degrade the user experience.

In our proposed TTS, the advantages of two benchmark solutions are integrated. On the one hand, the base-tier long buffer can effectively absorb the errors in both bandwidth estimation and view prediction, and therefore provides continuous playback with minimum freeze. On the other hand, the received enhancement-

Table 7.2: Performance Evaluations against Benchmark Solutions in Average Video Rendering Rate (Mbps) / Video Freeze Ratio (%)

Network Trace	Solution	RollerCoaster	Amsterdam
1	BS1	2.8/12%	2.8/12%
	BS2	10.8/27%	10.5/25%
	TTS	7.9/6%	7.7/6%
2	BS1	5.9/4%	5.9/4%
	BS2	27.0/10%	27.3/10%
	TTS	21.1/4%	22.3/3%
3	BS1	9.0/1%	9.0/1%
	BS2	40.2/2%	39.3/2%
	TTS	36.6/0%	36.1/0%
4	BS1	23.0/0%	23.0/0%
	BS2	93.1/8%	91.5/7%
	TTS	108.0/0%	106.3/0%

tier chunks boost the quality when extra bandwidth is available. Compared with $BS1$, a 3.7x gain in delivered VRR is achieved on average. The delivered VRR margin between proposed TTS and $BS2$ is primarily caused by the base-tier representation. Specifically, our BT is coded to cover the entire 360 video scope with fixed rate $R_0 = 10$ Mbps and only 1/6 of the total base-tier rate contributes to the delivered VRR, resulting in an initial loss of approximately 8 Mbps. However, when the network average throughput is large, this initial margin becomes negligible and our proposed TTS outperforms $BS2$, as shown from Trace 4 result in Table 7.2. Besides, with the prefetched base-tier, TTS is much more robust against sudden bandwidth decrease and view prediction error than $BS2$, and therefore has much lower video freeze ratio (VFR).

Chapter 8

Conclusions and Future Work

8.1 Summary

8.1.1 Fast Screen Content Compression

In this thesis, we propose three frameworks to accelerate screen content encoding and transcoding, as summarized below.

Firstly, a novel fast screen content encoding system is designed based on decision tree classifiers. By exploiting CU features, three classifiers are trained and incorporated into SC encoder for fast partition and mode decisions. Classifier 1 aims to categorize the incoming CU into either an "NIB" or an "SCB". "NIB" will be encoded using only Intra mode while "SCB" will be encoded using SCC modes at the current CU level. Classifier 2 makes fast partition decision and classifies "NIB"s into "Partitioned Block" (P-Block) and "Non-Partition Block" (NP-Block), where P-Block will bypass the current level Intra processing and NP-Block will terminate RDOs immediately after the current level Intra processing. Classifier 3 further classifies NP-Blocks into either "Directional Blocks" (D-Block) or “Non-Directional Blocks” (ND-Block) and only corresponding subset of Intra sub-modes are examined. The trade-off between the encoding efficiency and complexity can be flexibly tuned by adjusting the classifier confidence configurations and the rate thresholds. The proposed framework achieves a 40% average complexity reduction with only 1.46% BD-rate loss under “RD-Preserving” configuration and yields a 52% complexity reduction with 3.65% BD-rate loss under “Complexity Reduction Boosting” configuration over typical screen content videos under All-Intra configurations.

Secondly, a novel HEVC-SCC heterogeneous transcoding framework is proposed, based on screen content characteristics study and machine learning. In our transcoding system, a neural network based classifier is trained to classify the incoming CU into either an “NIB” or an “SCB”, based on the CU level features and the decoded residual sparsity. Over “NIB”, our transcoding system mimics the HEVC mode and partition behavior, while over “SCB”, the HEVC-partitioned blocks can directly bypass the current level Intra mode and only examine the SCC modes. Compared with the SCM-4.0 encoder, our proposed framework introduces a 48% average complexity reduction with only 2% BD-Rate increase. The proposed framework is the first work in HEVC-SCC transcoding and can significantly benefit SC video delivery for bandwidth-critical applications.

Finally, a novel SCC-HEVC heterogeneous transcoding framework is introduced based on mode mapping techniques to support backward compatibilities over legacy HEVC devices. Based on the statistical studies and the side information extracted from the decoded SCC bitstream, our proposed transcoding framework can efficiently and accurately determine the corresponding HEVC mode and partition. 51% and 81% average re-encoding complexity reductions are achieved under

All-Intra (AI) and Low-Delay (LD) configurations, respectively, compared with the trivial SCC-HEVC transcoding solution. We further integrate the proposed framework into the single-input-multiple-output (SIMO) paradigm to transcode one high-quality SCC bitstream into multiple reduced-quality HEVC bitstreams and achieves 49% and 79% complexity reductions under All-Intra (AI) and Low-Delay (LD) configurations, respectively. The proposed framework is the first work in SCC-HEVC transcoding and significantly reduces the system processing complexity and delay to facilitate adaptive SC streaming services over the edge clouds.

8.1.2 Two-Tier 360-degree Video Streaming

In this thesis, a two-tier 360-degree video streaming framework is proposed. The base-tier (BT) streams video chunks covering the entire 360 view span encoded with a basic quality and pre-fetched in a long video buffer to handle dynamics in both network bandwidth and user viewing directions. The enhancement-tier (ET) streams video chunks in higher quality covering the predicted user FoV to boost the delivered video quality when extra bandwidth is available. A prioritized buffer-control based scheduling algorithm is proposed to adaptively determine the requesting tier and rate. Our proposed framework fully utilizes the potential of network pre-fetching and improve the bandwidth utilization while simultaneously accommodating the users viewing direction changes. Through our simulations driven by real network and view direction traces, our proposed framework demonstrates superior delivered video rendering rate (i.e., 3.7x) and system robustness against the view and bandwidth dynamics, compared with the benchmark 360-degree video streaming solutions.

8.2 Future Work

8.2.1 Hardware Accelerated Screen Content Encoding

In this thesis, software-based encoder acceleration framework and algorithms are proposed based on machine learning techniques. The framework is designed without changing the low-level implementations of Intra, IBC and PLT modes. Therefore, the prior HEVC and SCC fast algorithms on each individual mode could be incorporated into our proposed framework for an additional encoder speed-up. The overall performance is still far from realtime SC applications and therefore requires hardware level parallel processing (for example, using DSP, FPGA or GPU for encoder acceleration), Processing Unit coordination (i.e., between control core and encoding cores for request scheduling and corresponding I/O optimization) and Data-level optimization (e.g., using SIMD implementation). Besides, the incoming chipset with specialized internal silicon designed for machine learning (ML) and artificial intelligence (AI) can further promote the machine learning based SC encoder optimization.

8.2.2 Adaptive Screen Content Distribution over Cloud

In this thesis, a Single-Input-Multiple-Output (SIMO) screen content transcoding system prototype is proposed, to support screen content distributions over the cloud. This framework efficiently compresses the SC video to reduce the backbone traffic throughput between the central content server and the local edge servers. Inside the edge clouds, multiple copies of SC videos with different quality levels are generated to support the adaptive SC video streaming, in which subscribers can flexibly request the most suitable video version given the network and de-

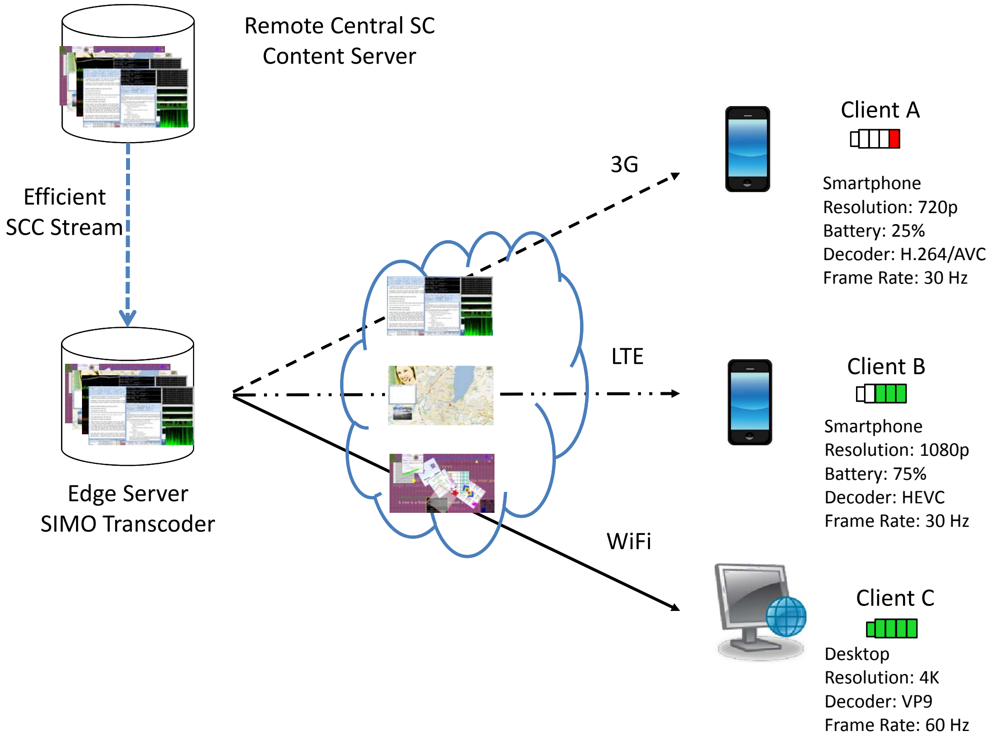


Figure 8.1: SIMO Screen Content Streaming over Edge Cloud

vice constraints, such as the available bandwidth, display resolution, battery life, computing capability, etc.

Within the scope of this thesis, for simplicity but without loss of generality, only the quality scaling is considered. In practice, temporal and spatial resolution scalings can be combined and optimized. On the other hand, the proposed system prototype at the edge clouds only tackles the bitstream conversions from SCC to HEVC. In practice, heterogeneous transcoding can be extended to support other coding standards, such as H.264/AVC, VP9, etc. The generalized paradigm is structured as illustrated in Figure 8.1.

8.2.3 Multi-tier Multi-path 360-degree Video Streaming

In this thesis, a general two-tier 360-degree video streaming framework is proposed for VR/AR content distribution. This general framework can be further extended (for example, based on motion-constrained tile set, as studied in [63]) and improved in many aspects, as briefly summarized into the followings.

View Prediction: In 360-degree video streaming, the view prediction methodology has a great impact on the system performance. The sample-based linear prediction method in this thesis can be definitely improved, for instance, using some advanced machine learning based regression solutions or even incorporated with content-based visual clues (e.g., via saliency analysis). Additionally, the viewing histories from other users watching the same or similar video can be mined as the "side" information to facilitate the view prediction of the target user. The spatial audio information and its semantic implication can also serve as additional side information to assist view prediction, as studied in [20]

Network Scheduling: Deep learning techniques have been widely used in networking applications in recent years and demonstrate superior performance beyond the traditional throughput-based or buffer-based approaches. For example, in a recent work [44], reinforcement learning is applied to derive target adaptive bitrate (ABR) during video streaming and achives significant performance improvement. A potential future direction is to utilize deep learning techniques to make scheduling decisions adaptively for 360 video or VR/AR to improve end user QoE.

Multi-tier Streaming: This thesis presents a general two-tier system prototype from a high-level and demonstrates the potential. In this framework, base-tier (BT) and enhancement-tier (ET) can be treated as video chunks with different view coverage, rate allocation, scheduling priority and prefetching configurations.

Therefore, an intuitive yet potential extension is to generalize the two-tier system to a multiple-tier system. The video segments in different tiers provide additional combination flexibility to increase bandwidth utilization and streaming robustness against the bandwidth and user viewport dynamics. For example, if video retransmission is allowed, we can design a third-tier of video segments (coded using tiles) to pack up the coverage gap between the predicted viewing direction and the user's ground-truth FoV as the "correction tier" (CT), assuming the small video tile requires less time to be delivered and the tiling does not introduce significant coding efficiency degradation.

Multi-path Streaming: This thesis only presents a general two-tier system prototype simulated over a 4G environment. In fact, the client may have multiple access links (e.g., WiFi, cellular, future 5G network) with different channel characteristics in terms of throughputs, latency, reliability and service cost, as illustrate in Figure 8.2. Therefore, in practice, multi-path 360-degree video streaming can jointly utilize the advantages from each path and potentially improve the delivered 360 video QoE. Some preliminary studies and results are summarized and provided in [62].

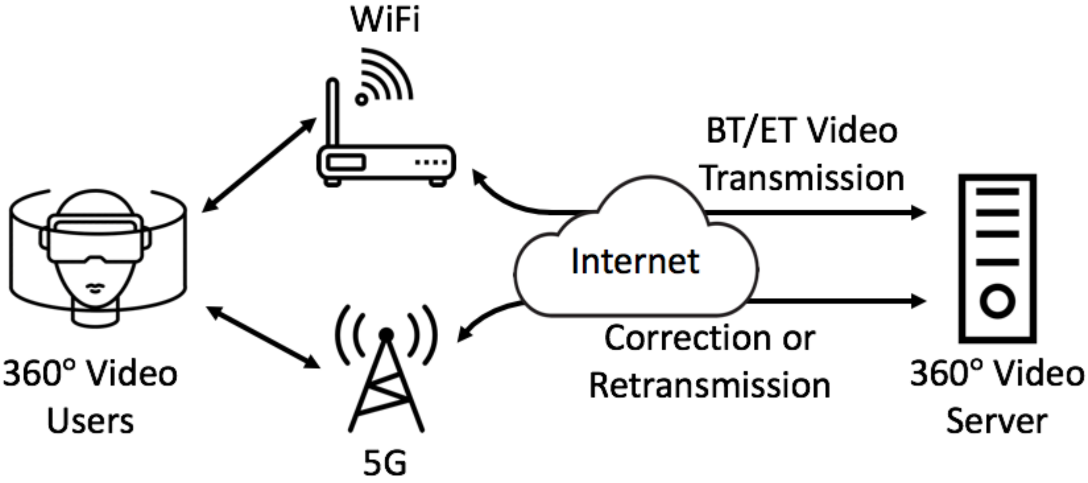


Figure 8.2: Multi-Path 360 Video Streaming Demonstration

References

[1] E. Alshina, J. Boyce, A. Abbas, and Y. Ye. *JVET common test conditions and evaluation procedures for 360° video, JCTVC Doc. G1030*, 2017.

[2] G. Bjontegaard. *Calculation of Average PSNR differences Between RD Curves, VCEG Doc. M33*, 2001.

[3] S. G. Blasi, E. Peixoto, B. Macchiavello, E. M. Hung, I. Zupancic, and E. Izquierdo. Context adaptive mode sorting for fast hevc mode decision. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1478–1482, Sept 2015.

[4] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson. 360 degrees video coding using region adaptive smoothing. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pages 750–754. IEEE, Sept. 2015.

[5] J. Chen, Y. Chen, T. Hsieh, R. Joshi, M. Karczewicz, W.-S. Kim, X. Li, C. Pang, W. Pu, K. Rapaka, J. Sole, L. Zhang, and F. Zou. *Description of screen content coding technology proposal by Qualcomm, JCTVC Doc. Q0031*, 2014.

[6] C.-H. Cheung and L.-M. Po. A novel cross-diamond search algorithm for fast

block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(12):1168–1177, Dec 2002.

[7] R. Cohen, F. Liu, C. S. Ping, N.-M. Cheung, Y. Chau, and S.-K. Yeung. *AHG8: 4:4:4 game content sequences for HEVC Range Extensions development, JCTVC Doc. N0294*, 2013.

[8] A. J. Daz-Honrubia, J. L. Martnez, J. M. Puerta, J. A. Gmez, J. D. Cock, and P. Cuenca. Fast quadtree level decision algorithm for h.264/hevc transcoder. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2497–2501, Oct 2014.

[9] G. V. der Auwera, H. M. Coban, and M. Karczewicz. *Truncated Square Pyramid Projection (TSP) For 360 Video, JVET Doc. D0071*, 2016.

[10] W. Ding, Y. Shi, and B. Yin. *YUV444 test sequences for screen content, JCTVC Doc. L0431*, 2013.

[11] F. Duanmu, Y. He, X. Xiu, P. Hanhart, Y. Ye, and Y. Wang. Hybrid cubemap projection format for 360-degree video coding. In *2018 Data Compression Conference*, pages 404–404, March 2018.

[12] F. Duanmu, E. Kurdoglu, S. A. Hosseini, Y. Liu, and Y. Wang. Prioritized buffer control in two-tier 360 video streaming. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, VR/AR Network '17, pages 13–18, New York, NY, USA, 2017. ACM.

[13] F. Duanmu, E. Kurdoglu, Y. Liu, and Y. Wang. View direction and bandwidth adaptive 360 degree video streaming using a two-tier system. In *2017 IEEE*

International Symposium on Circuits and Systems (ISCAS), pages 1–4, May 2017.

[14] F. Duanmu, E. Kurdoglu, Y. Liu, and Y. Wang. View direction and bandwidth adaptive 360 degree video streaming using a two-tier system. In *Proc. IEEE International Symposium on Circuits and Systems*, ISCAS '17, Baltimore, MD, USA, 2017. IEEE.

[15] F. Duanmu, Z. M, M. Xu, W. Wang, and H. Yu. *Non-SCCE1: Analysis of Full Frame IBC Block Vector Distribution, JCTVC Doc. R0269*, 2014.

[16] F. Duanmu, Z. Ma, W. Wang, M. Xu, and Y. Wang. A novel screen content fast transcoding framework based on statistical study and machine learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4205–4209, Sept 2016.

[17] F. Duanmu, Z. Ma, and Y. Wang. Fast cu partition decision using machine learning for screen content compression. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4972–4976, Sept 2015.

[18] F. Duanmu, Z. Ma, and Y. Wang. Fast mode and partition decision using machine learning for intra-frame coding in hevc screen content coding extension. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4):517–531, Dec 2016.

[19] F. Duanmu, Z. Ma, M. Xu, and Y. Wang. An hevc-compliant fast screen content transcoding framework based on mode mapping. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2018.

[20] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang. A subjective study of viewer navigation behaviors when watching 360-degree videos on computers. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2018.

[21] F. Duanmu, M. Xu, Y. Wang, and Z. Ma. Hevc-compliant screen content transcoding based on mode mapping and fast termination. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, Dec 2017.

[22] L. Guo, M. Karczewicz, and J. Sole. *RCE3: Results of Test 3.1 on Palette Mode for Screen Content Coding, JCTVC Doc. N0247*, 2013.

[23] P. Hanhart, X. Xiu, F. Duanmu, Y. He, and Y. Ye. *InterDigitals Response to the 360 Video Category in Joint Call for Evidence on Video Compression with Capability beyond HEVC, JVET Doc. G0024*, 2017.

[24] Y. He, X. Xiu, P. Hanhart, Y. Ye, F. Duanmu, and Y. Wang. Content-adaptive 360-degree video coding using hybrid cubemap projection. In *2018 Picture Coding Symposium (PCS)*, pages 313–317, June 2018.

[25] A. J. D. Honrubia, J. L. Martnez, P. Cuenca, J. A. Gmez, and J. M. Puerta. A data-driven probabilistic ctu splitting algorithm for fast h.264/hevc video transcoding. In *2015 Data Compression Conference*, pages 449–449, April 2015.

[26] J. Hou, D. Li, Z. Li, and X. Jiang. Fast cu size decision based on texture complexity for hevc intra coding. In *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, pages 1096–1099, Dec 2013.

[27] W. Jiang, H. Ma, and Y. Chen. Gradient based fast mode decision algorithm for intra prediction in hevc. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 1836–1840, April 2012.

[28] R. Joshi, J. Xu, R. Cohen, S. Liu, Z. Ma, and Y. Ye. *Screen Content Coding Test Model 4 Encoder Description (SCM 4), JCTVC Doc. T1014*, 2015.

[29] J. Jung, B. Bross, P. Chen, and W.-J. Han. *Description of Core Experiment 9 (CE9): MV Coding and Skip/Merge operations, JCTVC Doc. D609*, 2011.

[30] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, and J. Lyu. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, New York, NY, USA, 2016. ACM.

[31] E. Kuzyakov. Under the hood: Building 360 video, 2015.

[32] E. Kuzyakov. Next-generation video encoding techniques for 360 video and vr, 2016.

[33] D. K. Kwon and M. Budagavi. Fast intra block copy (intrabc) search for hevc screen content coding. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 9–12, June 2014.

[34] D. Lee, S. Yang, H. J. Shim, and B. Jeon. Fast transform skip mode decision for hevc screen content coding. In *2015 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–4, June 2015.

[35] B. Li and J. Xu. *Hash-based motion search, JCTVC Doc. Q0245*, 2014.

[36] B. Li and J. Xu. A fast algorithm for adaptive motion compensation precision in screen content coding. In *2015 Data Compression Conference*, pages 243–252, April 2015.

[37] B. Li, J. Xu, G. J. Sullivan, Y. Zhou, and B. Lin. *Adaptive motion vector resolution for screen content, JCTVC Doc. S0085*, 2014.

[38] B. Li, J. Xu, and F. Wu. A unified framework of hash-based matching for screen content coding. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 530–533, Dec 2014.

[39] R. Li, B. Zeng, and M. L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438–442, Aug 1994.

[40] X. Li, J. Sole, and M. Karczewicz. *Adaptive MV precision for Screen Content Coding, JCTVC Doc. P0283*, 2014.

[41] H.-C. Lin, C.-Y. Li, J.-L. Lin, S.-K. Chang, and C.-C. Ju. *An efficient compact layout for octahedron format, JVET Doc. D0142*, 2016.

[42] J.-L. Lin, Y.-P. Tsai, Y.-W. Huang, and S. Lei. *Improved Advanced Motion Vector Prediction, JCTVC Doc. D125*, 2011.

[43] Y. Liu, Z. Chen, J. Fang, and P. Chang. SVM based fast intra CU depth decision for hevc. In *2015 Data Compression Conference*, pages 458–458, April 2015.

[44] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest*

Group on Data Communication, SIGCOMM '17, pages 197–210, New York, NY, USA, 2017. ACM.

[45] MATLAB. *Neural Network Toolbox, Release 2013b, Natick, MA*, 2013.

[46] MATLAB. *Statistics Toolbox, Release 2013b, Natick, MA*, 2013.

[47] K. McCann, B. Bross, S. Sekiguchi, and W.-J. Han. *Encoder-Side Description of HEVC Test Model (HM), JCTVC Doc. C402*, 2010.

[48] A. Nagaraghatta, Y. Zhao, G. Maxwell, and S. Kannangara. Fast h.264/avc to hevc transcoding using mode merging and mode mapping. In *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pages 165–169, Sept 2015.

[49] C. Pang, J. Sole, L. Guo, M. Karczewicz, and R. Joshi. *Non-RCE3: Intra Motion Compensation with 2-D MVs, JCTVC Doc. N0256*, 2013.

[50] R. Pantos and W. May. HTTP live streaming. *RFC*, 8216:1–60, 2017.

[51] E. Peixoto, B. Macchiavello, R. L. de Queiroz, and E. M. Hung. Fast h.264/avc to hevc transcoding based on machine learning. In *2014 International Telecommunications Symposium (ITS)*, pages 1–4, Aug 2014.

[52] E. Peixoto, B. Macchiavello, E. M. Hung, and R. L. de Queiroz. A fast hevc transcoder based on content modeling and early termination. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2532–2536, Oct 2014.

[53] E. Peixoto, B. Macchiavello, E. M. Hung, A. Zaghetto, T. Shanableh, and E. Izquierdo. An h.264/avc to hevc video transcoder based on mode mapping.

In *2013 IEEE International Conference on Image Processing*, pages 1972–1976, Sept 2013.

[54] Y. Piao, J. Min, and J. Chen. *Encoder Improvement of Unified Intra Prediction, JCTVC Doc. C207*, 2010.

[55] L.-M. Po and W.-C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):313–317, Jun 1996.

[56] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 Video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ATC '16, pages 1–6, New York, NY, USA, 2016. ACM.

[57] K. Rapaka and J. Xu. *Software for SCM with hash based motion search, JCTVC Doc. Q0248*, 2014.

[58] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE Multimedia*, 18(4):62–67, 2011.

[59] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 2011.

[60] G. J. Sullivan, J. M. Boyce, Y. Chen, J. R. Ohm, C. A. Segall, and A. Vetro. Standardized extensions of high efficiency video coding (hevc). *IEEE Journal of Selected Topics in Signal Processing*, 7(6):1001–1016, Dec 2013.

[61] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high

efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[62] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. Multi-path multi-tier 360-degree video streaming in 5g networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 162–173, New York, NY, USA, 2018. ACM.

[63] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):43–57, March 2019.

[64] G. Tian and Y. Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 109–120. ACM, 2012.

[65] A. M. Tourapis. Enhanced predictive zonal search for single and multiple frame motion estimation. In *Visual Communications and Image Processing 2002, San Jose, CA, USA, January 19, 2002*, pages 1069–1079, 2002.

[66] S. H. Tsang, Y. L. Chan, and W. C. Siu. Fast and efficient intra coding techniques for smooth regions in screen content coding based on boundary prediction samples. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1409–1413, April 2015.

[67] A. Vetro, C. Christopoulos, and H. Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29, Mar 2003.

[68] C. Wang, B. Li, J. Wang, H. Zhang, H. Chen, Y. Xu, and Z. Ma. Single-input-multiple-ouput transcoding for video streaming. In *2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5, Sept 2016.

[69] S. Wang, M. Jiao, T. Lin, and K. Zhou. *AHG8: YUV444 and RGB screen content test sequences, JCTVC Doc. L0317*, 2013.

[70] Y.-K. Wang, Hendry, and M. Karczewicz. *Tile based VR video encoding and decoding schemes, JCTVC Doc. X0077*, 2016.

[71] M. Wien, V. Baroncini, J. Boyce, A. Segall, and T. Suzuki. *Joint Call for Evidence on Video Compression with Capability beyond HEVC, JCTVC Doc. F1002*, 2017.

[72] M. Xu, W. Wang, Z. Ma, and H. Yu. *AHG6: Information on the Usage of IBC, Palette, and Intra Prediction in SCC, JCTVC Doc. T0194*, 2015.

[73] Y. Xu, W. Huang, W. Wang, F. Duanmu, and Z. Ma. 2-d index map coding for hevc screen content compression. In *2015 Data Compression Conference*, pages 263–272, April 2015.

[74] H. Yu, R. Cohen, W. Gao, Y. Cao, J. Ye, X. Wang, A. Vetro, and H. Sun. *AHG8: New 4:4:4 screen-content sequences for HEVC extension development, JCTVC Doc. L0301*, 2013.

[75] H. Yu, R. Cohen, K. Rapaka, and J. Xu. *Common Test Conditions for Screen Content Coding, JCTVC Doc. T1015*, 2015.

[76] C. Zhang, Y. Lu, J. Li, and Z. Wen. *segmented sphere projection (SSP) for 360-degree video content, JVET Doc. D0030*, 2016.

[77] H. Zhang and Z. Ma. Early termination schemes for fast intra mode decision in high efficiency video coding. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 45–48, May 2013.

[78] H. Zhang and Z. Ma. Fast intra mode decision for high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 24(4):660–668, April 2014.

[79] H. Zhang, Q. Zhou, N. Shi, F. Yang, X. Feng, and Z. Ma. Fast intra mode decision and block matching for hevc screen content compression. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1377–1381, March 2016.

[80] L. Zhang, J. Chen, J. Sole, M. Karczewicz, X. Xiu, Y. He, and Y. Ye. *SCCE5 Test 3.2.1: In-loop Color-Space Transform, JCTVC Doc. R0147*, 2014.

[81] M. Zhang, Y. Guo, and H. Bai. Fast intra partition algorithm for hevc screen content coding. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 390–393, Dec 2014.

[82] M. Zhang, J. Qu, and H. Bai. Entropy-based fast largest coding unit partition algorithm in high-efficiency video coding. *Entropy*, 15(6):2277–2287, 2013.

[83] L. Zhao, L. Zhang, S. Ma, and D. Zhao. Fast mode decision algorithm for intra prediction in hevc. In *2011 Visual Communications and Image Processing (VCIP)*, pages 1–4, Nov 2011.

[84] F. Zheng, Z. Shi, X. Zhang, and Z. Gao. Effective h.264/avc to hevc transcoder based on prediction homogeneity. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 233–236, Dec 2014.

[85] F. Zheng, Z. Shi, X. Zhang, and Z. Gao. Fast h.264/avc to hevc transcoding based on residual homogeneity. In *2014 International Conference on Audio, Language and Image Processing*, pages 765–770, July 2014.

[86] M. Zhou. *A study on compression efficiency of icosahedral projection, JVET Doc. D0023*, 2016.

[87] C. Zhu, X. Lin, and L.-P. Chau. Hexagon-based search pattern for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(5):349–355, May 2002.

[88] S. Zhu and K.-K. Ma. A new diamond search algorithm for fast block matching motion estimation. In *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications*, volume 1, pages 292–296 vol.1, Sep 1997.