# การเข้ารหัสด้วยขั้นตอนวิธีทางพันธุกรรมสำหรับการเข้ารหัสวีดิทัศน์ประสิทธิภาพสูง

นางสาว ไอไอ ทุน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

# GENETIC ALGORITHM BASED CODING FOR HIGH EFFICIENCY VIDEO CODING

Miss Ei Ei Tun

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2018

| | |
|---|---|
| Thesis Title | GENETIC ALGORITHM BASED CODING FOR HIGH EFFICIENCY VIDEO CODING |
| By | Miss Ei Ei Tun |
| Field of Study | Electrical Engineering |
| Thesis Advisor | Associate Professor Supavadee Aramvith, Ph.D |
| Thesis Co-Advisor | Professor Yoshikazu Miyanaga, Ph.D |

---

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

..... Dean of the Faculty of Engineering
(Professor Supot Teachavorasinkun, D.Eng.)

THESIS COMMITTEE

..... Chairman
(Professor Prasit Prapingmongkolkarn, Ph.D)

..... Thesis Advisor
(Associate Professor Supavadee Aramvith, Ph.D)

..... Thesis Co-Advisor
(Professor Yoshikazu Miyanaga, Ph.D)

..... Examiner
(Assistant Professor Suree Pumrin, Ph.D)

..... Examiner
(Assistant Professor Charnchai Pluempitiwiriyawej, Ph.D)

..... External Examiner
(Professor Kosin Chamnongthai, Ph.D)

ไอไอ ทุน : การเข้ารหัสด้วยขั้นตอนวิธีทางพันธุกรรมสำหรับการเข้ารหัสวีดิทัศน์ประสิทธิภาพสูง (Genetic Algorithm based Coding for High Efficiency Video Coding) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร. สุภาวดี อร่ามวิทย์, 65 หน้า.

การเข้ารหัสวิดิทัศน์ประสิทธิภาพสูง (High efficiency video coding, HEVC) เป็นมาตรฐานการเข้ารหัสรูปแบบใหม่ที่สามารถเพิ่มประสิทธิภาพให้กับการเข้ารหัสวิดิทัศน์ H.264/AVC แบบดั้งเดิมได้อย่างมาก เทคนิคดังกล่าวอาศัยคุณลักษณะแบบพิเศษของหน่วยการเข้ารหัส (Coding Unit, CU) ประกอบด้วยการแยกส่วนหน่วยการเข้ารหัสที่อยู่บนพื้นฐานของ quadtree (Quadtree-based CU partitioning) ตัวกรองวิดิทัศน์แบบ deblocking (deblocking filter) ซึ่งจะช่วยให้ภาพไม่แตกเป็นบล็อก และเทคนิคการเข้ารหัสขั้นสูงอื่นๆ อย่างไรก็ดี HEVC มีความซับซ้อนในการคำนวณที่สูงมาก ซึ่งทำให้การค้นหาโดยวิธีทาง optimization (optimization search) ใน Quadtree-based CU partitioning มีความผิดเพี้ยนอย่างมาก

ในส่วนแรก งานวิจัยนี้นำเสนอวิธีการลดคุณลักษณะ (feature reduction) ด้วยการใช้วิธีการตัดสินใจ (decision method) เชิงขนาดของ CU ซึ่งอยู่บนพื้นฐานของฟัซซี่ support vector machine (SVM) ร่วมกับตัวควบคุมอัตรา (rate control, RC) เพื่อลดความซับซ้อนในการคำนวณ (computational complexity) โดยวิธีที่นำเสนออาศัยการกำจัดคุณลักษณะที่สัมพันธ์กัน (correlated) ของฟัซซี่ SVM บางตัวออกไปโดยที่ประสิทธิภาพการเข้ารหัสคงเดิมจากผลการทดลอง พบว่าวิธีที่นำเสนอสามารถลดความซับซ้อนในการคำนวณด้วยวิธีฟัซซี่ SVM ได้ถึง 3% ภายใต้สมรรถนะของอัตราความผิดเพี้ยนเดียวกัน

นอกจากนี้ งานวิจัยนี้ยังนำเสนอวิธีการทาง optimization เพื่อหารูปแบบการแยกส่วนแบบเหมาะที่สุดสำหรับ CU แทนการใช้ขั้นตอนวิธีอย่างเร็ว (fast algorithm) โดยวิธีที่นำเสนออยู่บนพื้นฐานของขั้นตอนวิธีเชิงพันธุกรรม (Genetic algorithm, GA) เพื่อลดทรัพยากรในการคำนวณของ Quadtree-based CU partitioning เชิงลำดับชั้น โดยที่รูปแบบ (pattern) ของการแยกส่วน CU อย่างละเอียด (Exhaustive partitioning) และอัตราความผิดเพี้ยน (distortion rate) ถูกพิจารณาเสมือนว่าเป็นโครโมโซม และการทำงานที่เหมาะสมที่สุด (fitness function) ของขั้นตอนวิธีเชิงพันธุกรรม ตามลำดับ รูปแบบการแยกส่วน CU ของเฟรมหลัก (key frame) จะถูกค้นหา และส่งต่อไปยังเฟรมถัดไปที่ติดกันภายใต้ความสัมพันธ์เชิงเวลา (temporal correlation) ที่สูง เพื่อลดเวลาในการคำนวนลง จากการทดสอบ ขั้นตอนวิธีที่นำเสนอสามารถลดทรัพยากรในการคำนวณภายใต้การกำหนด delay P ที่ต่ำด้วย rate control ได้ถึง 62.5% และ 16.7% ที่ 8 Mbps ด้วยการลดคุณภาพในส่วนที่ละเลยได้ และสามารถลดทรัพยาการในการคำนวณได้ถึง 64.1% และ 15.1% ภายใต้การกำหนด delay ต่ำๆ เมื่อเทียบกับ HM16.5 และขั้นตอนวิธีอย่างเร็วซึ่งอยู่บนพื้นฐานของ SVM ตามลำดับ

**ภาควิชา** วิศวกรรมไฟฟ้า **ลายมือชื่อนิสิต** .....
**สาขาวิชา** วิศวกรรมไฟฟ้า **ลายมือชื่อ อ.ที่ปรึกษาหลัก** .....
**ปีการศึกษา** 2561

# # 5971459321 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: FAST ENCODING/ GENETIC ALGORITHM / HIGH EFFICIENCY VIDEO CODING/ QUADTREE-BASED CODING UNIT PARTITIONING.

EI EI TUN : GENETIC ALGORITHM BASED CODING FOR HIGH EFFICIENCY VIDEO CODING.
ADVISOR: ASSOC. PROF. SUPAVADEE ARAMVITH, Ph.D,
CO-ADVISOR: PROF. YOSHIKAZU MIYANAGA, Ph.D, 65 pp.

High efficiency video coding (HEVC) is the newest video coding standard to greatly increase the coding efficiency of its ancestor H.264/AVC with the aids of its new features such as the quadtree-based coding unit (CU) partitioning, a simple deblocking filter, and other advanced coding techniques. However, HEVC delivers a highly increased computation complexity, which is mainly due to the exhaustive rate distortion optimization search of quadtree-based CU partitioning.

Firstly, a feature reduction approach is proposed on a fuzzy support vector machine (SVM) based CU size decision method. The proposed feature reduction approach with rate control (RC) can reduce computational complexity by eliminating some correlated features of a fuzzy SVM-based CU size decision method under a similar coding efficiency. According to the empirical results, our approach can achieve up to 3% of complexity reduction under the same rate distortion (RD) performance over a fuzzy SVM-based approach.

Secondly, instead of machine learning (ML) based fast algorithm approach, a CU partitioning pattern optimization method based on genetic algorithm (GA) is proposed to save the computational complexity of a hierarchical quadtree-based CU partitioning. The required coding unit partitioning pattern for an exhaustive partitioning and the rate distortion cost are efficiently considered as the chromosome and the fitness function of the genetic algorithm, respectively. To reduce the computational time, CU partitioning patterns of the key frame is searched and shared to other consecutive frames by taking into account the highly temporal correlation. Our evaluation results show that the proposed method can achieve 62.5% and 16.7% computational complexity reduction on average at 8 Mbps with a negligible average quality degradation compared with HM16.5 and state-of-the-art support vector machine-based fast algorithm, respectively, under low delay P configuration with rate control while 64.1% and 15.1% under low delay configuration with rate control.

**Department :** Electrical Engineering **Student's Signature** .....
**Field of Study :** Electrical Engineering **Advisor's Signature** .....
**Academic Year :** 2018

# Acknowledgements

Firstly, I wholeheartedly thank my advisor Associate Professor Dr. Supavadee Aramvith and my co-advisor Professor Dr. Yoshikazu Miyanaga, for supporting me to face and overcome the most challenging part of study life as their advisee and for guiding my study at Chulalongkorn University and Hokkaido University.

I also would like to express my heartfelt thanks to the members of the committee, Assistant Professor Dr. Suree Pumrin, Assistant Professor Dr. Charnchai Pluempitiwiriyawej, Professor Dr. Kosin Chamnongthai, and the Chairperson Professor Dr. Prasit Prapingmongkolkarn, for their valuable time, important review and advice for this research.

Thanks to all group members of the Video Processing Research Group (VTRG) for their suggestions, comments, and guidance. I believe that I have improved the presentation skill under the suggestions from my advisor and the group members from VTRG. Those suggestions are really valuable not only for this research work but also for my study life. I will always remember every single word from them in my mind.

I would like to thank also my parents, all of my friends for their help, their understanding and kind encouragements through my journey.

I would like to express my gratitude to AUN/SEED-Net Scholarship Program which provides the financial support for studying Doctoral Engineering Degree in the Chulalongkorn University, and I would like to thank the people at ISE and AUN/SEED-Net for their support and advice.

This work was supported in part by the Collaborative Research Project entitled Video Processing and Transmission, in part by the JICA Project for AUN/SEED-Net, This research has been supported in parts by the Collaborative Research Project entitled Video Processing and Transmission, JICA Project for AUN/SEEDNet, Japan, and the Ministry of Internal Affairs and Communications for SCOPE Program (185001003).

# Contents


Page

Thai Abstract . . . iv
English Abstract . . . v
Acknowledgements . . . vi
Contents . . . vii
List of Tables . . . vii
List of Figures . . . viii
1 Introduction . . . 1
1.1 Research Motivation and Problem Statement . . . 1
1.2 Contribution . . . 5
2 Background and Literature Review . . . 7
2.1 Background . . . 7
2.1.1 Overview of HEVC . . . 7
2.1.2 The Hierarchical Quadtree-based CU Partitioning . . . 11
2.1.3 Genetic Algorithm . . . 15
2.2 Literature Review . . . 16
2.2.1 Fast Algorithms for Intra-coding of HEVC . . . 17
2.2.2 Fast Algorithms for Inter-coding of HEVC . . . 17
3 Fast Coding Algorithm for High Efficiency Video Coding . . . 20
3.1 Reducing Redundant Feature from Fuzzy SVM-Based Coding Unit Decision in HEVC . . . 20
3.2 GA-Based Fast CU Partitioning . . . 23
3.2.1 Proposed Data Structure . . . 23
3.2.2 Proposed Fitness Function . . . 26
3.2.3 Selection, Crossover and Mutation . . . 28
3.2.4 Optimization Criteria . . . 29
3.2.5 Genetic Algorithm with the Proposed Chromosome and Fitness Function 29
3.2.6 Overall Algorithm . . . 30
4 Evaluation . . . 35
4.1 Test Video Sequences . . . 35
4.2 Experimental Setup . . . 40
4.3 Performance Metric . . . 40
4.4 Feature Reduction on Conventional Fuzzy SVM-based Approach . . . 41
4.5 GA-Based Fast CU Partitioning . . . 43
4.5.1 Performance Comparison with Original HM and State-of-the-art Approach 43
4.5.2 Stability of the Proposed Method . . . 48
4.6 Discussion . . . 52
4.6.1 Performance Comparison with CTU Level Sharing . . . 52

4.6.2 RD Cost Calculation with Two Most Common Modes and CU Prediction without Two Most common Modes . . . . . 56
4.6.3 GA-Based Fast CU Partitioning without Utilizing Temporal Correlation and with QP . . . . . 59
5 Conclusion and Future Works . . . . . 60
References . . . . . 62
Biography . . . . . 65

# List of Tables

Page


1.1 **Three categories for thirteen features.** . . . . . 3
1.2 **Selected Features of different classifiers.** . . . . . 4
3.1 **Example Feature Combinations of FuzzySVM [17].** . . . . . 21
3.2 **Selected Features for different classifiers [25] after eliminating redundant features of [17].** . . . . . 23
4.1 **Testing sequences of the JCT-VC data set.** . . . . . 40
4.2 **Performance comparison with HM16.5 and start-of-the-art fast algorithm, FuzzySVM[17] (LDP).** . . . . . 45
4.3 **Performance comparison with HM16.5 and start-of-the-art fast algorithm, FuzzySVM[17] (LD).** . . . . . 46
4.4 **Performance comparison of HM16.5, start-of-the-art fast algorithm [17], previous and proposed method [26]** . . . . . 58
4.5 **Performance Analysis of the proposed method [27] with HM16.5 and start-of-the-art fast algorithm, FuzzySVM [17].** . . . . . 59

# List of figures

Page


1.1 **Four consecutive frames of sequence "PartyScene" ($N = 4$).** . . . . . 6
2.1 **Block diagram of an H.264 encoder.** . . . . . 8
2.2 **Block diagram of an HEVC encoder with built-in decoder (gray shaded).** 9
2.3 **Demonstration of the partitioning of a frame with 1280 × 720 luma samples into (a) macroblocks. (b) CTU.** . . . . . 10
2.4 **Available partitioning modes for partitioning a CU into 1, 2, or 4 PUs.** 11
2.5 **Eleven PU partition modes.** . . . . . 12
2.6 **Illustration of RD cost calculating and comparing between a parent/current CU and its children/sub-CUs.** . . . . . 12
2.7 **The order of RD cost calculation for 85 CUs of a 64 x 64 CTU.** . . . 13
2.8 **Quadtree partitioning from CTU to CU.** . . . . . 14
2.9 **Quadtree partitioning of a CTU into CU based on recursive RD cost comparsion. (a) Subdivision into 64×64 CTUs. (b) Coding quatdree with CUs.** . . . . . 14
2.10 **Flowchart of GA.** . . . . . 16
3.1 **Finding optimal three feature sets based on misclassification. (a) C0. (b) C1. (c) C2.** . . . . . 22
3.2 **Hierarchical chromosome of a 64×64 CTU.** . . . . . 24
3.3 **Possible CU partition patterns of a 64×64 CU (Maximum CU Depth = 2).** . . . . . 25
3.4 **Usage of MERGE/SKIP mode (in green color) for PUs with the lowest RD cost in HEVC.(a) Traffic (Class A). (b) Kimono1 (Class B). (c) BQMall (Class C). (d) BlowingBubbles (Class D). (e) Johnny (Class E). (f) ChinaSpeed (Class F).** . . . . . 27
3.5 **Required 85 RD costs of a 64 x 64 CTU.** . . . . . 28
3.6 **Example splitting pattern of a CTU.** . . . . . 28
3.7 **Illustration for sharing partitioning pattern.** . . . . . 31
3.8 **Flowchart for quadtree-based CU partitioning of HM16.5.** . . . . . 32
3.9 **Flowchart for quadtree-based CU partitioning of FuzzySVM[17].** . . . 33
3.10 **Flowchart for quadtree-based CU partitioning of the proposed algorithm based on GA [26].** . . . . . 34
4.1 **Traffic test sequence of Class A.** . . . . . 35
4.2 **Class B. (a) Kimono1. (b) ParkScene. (c) Cactus. (d) BasketballDrive.** 36
4.3 **Class C. (a) BQMall. (b) PartyScene. (c) BasketballDrillText.** . . . . 37
4.4 **Class D. (a) BQSquare. (b) BlowingBubbles.** . . . . . 37
4.5 **Class E. (a) BQMall. (b) PartyScene. (c) BasketballDrillText.** . . . . 38

4.6 **ChinaSpeed test sequence of Class F.** . . . . . 38
4.7 **Example test sequences for each test sequence class printed in with appropriate relative size.** . . . . . 39
4.8 **Performance comparison with FuzzySVM [17] under LDP Configuration without RC and 4 QPs. (a) Computational time saving (%). (b) Video quality degradation BDPSNR (dB). (c) Overhead bitrate BDBR (%).** . . . . . 42
4.9 **Performance comparison with FuzzySVM [17] under LDP Configuration with RC. (a) Bitrate = 256 kbps.(b) Bitrate = 512 kbps.** . . . . . 43
4.10 **Average performance comparison with FuzzySVM [17] under LDP configuration. (a) Video quality (dB). (b) Computational time saving (%).** . . . . . 47
4.11 **Average performance comparison with FuzzySVM [17] under LD configuration. (a) Video quality (dB). (b) Computational time saving (%).** . . . . . 47
4.12 **Subjective video quality for POC 242 of BasketballDrillText video sequence.(a) Encoded frame of HM16.5. (b) Encoded frame of FuzzySVM. (c) Encoded frame of the proposed method [26].** . . . . 48
4.13 **Stabilization of proposed method[26] over FuzzySVM[17]. (a) Time saving of BQMall. (b) Time saving of Johnny.** . . . . . 49
4.14 **Time utilization percentage of the proposed method [26].** . . . . . 50
4.15 **SVM decision percentage of FuzzySVM[17] for BQMall (a) Depth 0 (b) Depth 1 (c) Depth 2.** . . . . . 50
4.16 **SVM decision percentage of FuzzySVM[17] for Johnny (a) Depth 0 (b) Depth 1 (c) Depth 2.** . . . . . 51
4.17 **Total SVM decision percentage of FuzzySVM[17] for depth 0, 1 and 2 (a) BQMall (b) Johnny.** . . . . . 51
4.18 **CU Partition of a CTU encoded by FuzzySVM[17] at 1 Mbps, 4 Mbps, and 8 Mbps.** . . . . . 52
4.19 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Traffic (Class A). (b) Time saving of Traffic (Class A).** . . . . . 53
4.20 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Kimono1 (Class B). (b) Time saving of Kimono1 (Class B).** . . . . . 53
4.21 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of PartyScene (Class C). (b) Time saving of PartyScene (Class C).** . . . . . 54

4.22 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of BlowingBubbles (Class D). (b) Time saving of BlowingBubbles (Class D).** . . . . . 55
4.23 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Johnny (Class E). (b) Time saving of Johnny (Class E).** . . . . . 55
4.24 **Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of ChinaSpeed (Class F). (b) Time saving of ChinaSpeed (Class F).** . . . . . 56
4.25 **Flowcharts of previous and proposed GA-based fast CU partitioning.** 57

# Chapter 1

# Introduction

## 1.1 Research Motivation and Problem Statement

Nowadays, video distribution for various purposes is proliferating over the Internet with the aids of handy communication networks and smart mobile devices. Besides, video consumers increasingly demand high definition (HD) and ultra-high definition (UHD) videos to experience better visual quality. As a result, the delivery of HD/UHD videos to the mobile devices' users over the Internet is becoming a popular trend. However, the data quantity for HD/UHD videos is huge due to the higher video resolution and frame rate. The data size of a 10-second video with $3840 \times 2160$ resolution at a frame rate of 60 frames per second reaches nearly 15 GB. Due to this, the delivery of HD/UHD videos demands a more substantial amount of network bandwidth and data storage compared to the lower resolution standard definition (SD) videos. To achieve the saving on network resources and storage requirement, an efficient compression technique is crucially important.

Joint Collaborative Team on video coding (JCT-VC), a collaborative project group of ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Expert Group (MPEG), has implemented a highly efficient video coding standard called High Efficiency Video Coding (HEVC)/H.265[1] as a solution to the issue of increased video resolution. HEVC delivers a twofold coding efficiency as compared to its preceding coding standard, namely, H.264/Advance Video Coding (AVC)[2]. Accurately, HEVC achieves the bit-rate saving of nearly 50% under the same visual quality compared to the H.264/AVC. As a result, HEVC becomes a popular video codec. HEVC brings such a significant coding efficiency thanks to its innovation coding features such as hierarchical quadtree-based partitioning, 33 directional modes for intra-picture prediction and simplified in-loop deblocking filtering, etc.

On the other hand, these coding features are highly expensive in terms of computational complexity and hardware requirements. Due to this, HEVC struggles to realize HD/UHD video delivery in real-time applications, for example, real-time video chat and remote surveillance, etc. Many solutions on mitigating computational complexity of HEVC

have been proposed in the literature since the last few years. Most of these solutions focus on a hierarchical quadtree-based coding unit (CU) partitioning because of the heaviest load of recursive CU partitioning in the HEVC encoder. In a hierarchically exhaustive CU partitioning of HEVC, the encoder firstly starts trial encoding which includes two processes: top-down rate distortion (RD) cost calculation and bottom-up RD cost comparison to find an optimal quadtree splitting pattern based on RD performance gain for each coding tree unit (CTU). To get an optimal CU partitioning pattern, every possible combination of CU size, prediction unit (PU) modes and transform unit (TU) sizes are exhaustively examined. Due to this, trial encoding especially the RD cost calculation is the most time-consuming module of HEVC, over 80% in the HEVC test model (HM) [3]. To tackle this problem, most of the studies aim to achieve the fast algorithm by replacing an exhaustive CU partitioning with a simple operation under the negligible RD loss.

In order to efficiently save the computational time, two main researches are focusing on complexity reduction in HEVC: intra-coding and inter-coding. Based on the new features of HEVC intra-coding such as two partitioning modes in quadtree-based partitioning and 35 intra prediction modes, the previous solutions for intra-coding [4]–[7] focus on intra-mode decision and CTU size prediction. For inter-coding, several time-consuming modules such as quadtree-based CU partitioning[8]–[20], CU mode estimation[21], motion estimation (ME)[22] have been improved to achieve a low complexity encoder. In specific, there are two groups on the fast inter-coding algorithms for quadtree-based CU partitioning. One is a statistical-based fast algorithm, and the other is a learning-based fast algorithm. The early statistical-based approaches have statistically decided CU size by observing the nature of original block partitioning in HEVC such as in [8]–[12]. These approaches determine some CU-related features and spatiotemporal-based hard threshold to decide CU size without looping RD optimization (RDO) process exhaustively. Starting a few years ago, most researches reported in[13]–[20] have focused on learning approach for fast algorithm due to the learning property from the largely complicated amount of data to the best decision. Among these learning-based fast inter-coding algorithms, CU size decision based on online SVM training (denoted by FuzzySVM)[17] is one of the best approaches in which the first group of picture (GOP) encoded with original HM was using as the training data for three

SVM classifiers. For feature selection, misclassification-based feature selection approach was utilized to get three different feature sets for depth 0, 1, and 2. The misclassification costs of all feature combinations were energetically calculated and the minimum one for each depth level (Depth 0 to Depth 2) was selected. The best feature set can make to get a better RD performance because feature selection is one of the most important part of a classifier and there is a relationship between features and estimation accuracy. As shown in Table 1.1, there are three categories for thirteen features: the temporal domain (Index 1), by-product feature of the current CU (Index 2 - 8) and the spatial domain (Index 9 - 13). The misclassfication-based optimal feature sets for three level classifiers of [17] are shown in Table 1.2. But there may have some redundant features according to their relationships. For example, distortion, bits, and RD cost are the RD performance related features according to the Eq. (1.1).

$$J_{RDO} = D + \lambda \times R \tag{1.1}$$

where, $J_{RDO}$, $D$ , $R$ and $\lambda$ are the RDO cost, distortion, bits and Lagrangian multiplier, respectively. Because of some correlations between three of them, some redundant features can be eliminated from the feature set to save the time consumed by that features with a negligible quality degradation. Therefore, for the first part of this thesis, we propose a redundant feature reduction for FuzzySVM [17] to reduce the amount of time for wasting some redundant features.

**Table 1.1: Three categories for thirteen features.**

| Category | Index | Candidates | Description |
|---|---|---|---|
| Temporal Domain | 1 | $x_{SAD}$ | Sum Absolute Difference of the current CU and co-located CU |
| By-product information | 2 | $x_{RDCost}$ | RD performance |
| | 3 | $x_{SkipFlag}$ | The skip flag PU level |
| | 4 | $x_{Distortion}$ | RD performance |
| | 5 | $x_{Bits}$ | RD performance |
| | 6 | $x_{CtxSkipFlag}$ | The flag of skipping in neighboring blocks |
| | 7 | $x_{QP}$ | The regulating element between distortion and bits |
| | 8 | $x_{CBF\_SKIP}$ | The important flag representing coded block |
| Spatial domain | 9 | $x_{MergeFlag}$ | The flag of merge mode |
| | 10 | $x_{MV}$ | The moving information |
| | 11 | $x_{Partition}$ | Selected PU size from possible modes |
| | 12 | $x_{Depth}$ | The block size |
| | 13 | $x_{CBF\_NB}$ | The coded block flags from neighboring block |

**Table 1.2: Selected Features of different classifiers.**

| Index | Candidates | Classifier C0 | Classifier C1 | Classifier C2 |
|---|---|---|---|---|
| 1 | $x_{SAD}$ | √ | | √ |
| 2 | $x_{RDCost}$ | √ | √ | √ |
| 3 | $x_{SkipFlag}$ | | √ | √ |
| 4 | $x_{Distortion}$ | √ | √ | |
| 5 | $x_{Bits}$ | | √ | √ |
| 6 | $x_{CtxSkipFlag}$ | | | √ |
| 7 | $x_{QP}$ | √ | | √ |
| 8 | $x_{CBF\_SKIP}$ | √ | | √ |
| 9 | $x_{MergeFlag}$ | | | √ |
| 10 | $x_{MV}$ | | | √ |
| 11 | $x_{Partition}$ | | | √ |
| 12 | $x_{Depth}$ | √ | | √ |
| 13 | $x_{CBF\_NB}$ | √ | | √ |

However, the training data of SVM classifiers of FuzzySVM is only from the first GOP, causing the classifier to confuse on the depth decision and make a misclassification. False positive (FP) is the misclassification when SVM classifier incorrectly decides the splitting (+) decision for the current CU instead of the non-splitting (–) decision. Due to this, the RD costs for unnecessary CUs are calculated and FuzzySVM may take a certain amount of time for unnecessary depth levels and may not effectively save the computational burden of the quadtree-based CU partitioning. Another critical situation is the risk area. If a sample is located in the risk area, the original HM is triggered that can consume the computational complexity of HEVC and may not significantly reduce the computational burden for risk area case. Another main factor is the target bit rate of rate control (RC) to assign the required bits for input video sequences. If the target bit rate is higher, the chance of the splitting decision can be higher. As a result, FuzzySVM may need to go to the high depth level and the number of CUs which need to be calculated the RD costs may be higher. Therefore, FuzzySVM may take a big computational time for calculating the RD costs of CUs at the high target bit rate. Additionally, all fast algorithms mentioned above do not consider finding a partitioning pattern of a CTU as an optimization problem. Finding an optimal CU partition pattern from all possible outcomes can be modeled as an optimization problem and can be solved by a simply useful optimizer instead of machine learning based approach.

Therefore, the purpose of the second part of this thesis is to save the computational

burden for quadtree-based CU partitioning by utilizing a genetic algorithm (GA). GA is a metaheuristic based on mechanisms of natural systems such as natural genetics and selection [23]. GA is a member of evolutionary algorithm (EA) and has been started by John Holland at the University of Michigan in the 1960s based on the biologically evolutionary theory called Darwinism. The aim of GA is for solving complex problems such as large-scale combinatorial and highly constrained optimization problems. Therefore, a simple fast CU encoding based on GA should be proposed and implemented for saving the computational time of quadtree-based CU partitioning of HEVC.

## 1.2 Contribution

There are two parts in our research work: machine learning -based and optimization method -based fast encoding. For the first part, we propose a feature reduction method by eliminating some features which correlate with other selected features in order to save the computational time of the exhausted RDO search.

For the second part, we present a GA-based fast CU encoding for inter-coding of HEVC intending to save the computational complexity of HEVC. We study the CU partitioning procedure and formulate it as an optimization problem. Then, good CU partitioning patterns of each CTU are searched by utilizing a simple optimizer, called GA. Nowadays, due to the higher frame rate of video sequences such as 50-60 frames per second (fps) and up to 120 fps for HD and UHD videos, respectively, the temporal correlation between consecutive frames is extremely high. The temporal correlation refers to the condition that the video data between consecutive frames of a video sequence are temporally correlated under the same background scene with the same moving objects. In details, for 60 fps video sequence, the time intervals between two, four, six, and eight consecutive frames are 0.03, 0.07, 0.1, and 0.13 second, respectively. Due to these small intervals, it is possible to share partitioning patterns of one frame to its consecutive frames without severely affecting the quality. Therefore, frame level partitioning pattern sharing is one of our contributions to further lower the computational burden of HEVC under a comparable video quality. In order to be suitable for both low to high frame rate (24 to 120 fps) and to follow a group of pictures (GOP) structure, a small GOP size 4 is reasonably utilized as a sharing range $N$, *i.e.*, $N$ is

4. As shown in Figure 1.1, the temporal correlation is relatively high between four consecutive frames since the motion information between them is low. Key frames usually use a low quantization parameter (QP) value to get a higher quality compared to other frames. Therefore, the CU partitioning patterns for the only key frame of every GOPs are searched by GA. Then, we share the CU partitioning pattern of each CTU at the key frame $f_{N\times n}$ to the collocated CTU at $(N-1)$ consecutive frames $f_{N\times n+j}$, where $n \in \{1, 2, 3, \ldots\}$ is the GOP number, and $j \in \{1, 2, 3\}$ is the displacement between the key frame and consecutive frames.

**Figure 1.1: Four consecutive frames of sequence "PartyScene" ($N = 4$).**

Our main contribution of GA-based fast encoding approach is two-fold.

1. We design a reasonably effective fitness function for GA which defines the correlation between CTU partitioning and RD performance.

2. Due to the possibility of a highly temporal correlation within consecutive frames, we share the CU partitioning patterns of a key frame with other frames to further save the computational time of the RD cost calculation.

This thesis is divided into five chapters such as introduction, background and literature review, proposed method, experimental results, and conclusion and future works. Chapter 2 introduces HEVC, the quadtree-based CU partitioning, and GA. Chapter 3 and 4 describe the detailed proposed method and evaluation of the proposed method, respectively. Finally, Chapter 5 concludes our thesis.

# Chapter 2

# Background and Literature Review

## 2.1 Background

### 2.1.1 Overview of HEVC

Due to the development of computing on multimedia data and the extreme distribution of video over the Internet with several purposes, the storage space and network bandwidth for uncompressed raw video are high. To reduce and remove redundant video information with a negligible distortion on the visual quality, an effective video compression technique can be used so that compressed digital video can be effectively stored on computer storage space and efficiently distributed over a network. The International Telecommunications Union (ITU) and the International Standardization Organization/International Electrotechnical Commission (ISO/IEC) are two dominant standardization organizations to emerge video coding standards for real-time video communication and distribution or broadcast of video content. H.261 and H. 263 were standardized by the ITU-T, MPEG-1 and MPEG-4 Visual were produced by ISO/IEC and the H.262/MPEG-2 Video was jointly produced by two organizations.

The Joint Video Team (JVT), which consists of both the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), was established to standardize the next video coding generation called H.264/MPEG-4 Advanced Video Coding (AVC) standards. Figure 2.1 depicts the block diagram of a hybrid video encoder for H.264/AVC standard. Due to the new features of H.264/MPEG-4 AVC, half of the bit rate of MPEG-2 can be reduced under the same perceptual quality.

For high-definition (HD) videos, prior standards and H.264/MPEG-4 AVC can be applied to store and transmit that videos. Because of the increasing demand of the HD video and the rising attention in the UHD, ITU-T and ISO/IEC have created an advanced joint group called the Joint Collaborative Team on Video Coding (JCT-VC) to invent a new video coding standard called High Efficiency Video Coding (HEVC). HEVC can get

the compressed video with the half-size bit-rate reduction and the same visual quality of H.264/MPEG-4 AVC. Figure 2.2 shows a block diagram of a block-based hybrid video encoder with some characteristic ingredients of HEVC regarding its novel block partitioning concept.

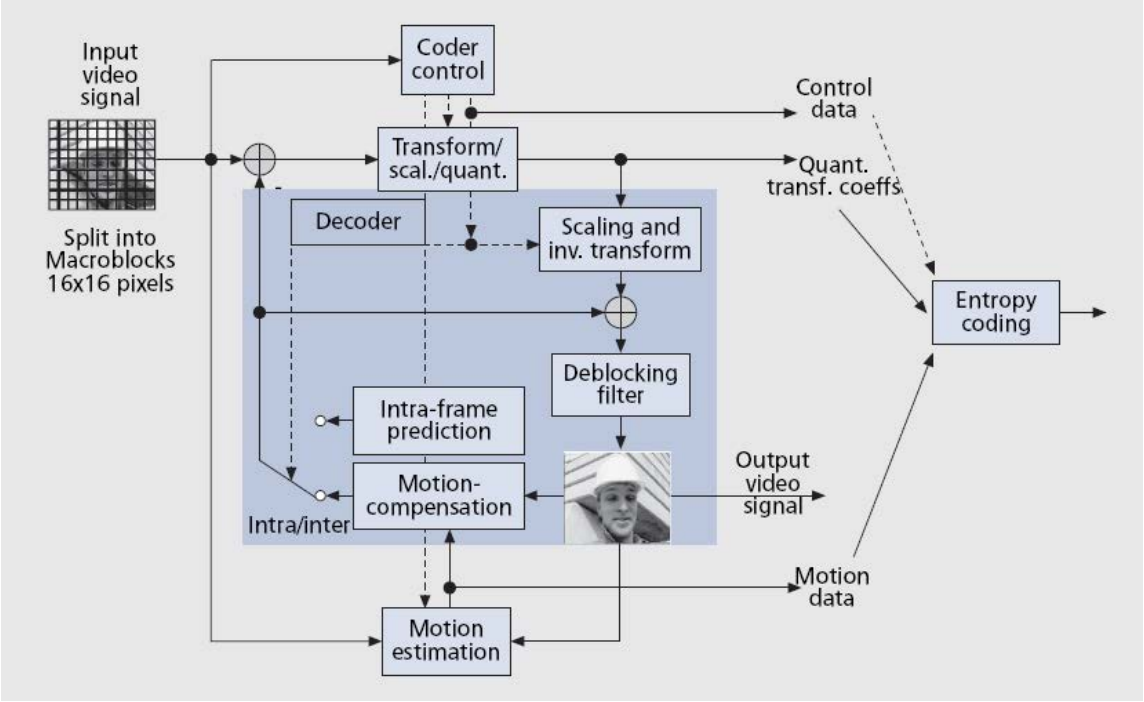



**Figure 2.1: Block diagram of an H.264 encoder.**

ITU-T and ISO/IEC are the main standardization bodies which have standardized all HEVC's antecedent standards in many years. They have utilized a 16×16 macroblock as a basic processing unit in HEVC's antecedent. Each frame is split into macroblocks. In the 4:2:0 chrominance subsampling formats, there are one 16×16 block of luma components to represent brightness and two 8×8 blocks of chroma components to refer color in each macroblock. Therefore, the macroblock is the largest block size to indicate the predicted information of intra-frame or inter-frame prediction in previous video coding standards.

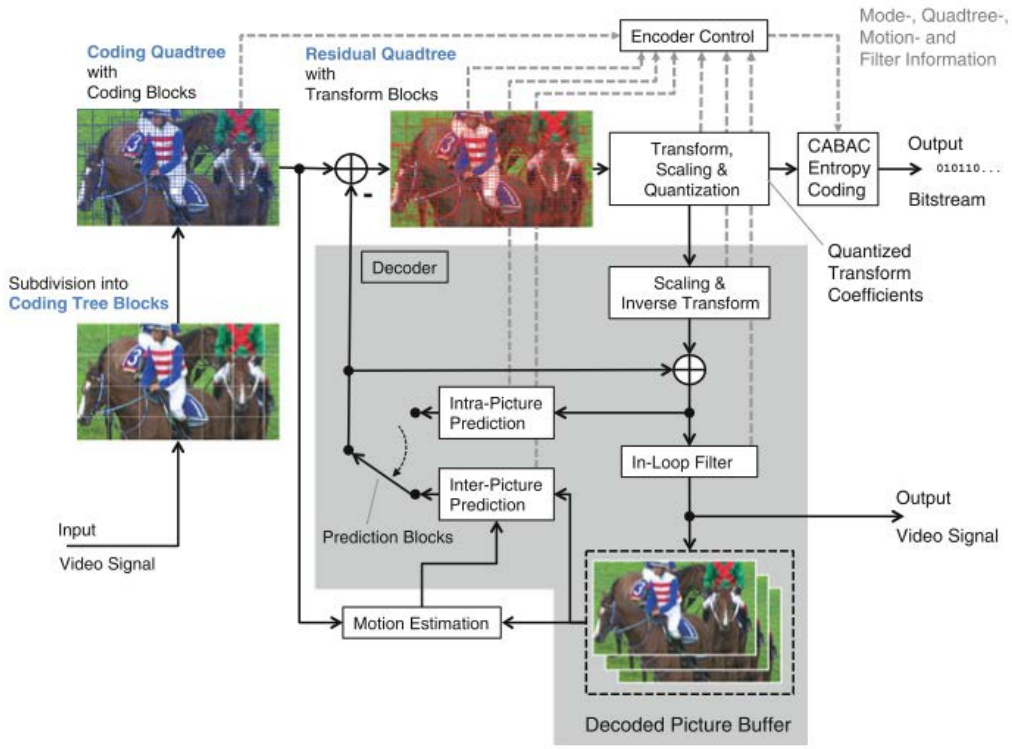
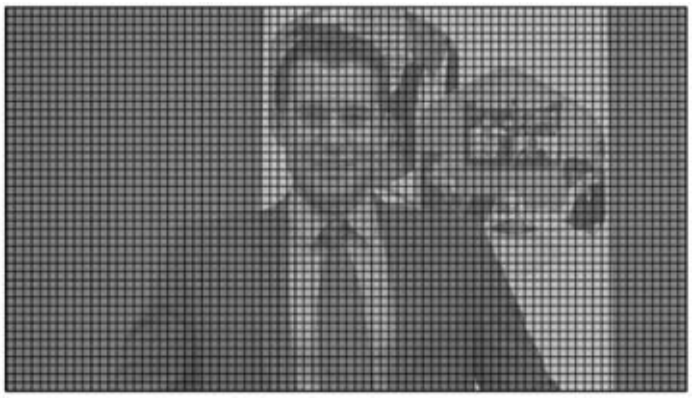



**Figure 2.2: Block diagram of an HEVC encoder with built-in decoder (gray shaded).**

However, typical HD and UHD videos have many larger frame regions than the macroblock, and those regions can represent the same moving information. If the macroblock is used as a basic processing unit for typical HD and UHD videos, a large number of bits are required to signal the prediction information. Correspondingly, the transform block size is bigger than the macroblock size. Therefore, HEVC supports a larger block size as a basic processing unit called Coding Tree Block (CTB) for intra-frame or inter-frame prediction and transform coding. For a non-monochrome video format, one luma CTB plus its two associated chroma CTBs and the corresponding syntax are combined to form the primary processing unit, called CTU. Figure 2.3 illustrates the frame partitioning from $1280 \times 720$ luma samples into $16 \times 16$ macroblock sizes and $64 \times 64$ CTU sizes. It may be concluded that $64 \times 64$ size of CTU covers a large region of a frame that can be characterized by the same motion parameters so that the largest CTU size can support a more appropriate representation. Therefore, in HEVC, the primary processing units are partitioned as large as $64 \times 64$ luma samples. Although a large block size is effective for high resolution video, it is not a good choice for low resolution video.



(a)

(b)

**Figure 2.3: Demonstration of the partitioning of a frame with 1280 × 720 luma samples into (a) macroblocks. (b) CTU.**

To be compatible with both high and low resolution videos, HEVC can flexibly partition the video frames into several square CTUs of $2^L \times 2^L$ samples, where $L \in \{4, 5, 6\}$. The encoder flexibly chooses a suitable value of $L$ for intended application to have the best trade-off between coding performance and cost such as memory storage, encoding time, and delay. However, using a larger block for selecting whether intra-mode or inter-mode at the prediction stage cannot guarantee to get a good RD performance for prediction stage. To achieve a better coding efficiency, HEVC introduced a new basic processing unit, called coding unit (CU) and a flexible quadtree partitioning from CTU to CU. Therefore, CU size can be 64×64, 32×32, 16×16 and 8×8 at depth 0, depth 1, depth 2 and depth 3, respectively.

For each CU, a prediction mode is indicated in the bitstream. The prediction mode represents whether the intra-frame or inter-frame prediction is selected to encode CU. If the intra-frame prediction is selected, one of the possible 35 modes for intra-frame prediction has to be chosen for the CU and signaled in the bitstream. If a CU is coded using inter-picture prediction, the CUs can be further split into prediction blocks (PUs). A PU is a block of samples which includes the same motion parameters for inter-frame prediction. To partition from a CU into PUs, HEVC provides eight different modes. As described in Figure 2.4, a CU can either be coded as a PU or it can be partitioned into 2 or 4 rectangular PUs. $2N \times 2N$ mode represents to partition the whole CU into a single PU. $N \times N$ mode represents to split a CU into four PUs and the resulting PUs are square in shape and same in size. For every $2^M \times 2^M$ CU and PU, each unit consists of one $2^M \times 2^M$ luma coding block (CB) and prediction block (PB) and two corresponding $2^{M-1} \times 2^{M-1}$ chroma CBs and PBs, respectively, if the chroma sampling format is 4:2:0.

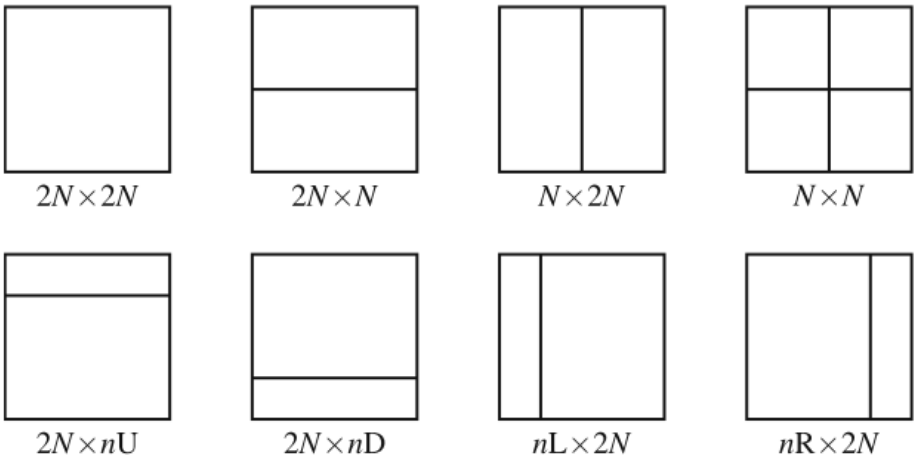



**Figure 2.4: Available partitioning modes for partitioning a CU into 1, 2, or 4 PUs.**

### 2.1.2 The Hierarchical Quadtree-based CU Partitioning

As we mentioned above, the CTU partitioning structure is one of the most important coding features to the HEVC standard. The default size of a CTU is $64 \times 64$ samples, and each CTU can be a solo CU or can be partition into four sub-CUs and then each sub-CU can be additionally divided into four sub-CUs until the maximum CU depth reaches. So, the CU size (depth level) can be in the range of $64 \times 64$ (Depth 0) to $8 \times 8$ (Depth 3). Additionally,

each CU can be further divided into smaller PU with various eleven partition modes, i.e., a SKIP/MERGE, eight Inter, and two Intra partition modes as shown in Figure 2.5 and the PU with the minimum RD cost will be selected. After getting the best partition mode for each CU, the optimal CU sizes are decided based on recursive RD cost comparisons.

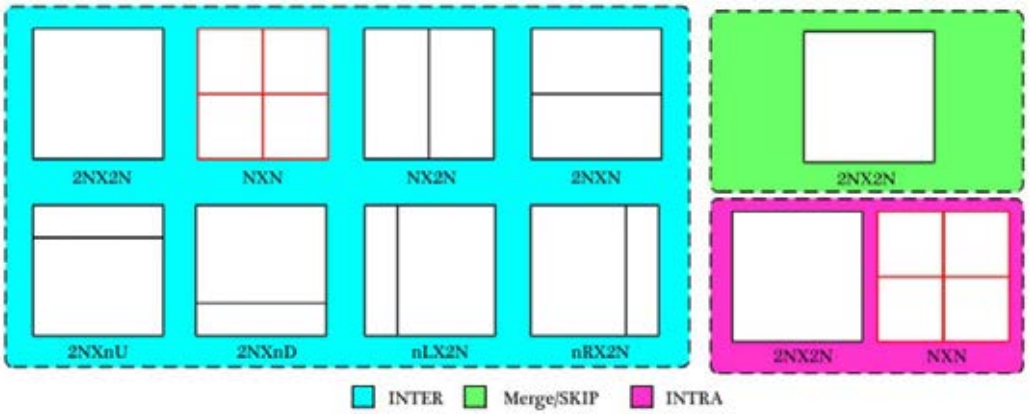



**Figure 2.5: Eleven PU partition modes.**

To define CU size or depth, HEVC starts a trial encoding which includes two main functions called the RD cost calculating/checking and comparison in a top-down and bottom-up manner, respectively, as mentioned in Section 1.1. Figure 2.6 illustrates the RD cost checking and comparison process between a parent/current CU and its children/sub-CUs.

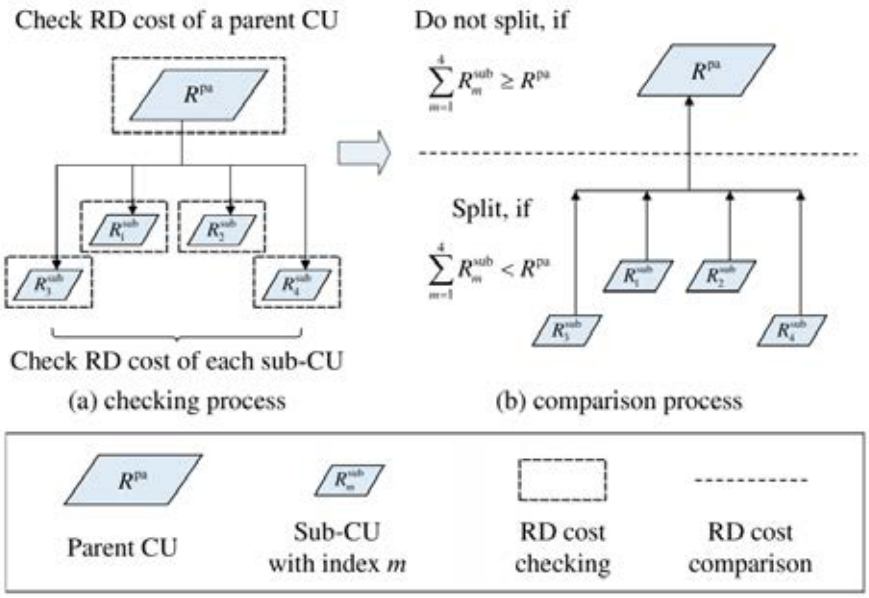



**Figure 2.6: Illustration of RD cost calculating and comparing between a parent/current CU and its children/sub-CUs.**

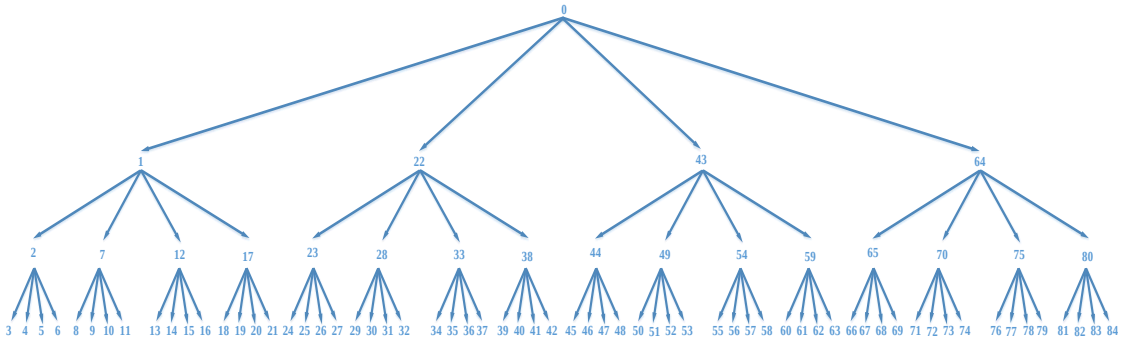



**Figure 2.7: The order of RD cost calculation for 85 CUs of a 64 x 64 CTU.**

In the top-down RD cost calculation of a 64×64 CTU, the RD costs for all possible 85 CUs are calculated in a preorder traversal of the quadtree, as shown in Figure 2.7, if the maximum CU depth is 3. In details, there are 1, 4, 16 and 64 CUs at depth 0, depth 1, depth 2 and depth 3, respectively, and the total number of CU is $\sum_{i=0}^{3} 4^i = 85$ CUs. After getting the RD costs for four children CUs of every one of parent CUs, HEVC turns to the RD cost comparison to decide whether a parent CU is split or not by comparing the RD cost of splitting and non-splitting conditions of parent CU. In Figure 2.6, the RD cost of a current/parent CU is denoted as $R^{pa}$, and the RD costs of its children/sub-CUs are denoted as $R_m^{sub}$ , where $m \in \{1, 2, 3, 4\}$ is the index of each child CU. Afterward, the RD cost of a parent CU ($R^{pa}$) and the total RD cost of four sub-CUs ($\sum_{m=0}^{3} R_c^{sub}$) are compared to decide whether a parent CU should be partitioned. Then, HEVC switches to the RD cost calculation or performs comparison again depending on the position of parent CU. Therefore, there are 85 calculations and 21 comparisons in the top-down RD cost calculation and bottom-up RD cost comparison of a 64×64 CTU, respectively. After finally comparing a root CU at depth 0 with its four children CUs at depth 1, the best CU quadtree structure of a CTU as shown in Figure 2.8 with the lowest RD cost is chosen among 83,522 possible quadtree structures.

Figure 2.9 shows the CU partitioning pattern of frame representation of picture order count (POC) 40 of sequence "BlowingBubbles" searched by an exhaustive RDO search of HM version 16.5 (HM16.5). The trial encoding of HEVC finds the best CU partition structure of each CTU after an exhaustive RDO search. Therefore, choosing an optimal

CU partitioning structure can be modeled as an optimization problem and can be solved by a lightly suitable optimization tool to search through a space of possible CU partition solutions. For the small space optimization process, traditional comprehensive techniques are proper to find the solution [24]. However, the techniques based on artificial intelligence (AI) are efficient for a vast search space and GA is one of AI techniques to search a good solution efficiently .

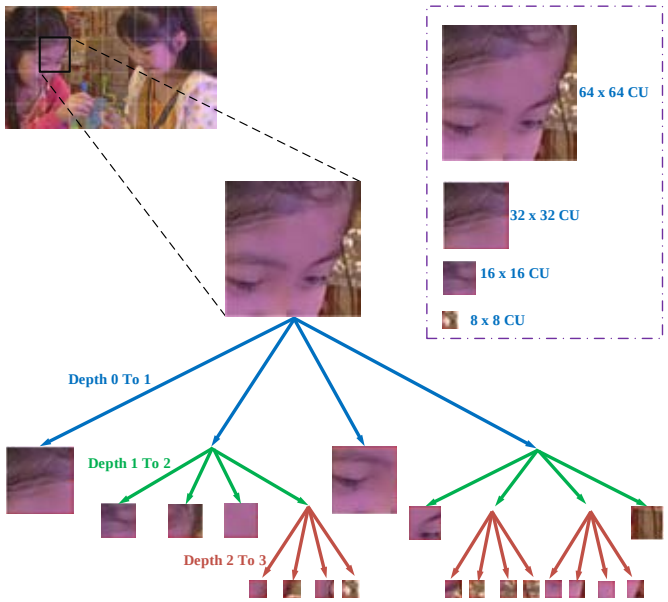



**Figure 2.8: Quadtree partitioning from CTU to CU.**

(a) (b)

**Figure 2.9: Quadtree partitioning of a CTU into CU based on recursive RD cost comparsion. (a) Subdivision into 64×64 CTUs. (b) Coding quatdree with CUs.**

### 2.1.3 Genetic Algorithm

For implementing GA for a particular problem, there are four components need to be considered as follows:

1. A genetic data structure including genes, called chromosome, is an efficient candidate solution to our problem.
2. A fitness function which measures how the chromosome fits to our problem.
3. Three genetic operators such as selection, crossover, and mutation to produce new offspring.
4. Some basic parameter values such as population size, crossover, and mutation probabilities.

By using these four components, GA has three major processes for creating a random initial population of chromosomes, calculating fitness value for each chromosome in the current population, and producing a newly better population until the optimization criteria meet, as shown in Figure 2.10. The termination point of a GA is an important factor where it stops GA with the best chromosome. Normally, there are three possible termination conditions described as follow:

1. The fitness value of the best chromosome has achieved a predefined value.
2. The best chromosome of the current population and previous population are still the same for $G$ generations.
3. The total number of generations has reached a predefined count.

The termination criteria highly depend on the problem domain and it is defined by trying all possible options to get the best termination criteria and point.

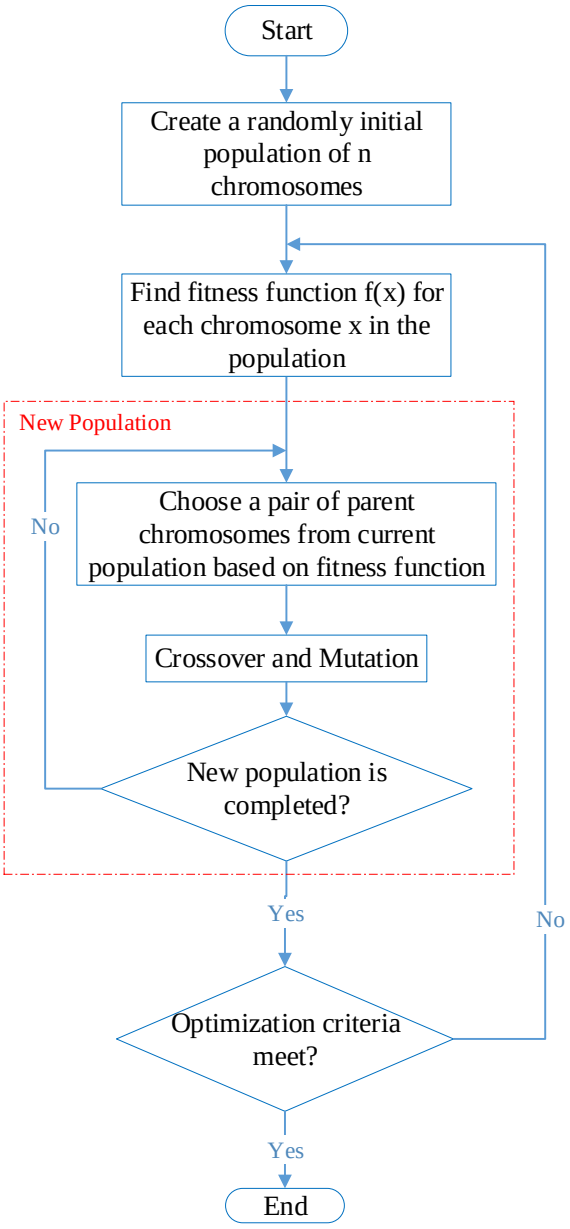



**Figure 2.10: Flowchart of GA.**

## 2.2 Literature Review

To efficiently save the computational cost of HEVC, researchers explored several efficient methods on the fast algorithms for HEVC. In general, the existing methods can be categorized into two parts: intra-coding and inter-coding, as we mentioned in Chapter 1.

### 2.2.1 Fast Algorithms for Intra-coding of HEVC

Cho *et al.* [4] applied a Bayes decision rule to decide whether CU is early split or not and whether CU is early pruned or not based on rough RD cost and full RD cost respectively. Min *et al.* [5] proposed a CU depth prediction based on global and local edge complexities of parent CU and four children CUs. To catch up on the current trend, Li *et al.* [6] utilized deep convolutional neural network (CNN) approach for the CU partitioning procedure instead of exhaustive RDO search for HEVC intra-coding. In order to reduce the further computational complexity of intra-coding, Zhang *et al.* [7] focused on quickly deciding intra-mode and CU depth by utilizing gradient-based block direction detection and Support Vector Machine (SVM). They presented fast intra-mode decision and CU size decision for decreasing the computational time of intra-coding of HEVC under a comparable RD performance. In order to get a quick decision for intra-mode, they proposed a method based on the gradient for reducing the candidate modes for rough mode decision (RMD) and RD optimization (RDO). In order to get a fast decision for CU size, firstly, the homogeneous CUs was early stopped to partition. Then, two linear SVM were utilized to make the decisions of early CU splitting and termination for the remaining CUs. These two SVM classifiers utilized the difference of depth and ratio of Hadamard transform-based costs (HAD costs) as their features.

### 2.2.2 Fast Algorithms for Inter-coding of HEVC

In order to save the computational time for inter-coding of HEVC, the researches reported in [8]–[12] were statistically studied the relationship between some important features and CU partitioning. Shen *et al.* [8] proposed a fast CU depth decision based on statistical analysis and spatiotemporal correlation. After these statistical analyses, they determined an adaptive CU depth range by skipping some uncommon depth levels of neighboring CUs and co-located CUs of the previous frame. Then, they skipped ME at high CU depth based on three early termination method based on motion similarity, SKIP mode, and RD threshold. In [9], an adaptive threshold based on the RD distribution of the previous frame was utilized whether the parent CU of the current frame is split or not. In [10], the quadtree traversal can be switched from an originally top-down order to an inversely bottom-up or-

der. As a result, the number of mode testing of parent CU at lower depth can be reduced by utilizing the mode information of its four children CUs at higher depth. In [11], Shen *et al.* utilized Bayesian decision rule with effectively relevant feature candidates to avoid unnecessary CU partitioning. Though these threshold-based CU size decision approaches can significantly save high computational time, they may not be applicable for all sequences since their statistically hard threshold and spatiotemporal distribution.

In [13]–[20], learning-based fast algorithms have utilized their efficient learning capability for complexity reduction on the quadtree-based CU partitioning of HEVC. To save the computational load of CU partitioning, Shen *et al.*[13] early terminated CU partitioning process with the aids of SVM and feature selection based on a wrapper approach. Also, they introduced different weights to the training of SVM in order to decrease the effect of outliers and RD loss when a misclassification happens. To early terminate unnecessary CUs, PUs, and TUs partition, Correa *et al.*[14] built decision trees by using a free open-source data mining (DM) tool. To create a robustly superior learning model for the joint SVM classifier, Zhang *et al.*[15] originated an optimally weighted parameter determination method and used offline training mode with CU partition-related features. They proposed the quadtree partitioning pattern of CTU as a three-level classification process, and two three-decision classifiers were designed to control the risk of false prediction. Mainly, the feature extraction process should be low complexity to avoid the computation overheads and the CU size are related with the video content' texture, motion information, context of spatiotemporal neighboring information, etc. According to these two principles, they considered the nine features for deciding CU size and nine features can be categorized into four groups: information of the SKIP/MERGE mode of current CU, motion information, context information and a quantization parameter. Heindel *et al.* [16] used support vector machine (SVM) with offline training mode to make a decision whether CU is partitioned or not. They built several SVM models for all selected video sequences for their experiment by utilizing the training data from all the other video sequences. Due to the usage of training data of other video sequences, the feature values from training data and testing data are quite different. This is one of the drawbacks of offline training mode of learning approach that may lead to decrease the prediction accuracy significantly. Additionally, due to the

hard threshold for classifying uncertain decision and the running of exhaustively original RD optimization for uncertain condition, the computational time save may not achieve significantly.

To improve the prediction accuracy, CU size decision based on online SVM training has implemented in [17] to firstly consider the adaptive regulation parameters due to the difference between False Positive (FP) and False Negative (FN) rates, different weights for training samples to deduct the negative effect of outliers, and the risk area. In order to terminate the CU partitioning process early, Kim *et al.*[18] applied Bayesian decision rule by jointing online and offline learning to train the selected frames for each scene and get loss matrix after training several frame sequences, respectively. In [19], Zhu *et al.* reduced the computational burden of CU and PU partitions by utilizing binary SVM classifiers and multi-class SVM classifiers with both online and off-line learning modes. However, SVM based CU size decision approach [17] considered feature selection based on misclassification cost and did not consider the correlation between features.

Additionally, all mentioned learning-based approaches have considered the CU partitioning problem of HEVC as a classification problem or decision problem. As a result, an original HM will be triggered instead of the original RDO process if the prediction accuracy is not enough to use the prediction output. Therefore, these approaches may not significantly reduce the computational burden for an inaccurate situation of prediction results. Finding an optimal CU partition pattern from all possible outcomes can be modeled as an optimization problem and can be solved by a simply effective optimizer. All fast algorithms mentioned above have not formulated a CU partition as an optimization problem. Therefore, a simple fast CU encoding based on GA is supposed to be proposed and implemented to reduce the computational burden of quadtree-based CU partitioning.

# Chapter 3

# Fast Coding Algorithm for High Efficiency Video Coding

As we mentioned above, there are two parts in our research work: machine learning -based [25] and optimization -based fast encoding [26].

## 3.1 Reducing Redundant Feature from Fuzzy SVM-Based Coding Unit Decision in HEVC

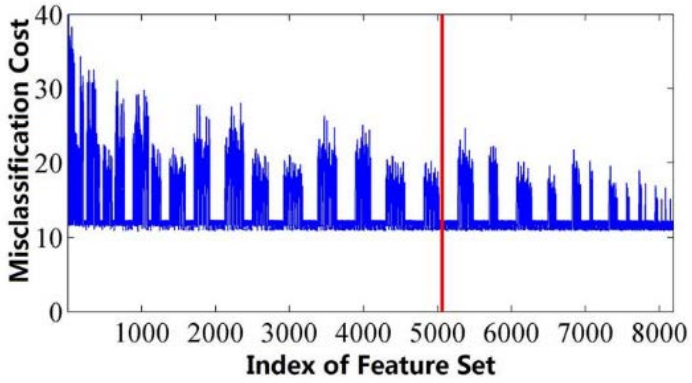
For ML-based prediction approach, choosing fruitful and proper features is the major contribution of a classifier and using those features efficiently can reduce the training time and the required storage. In order to achieve that goal, there are three issues as explained in the following paragraphs.

The first issue is how to avoid the computational overhead of the feature extraction. To avoid this, [17] utilized some ready-made features, except one temporal domain features, after checking with the most common modes. Therefore, the feature extraction stage does not spend too much processing time. The second issue is how to select the representative features that can distinguish different classes. Generally, the depth of CU can vary due to the image texture, motion information, spatio-temporal context information, etc. Based on this issue, [13], [15], [17], and [19] considered several effective features from the prediction error of the temporal domain, spatial domain and by-product information of the current block. The third issue is how many numbers of features will be used as small as possible to keep a low computational time at the prediction stage. To concern the third issue, in [13], eleven possible candidates were firstly proposed and then evaluated to get an effective feature subset by using the wrapper approach based on F-score. And finally, five features were selected to make a trade-off between accuracy and additional complexity introduced by feature extraction and prediction. In [17], they firstly considered thirteen features as potential candidates. They exhaustively checked all possible number of feature combinations

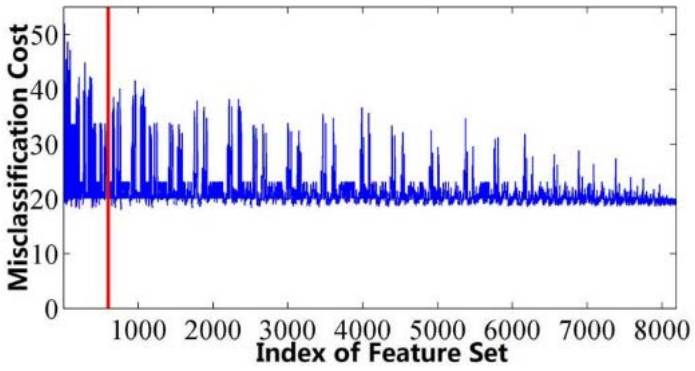
C and the value of C is $2^{13} - 1 = 8191$ feature combinations if the number of features is 13. Table 3.1 shows example feature combinations (feature sets) with their index and binary patterns which represent the position of the features and whether the feature is included in feature combination or not. Finally, an optimal feature set with minimum misclassification cost for each depth level was selected based on empirical results shown in Figure 3.1. It can be found that the indexes of optimal feature sets with the lowest misclassification cost are 5065, 598 and 8187 for classifier C1, C2, and C3, respectively.

**Table 3.1: Example Feature Combinations of FuzzySVM [17].**

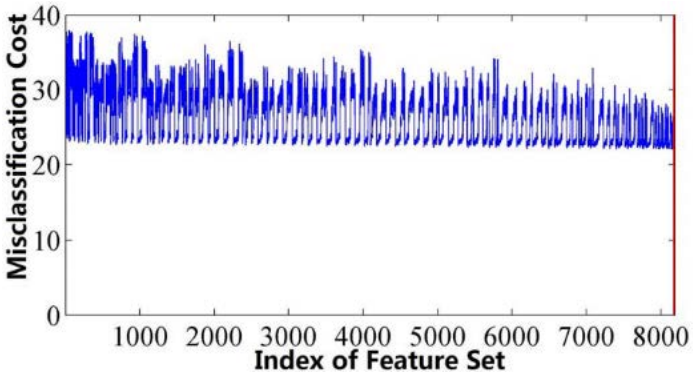
| Index | Feature | Binary Representation |
|---|---|---|
| 1 | $x_{SAD}$ | 1 0000000 00000 |
| 2 | $x_{RDCost}$ | 0 1000000 00000 |
| 3 | $x_{SkipFlag}$ | 0 0100000 00000 |
| . | . | . |
| . | . | . |
| . | . | . |
| 14 | $x_{SAD}, x_{RDCost}$ | 1 1000000 00000 |
| . | . | . |
| . | . | . |
| . | . | . |
| 598 | $x_{RDCost}, x_{SkipFlag}, x_{Distortion}, x_{Bits}$ | 0 1111000 00000 |
| . | . | . |
| . | . | . |
| . | . | . |
| 5065 | $x_{SAD}, x_{RDCost}, x_{Distortion}, x_{QP}, x_{CBF\_SKIP}, x_{Depth}, x_{CBF\_NB}$ | 1 1010011 00011 |
| . | . | . |
| . | . | . |
| . | . | . |
| 8187 | $x_{SAD}, ..., x_{Bits}, x_{CtxSkipFlag}, ..., x_{CBF\_NB}$ | 1 1101111 11111 |
| . | . | . |
| . | . | . |
| . | . | . |
| 8191 | $x_{SAD}, ..., x_{CBF\_NB}$ | 1 1111111 11111 |




(a)




(b)




(c)

**Figure 3.1: Finding optimal three feature sets based on misclassification. (a) C0. (b) C1. (c) C2.**

Therefore, each optimal features set shown in Table 1.2, getting from empirical results, may have some correlated features as we mentioned before. Because of this possible issue

for some correlated features, firstly, some correlated features of the optimal feature sets are determined in our work. In detail, according to the Eq. (1.1), there are two correlated features ( $x_{RDCost}$ and $x_{Distortion}$), three correlated features ( $x_{RDCost}$, $x_{Distortion}$ and $x_{Bits}$) , two correlated features ( $x_{RDCost}$ and $x_{Bits}$) for classifier C0, C1, and C2, respectively. Therefore, there may be possible to eliminate distortion or/and bit features when the RD cost feature is already included in the optimal feature set to save the time consumed by one or two redundant features. In our work [25], distortion and bits features are eliminated from three optimal feature sets. Table 3.2 shows the remaining selected features of different classifiers after eliminating two features.

**Table 3.2: Selected Features for different classifiers [25] after eliminating redundant features of [17].**

| Index | Candidates | Classifier C0 | Classifier C1 | Classifier C2 |
|---|---|---|---|---|
| 1 | $x_{SAD}$ | √ | | √ |
| 2 | $x_{RDCost}$ | √ | √ | √ |
| 3 | $x_{SkipFlag}$ | | √ | √ |
| 4 | $x_{CtxSkipFlag}$ | | | √ |
| 5 | $x_{QP}$ | √ | | √ |
| 6 | $x_{CBF\_SKIP}$ | √ | | √ |
| 7 | $x_{MergeFlag}$ | | | √ |
| 8 | $x_{MV}$ | | | √ |
| 9 | $x_{Partition}$ | | | √ |
| 10 | $x_{Depth}$ | √ | | √ |
| 11 | $x_{CBF\_NB}$ | √ | | √ |

## 3.2 GA-Based Fast CU Partitioning

As we mentioned in Section 2.2.2, all learning-based approaches have proposed the CU partitioning problem of HEVC as a classification or decision question. Therfore, after studying the procedure of CTU partition, we firstly consider quadtree-based CU partitioning as an optimization problem and utilize GA to find a good CU partition pattern for each CTU [26].

### 3.2.1 Proposed Data Structure

For the first component of GA, the CU partitioning pattern of a 64×64 CTU is considered as a chromosome of quadtree-based CU partitioning problem as shown in Figure 3.2.

Three hierarchical levels of chromosome structure are proposed to represent three CU depth levels of CU partitioning of HEVC. Assume that the maximum CU depth is 3, our proposed 21-bit data structure $C$ for GA is represented as follows:

$$
\begin{aligned}
C &= a \quad b_0 b_1 b_2 b_3 \quad c_0 c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{13} c_{14} c_{15} \\
a &= \begin{cases} 0, & if\ CU\ is\ not\ split \\ 1, & otherwise \end{cases} \\
b &= \begin{cases} null, & if\ a = 0 \\ 0, & if\ CU\ is\ not\ split \\ 1, & otherwise \end{cases} \\
c &= \begin{cases} null, & if\ a = 0\ or\ corresponding\ parent\ b = 0 \\ 0, & if\ CU\ is\ not\ split \\ 1, & otherwise \end{cases}
\end{aligned}
\tag{3.1}
$$

, where $a$, $b$ and $c$ are genes to represent the splitting decisions for depth 0, depth 1 and depth 2, respectively. The possible values for $a$ is 0 (non-splitting) and 1 (splitting). The

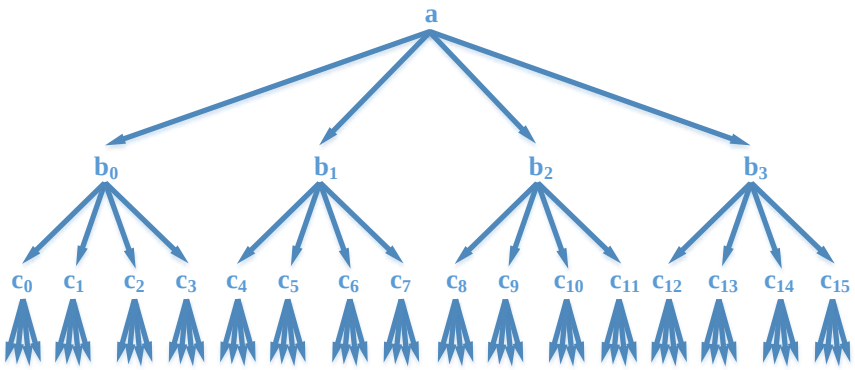



**Figure 3.2: Hierarchical chromosome of a 64×64 CTU.**

possible values for $b$ are null if a is 0, 0 (non-splitting) and 1 (splitting). The possible values for $c$ are null if its corresponding parent $b$ is 0, 0 (non-splitting) and 1 (splitting). It should be noted that the proposed data structure is composed of a group of dependent

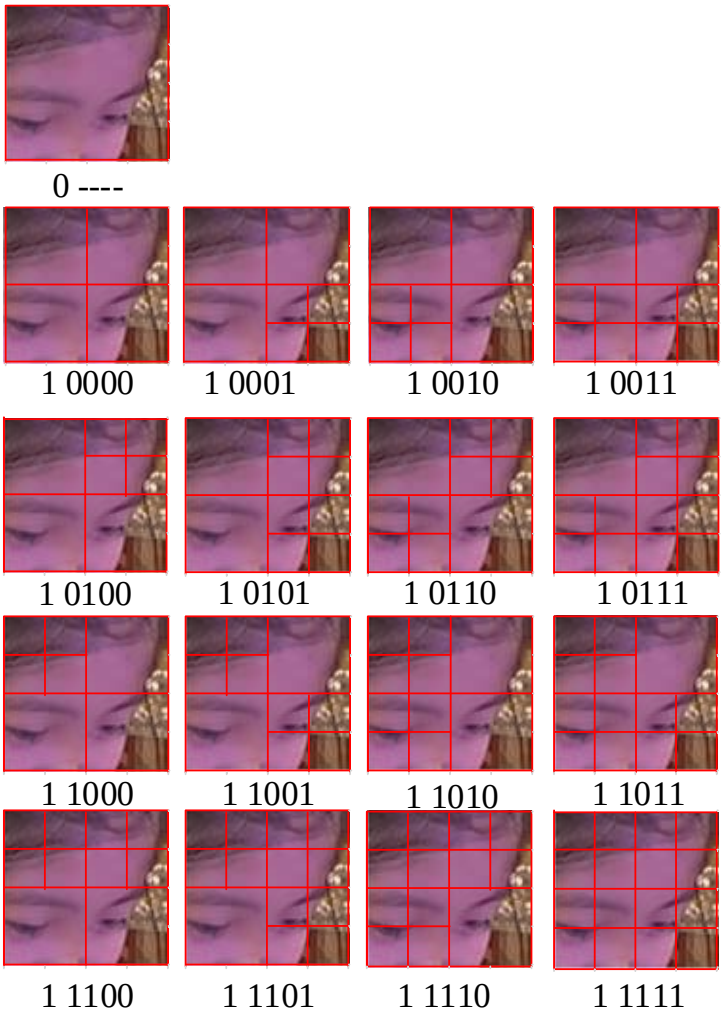



**Figure 3.3: Possible CU partition patterns of a 64×64 CU (Maximum CU Depth = 2).**

genes. Therefore, the total number of possible CU partitioning patterns $P$ is calculated as in (3.2),

$$P = (2^4 + 1)^{(d-1)^2} + (d \, mod \, 2) \tag{3.2}$$

, where $d \in \{1, 2, 3\}$ is the maximum CU depth and *mod* is the modulo operation for finding the remainder. If the maximum CU depth is 2 and 3, the total number of possible partitioning patterns is only 17 as shown in Figure 3.3, and 83,522 even there are five genes and 21 genes to represent the CU partitioning pattern, respectively.

### 3.2.2 Proposed Fitness Function

Considering the CU size decision of original HEVC mentioned in Section 2.1.2, RD cost-oriented fitness function is reasonably proposed to select top parents of current population to create a new better population. It should be noted that the smaller the RD cost of a chromosome, the higher the chance to become a good chromosome. To save the computational time for the proposed fitness function, we analyze most common modes of the original HEVC and take advantage of those modes to calculate the RD cost of CU. As shown in Figure 3.4, we can observe that the SKIP/MERGE mode is the most selected mode for PUs with the minimum RD cost in HEVC. Therefore, the RD costs of existing CUs encoded with the most common mode SKIP/MERGE are calculated before calculating the fitness function of GA.

The data structure as shown in Figure 3.2 and the RD costs for existing CUs as shown in Figure 3.5 are utilized in order to get the fitness function of each chromosome in population using (3.3),

$$F = (1-a)RDCost_a + a\left[\sum_{i=0}^{3}(1-b_i)RDCost_{b_i} + b_i\left(\sum_{j=4i}^{4i+3}(1-c_i)RDCost_{c_j} + c_j\sum_{k=4j}^{4j+3}RDCost_{d_k}\right)\right] \quad (3.3)$$

, where $F$ is the RD cost-oriented fitness function, $a$, $b_i$, and $c_j$ are the values of one gene, four genes, and sixteen genes of each chromosome to represent the splitting decision at depth 0, 1, 2, and 3, respectively. $RDCost_a$, $RDCost_{b_i}$, $RDCost_{c_j}$, and $RDCost_{d_k}$ are the RD cost values of one CU, 4 CUs, 16 CUs, and 64 CUs at depth 0, 1, 2, and 3, respectively. The chromosome value of the splitting pattern shown in Figure 3.6 is 0 1010 0000222210002222 which represents $a\ b_0b_1b_2b_3\ c_0c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}c_{14}c_{15}$. The gene 0, 1, and 2 means non-splitting, splitting, and null, respectively. The fitness value of the splitting pattern shown in Figure 3.6 are $\sum_{j=0}^{3}RDCost_{c_k} + RDCost_{b_1} + \sum_{k=16}^{19}RDCost_{d_k} + RDCost_{c_{10}} + RDCost_{c_{11}} + RDCost_{c_{12}} + RDCost_{b_3}$.

(a) (b)

(c) (d)

(e) (f)

**Figure 3.4: Usage of MERGE/SKIP mode (in green color) for PUs with the lowest RD cost in HEVC.(a) Traffic (Class A). (b) Kimono1 (Class B). (c) BQMall (Class C). (d) BlowingBubbles (Class D). (e) Johnny (Class E). (f) ChinaSpeed (Class F).**

.

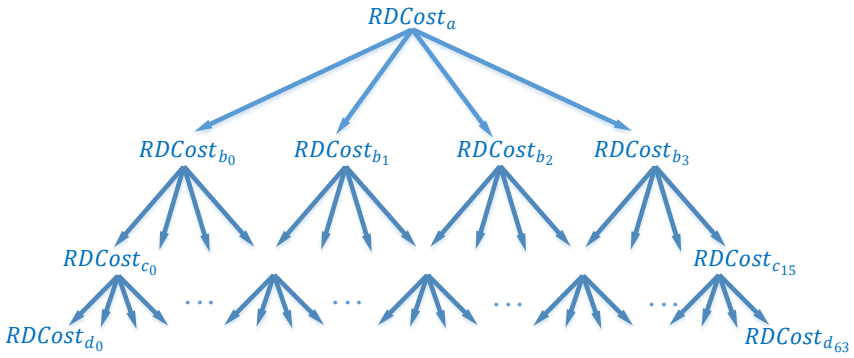



**Figure 3.5: Required 85 RD costs of a 64 x 64 CTU.**

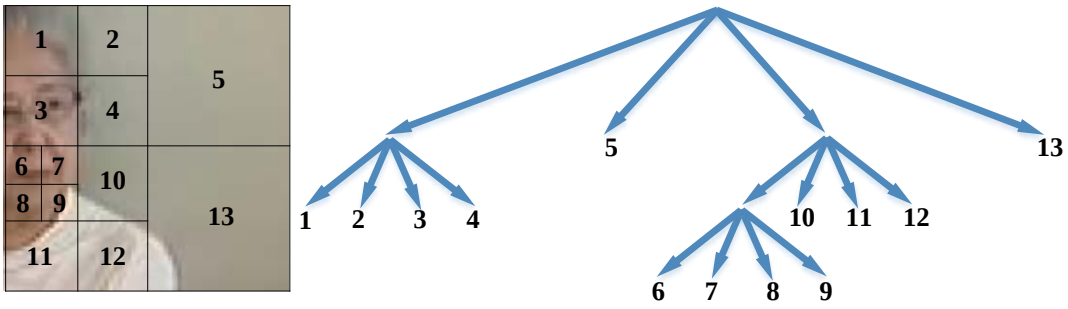



**Figure 3.6: Example splitting pattern of a CTU.**

### 3.2.3 Selection, Crossover and Mutation

After calculating the fitness function for each chromosome of the current population, individual chromosomes are selected based on their fitness function to be a good parent for the next population until the optimization criteria meet. The chromosomes with lower fitness values have a higher chance to become the parents. The total number of parents is about 10% of the total number of chromosomes. After selecting top parents, each gene of a new chromosome is created by filling a collocated gene from a randomly selected parent. To do more adaption for each chromosome, the mutation operator is applied to alter the value of genes. It should be noted that each generated chromosome is needed to check whether it is a valid chromosome or not.

### 3.2.4 Optimization Criteria

According to our problem domain, we use the second termination condition mentioned above that GA will terminate if the two best chromosomes of the current population and previous population are repetitive for $X$ times. To save the computational time for finding a good chromosome under an acceptable accuracy, the GA parameter, $X$, is assigned as 2 based on empirical results.

### 3.2.5 Genetic Algorithm with the Proposed Chromosome and Fitness Function

The detailed process of GA with the effective chromosome structure and fitness function is described as two steps in Algorithm 1. To have a trade-off between running time and accuracy for GA, we empirically assign 50, 5 and 2 for the value of $N, M$ and $X$, respectively. By utilizing the splitting pattern from GA, the CU depth is estimated without doing an exhaustively recursive RDO search.

**Algorithm 1** GA with the proposed chromosome and fitness function

**Input:** $rdcost_a$, $rdcost_{b_i}, i \in \{0, 1, 2, 3\}$,
$rdcost_{c_j}, j \in \{0, 1, ..., 15\}$,
$rdcost_{d_k}, k \in \{0, 1, ..., 63\}$

**Output:** SplittingPattern
$a\ \ b_0b_1b_2b_3\ \ c_0c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}c_{14}c_{15}$

**Step 1: Create $N$ Initially Random Chromosomes, $C$**

$n \leftarrow 0$
**for each** $n < N$ **do**
    *initial*: $C_n \leftarrow random\_Chromosome()$
    **if** $C_n$ *is duplicate* **then**
        **goto** *initial*
    **end if**
    $F_n \leftarrow (1-a)rdcost_a + a\left[\sum_{i=0}^{3}(1-b_i)rdcost_{b_i} + b_i\left(\sum_{j=4i}^{4i+3}(1-c_i)rdcost_{c_j} + c_j\sum_{k=4j}^{4j+3} rdcost_{d_k}\right)\right]$
**end for**

**Step 2: Select Best $M$ Parents $P$ from $N$ Chromosomes and Reproduce New Chromosomes**

$sameFitnessCount \leftarrow 0$, $generation \leftarrow 0$
**while** $sameFitnessCount\ != X$ **do**
    $generation++$
    **if** $generation > 1$ **then**
        $previousMinFitness \leftarrow min\_Fitness(C)$
    **end if**
    $P \leftarrow select\_BestParents(C, M)$
    **if** $(generation > 1\ \&\&\ previousMinFitness == min\_Fitness(P))$ **then**
        $sameFitnessCount++$
    **end if**
    $n \leftarrow M$
    **for each** $n < N$ **do**
        $C_n \leftarrow reproduce(P)$
    **end for**
**end while**

### 3.2.6 Overall Algorithm

Three main parts of the proposed method to find the CU partitioning patterns for every CTUs of a key frame are RD cost calculation, CU partitioning pattern finding, and CU size prediction. The overall procedure of the proposed algorithm [26] can be summarized

as follows:

1. For the first GOP, the CU partition patterns of the first inter frame (in yellow color) and fourth frame (key frame in blue color) are searched by GA and shared with two consecutive frames of current GOP and three consecutive frames of next GOP, respectively.

2. For the next GOP, each frame are checked whether it is a key frame or not. If it is the key frame, the CU partitioning patterns of that key frame are searched by GA and shared with three consecutive frames as shown in Figure 3.7.

3. For each CTU of the key frame, there are three main parts mentioned above. Firstly, the RD costs for 85 CUs are efficiently calculated, assuming that the size of CTU is 64×64 and the maximum CU depth is 3. Secondly, the CU partitioning pattern is quickly searched by using the efficient RD costs of 85 CUs and a simple GA. Finally, the CU size is predicted by using the CU partitioning pattern.

4. For each CTU of the non-key frame, the CU sizes are predicted by using the CU partitioning pattern of collocated CTU of key frame.

The quadtree CU partitioning flowchart of the original HM, FuzzySVM[17], and the proposed method are shown in Figure 3.8, Figure 3.9, and Figure 3.10, respectively.

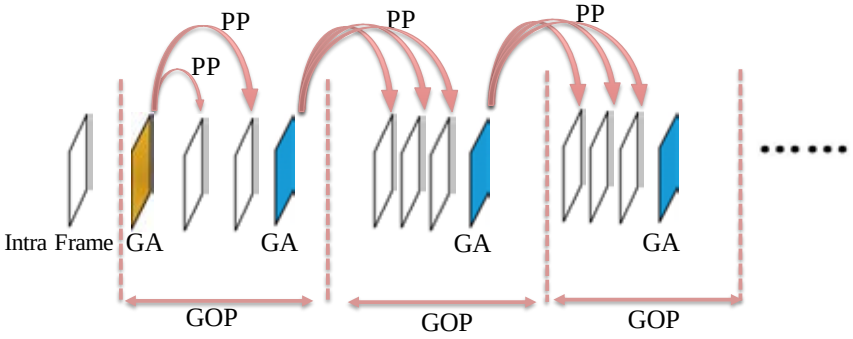



**Figure 3.7: Illustration for sharing partitioning pattern.**

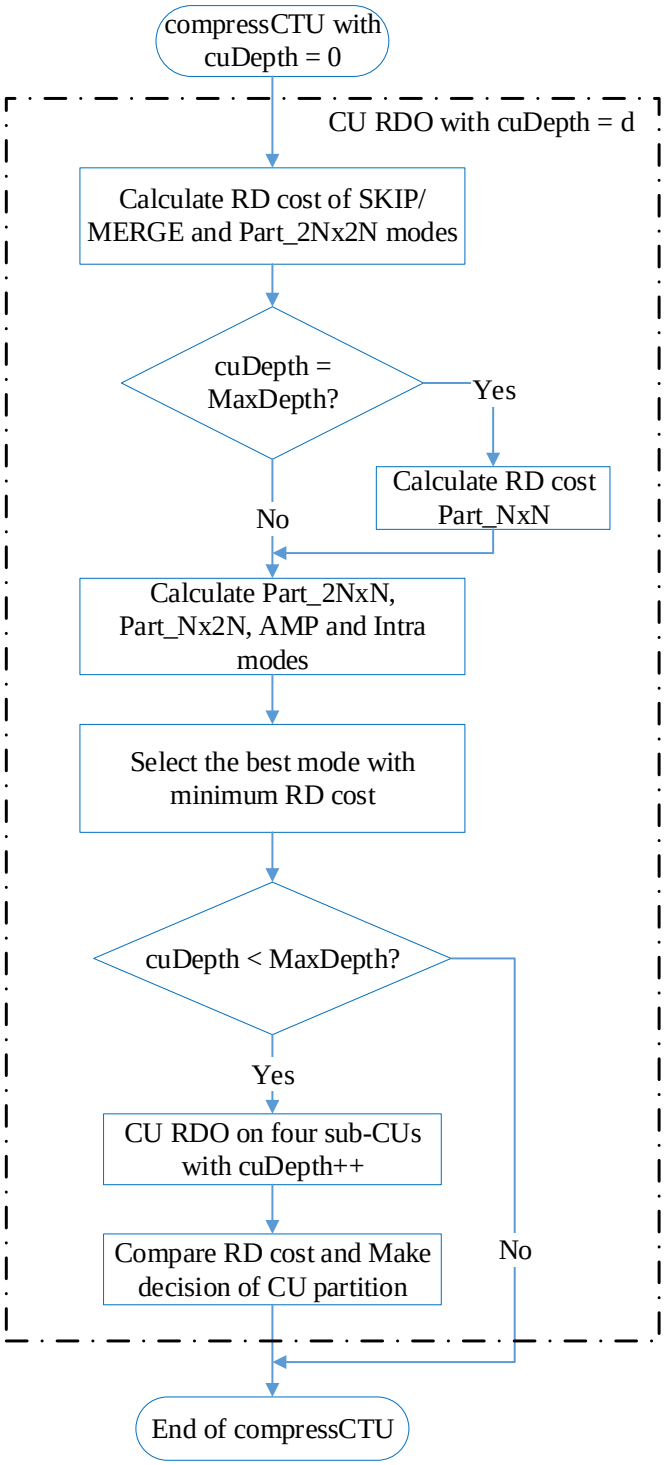



**Figure 3.8: Flowchart for quadtree-based CU partitioning of HM16.5.**

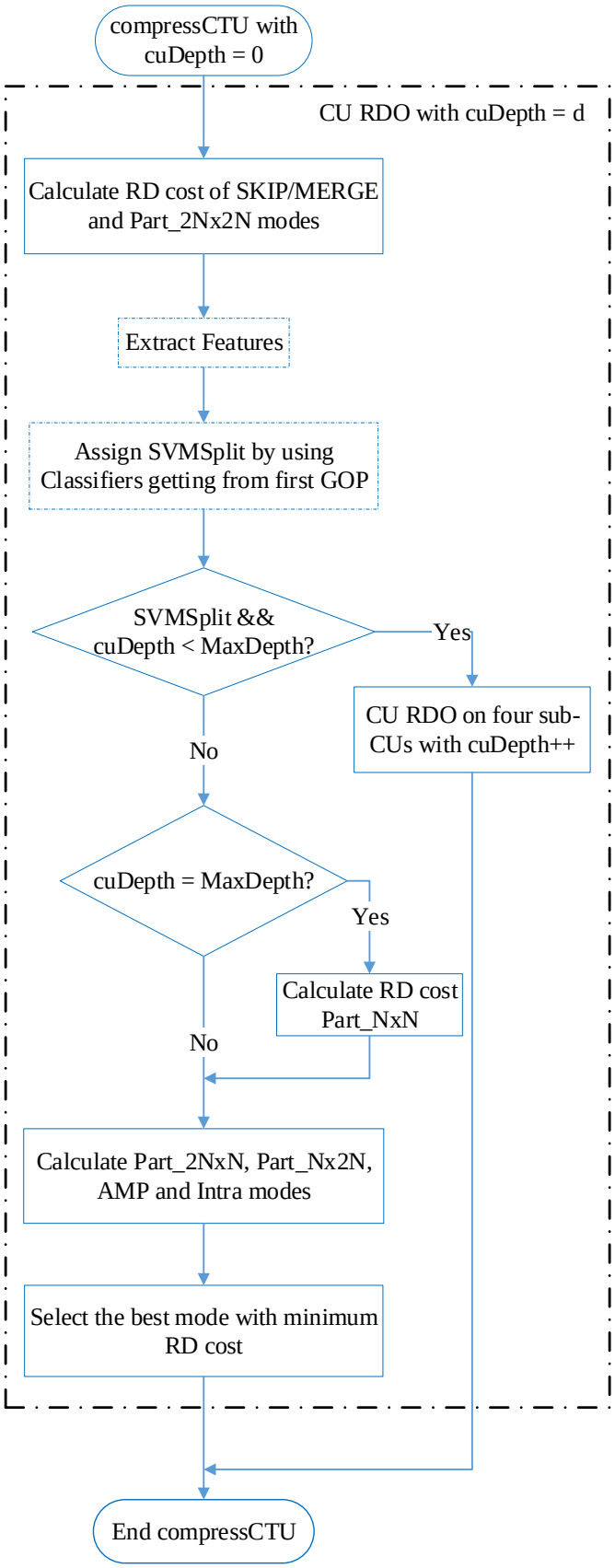



**Figure 3.9: Flowchart for quadtree-based CU partitioning of FuzzySVM[17].**

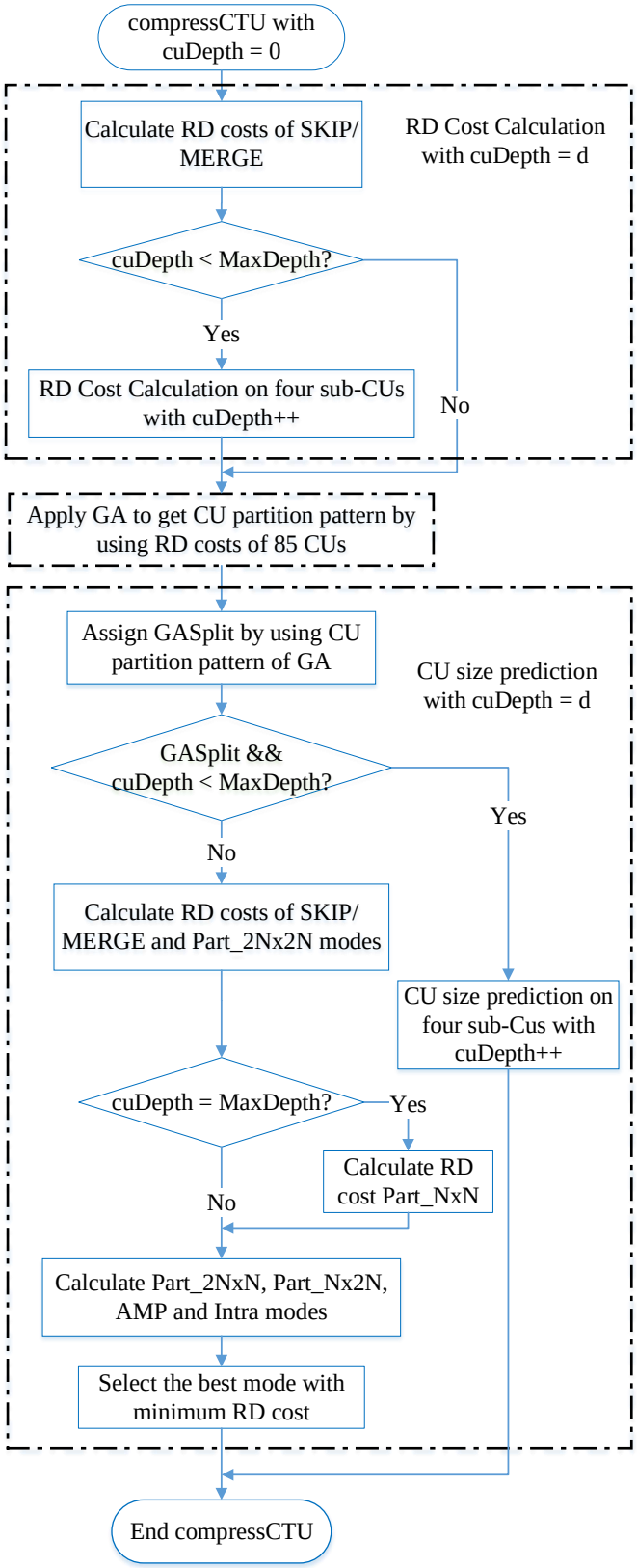



**Figure 3.10: Flowchart for quadtree-based CU partitioning of the proposed algorithm based on GA [26].**

# Chapter 4

# Evaluation

## 4.1 Test Video Sequences

As shown in Table 4.1, five sequences from three different resolutions ( Class B to Class E) are encoded with 4 QPs and Low Delay P (LDP) configuration as mentioned in [17] for the first part[25]. For second part [26], we use thirteen test video sequences of six classes ( Class A to Class F). All video sequences are from the source of common test conditions (CTC) of JCT-VC. In order to assess the coding efficiency of HD/UHD video, class A is the set of higher resolution video sequences than full HD and these video sequences are cropped to get frame resolution of 2560×1600 in order to reduce the encoding time. Class B aims to evaluate the coding efficiency of 1080p high definition television (HDTV) while class E uses for low latency video applications such as video conferencing. In order to measure the coding efficiency for mobile applications, class C and D video sequences can be used. In addition, captured video sequences, there is one different class called class F which contains video scenes which are not captured by the camera and captured by the device itself to get the screen content. These all test video sequences have different scene behaviors such as the scene with moving objects in the foreground and static background and scene with moving objects in the foreground and dynamic background in a crowded area. The video contents of class A, B, C, D, E, F, and the actual size for all classes are shown in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5, Figure 4.6, and Figure 4.7, respectively.

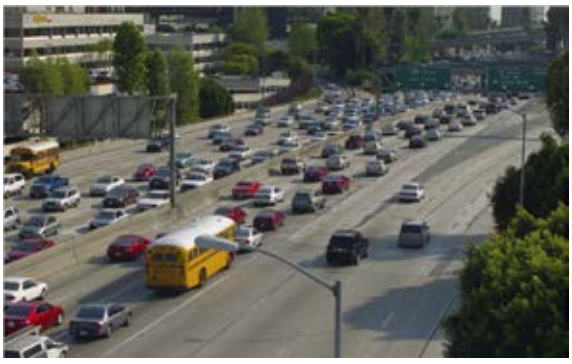


**Figure 4.1: Traffic test sequence of Class A.**

(a) (b)

(c) (d)

**Figure 4.2: Class B. (a) Kimono1. (b) ParkScene. (c) Cactus. (d) BasketballDrive.**

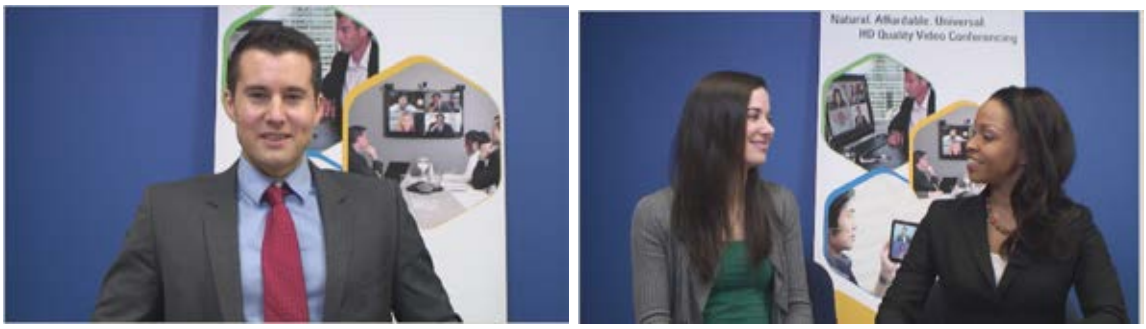

(a) (b)

(c)

**Figure 4.3: Class C. (a) BQMall. (b) PartyScene. (c) BasketballDrillText.**

(a) (b)

**Figure 4.4: Class D. (a) BQSquare. (b) BlowingBubbles.**

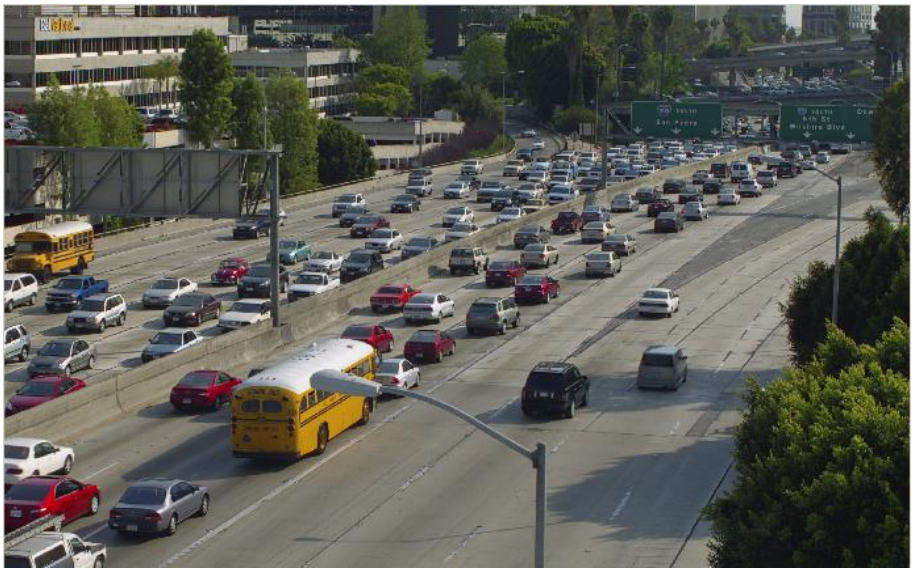



(a) (b)

(c)

**Figure 4.5: Class E. (a) BQMall. (b) PartyScene. (c) BasketballDrillText.**

**Figure 4.6: ChinaSpeed test sequence of Class F.**



Class A

Class B

Class C

Class D

Class E

Class F

**Figure 4.7: Example test sequences for each test sequence class printed in with appropriate relative size.**

**Table 4.1: Testing sequences of the JCT-VC data set.**

| Class | Resolution | Name | No. of Frames | Frame Rate |
|---|---|---|---|---|
| A | 2560×1600 | Traffic | 150 | 30 |
| B | 1920×1080 | Kimono1 | 240 | 24 |
| | | ParkScene | 240 | 24 |
| | | Cactus | 500 | 50 |
| | | BasketballDrive | 500 | 50 |
| C | 832×480 | BQMall | 600 | 60 |
| | | PartyScene | 500 | 50 |
| | | BasketballDrillText | 500 | 50 |
| D | 416×240 | BQSquare | 600 | 60 |
| | | BlowingBubbles | 500 | 50 |
| E | 1280×720 | Johnny | 600 | 60 |
| | | KristenAndSara | 600 | 60 |
| | | Vidyo4 | 600 | 60 |
| F | 1024×768 | ChinaSpeed | 500 | 30 |

## 4.2 Experimental Setup

In order to measure the performance of the proposed method, the experiments are done by using HM 16.5. All simulation are done by using the computer which are equipped with Intel Core i7-6700 CPU @ 3.40 GHz × 8 processor, 8 GB memory, and Ubuntu 16.04 LTS 64-bit Linux operating system.

For the first part [25], all experiments are carried out under low delay P (LDP) without RC (4 QPs) *i.e.*, QP = 22, 27, 32 and 37 and with enabled RC and two bit rates, *i.e.*, 256 Kbps and 512 Kbps.

For the second part [26], all experiments are carried out under low delay P (LDP) and low delay (LD) configuration with enabled RC and four bit rates, *i.e.*, 1 Mbps, 2 Mbps, 4 Mbps, and 8 Mbps.

## 4.3 Performance Metric

In this thesis, to measure the RD performance of a conventional SVM method [17] and feature reduction on that method [25] over the original HEVC test model (HM 16.5), Bjøntegard Delta Peak Signal To Noise Ratio (BDPSNR) and Bjøntegard Delta Bit Rate (BDBR) [28] are used. For both parts, we utilize the most important performance metric of fast encoding, called the computational time saving (TS). For TS based on QP, TS is

calculated as

$$TS = \frac{1}{4}\sum_{n=1}^{4}\frac{T_{HM16.5}(QP_n) - T_P(QP_n)}{T_{HM16.5}(QP_n)}, \tag{4.1}$$

, where $T_{HM16.5}(QP_n)$ is the encoding time of the original HM 16.5 and and $T_P(QP_n)$ is FuzzySVM[17] or the proposed method $P$ with $QP_n$ where $QP \in \{22, 27, 32, 37\}$ .

For TS based on RC, TS is searched as

$$TS = \frac{T_{HM16.5} - T_{FastAlgorithm}}{T_{HM16.5}} \times 100\% \tag{4.2}$$

, where $T_{HM16.5}$ is the computational time of the HM16.5 and $T_{FastAlgorithm}$ is FuzzySVM[17] or the proposed fast CU encoding. In addition, we use an another important factor for measuring the quality degradation of fast encoding, called the peak signal-to-noise ratio of luminance component (Y-PSNR) for the second part.

## 4.4 Feature Reduction on Conventional Fuzzy SVM-based Approach

The experiment results under LDP configuration without RC are shown in Figure 4.8. According to the experiment results, feature reduction on a conventional fuzzy SVM method [25] is unable to reduce the computational time. One of the reasons is that LDP configuration without RC may not reduce the computational time as we expected. But, according to the Figure 4.9, it can be found that a certain amount of computational time can be reduced by enabling RC without quality degradation.

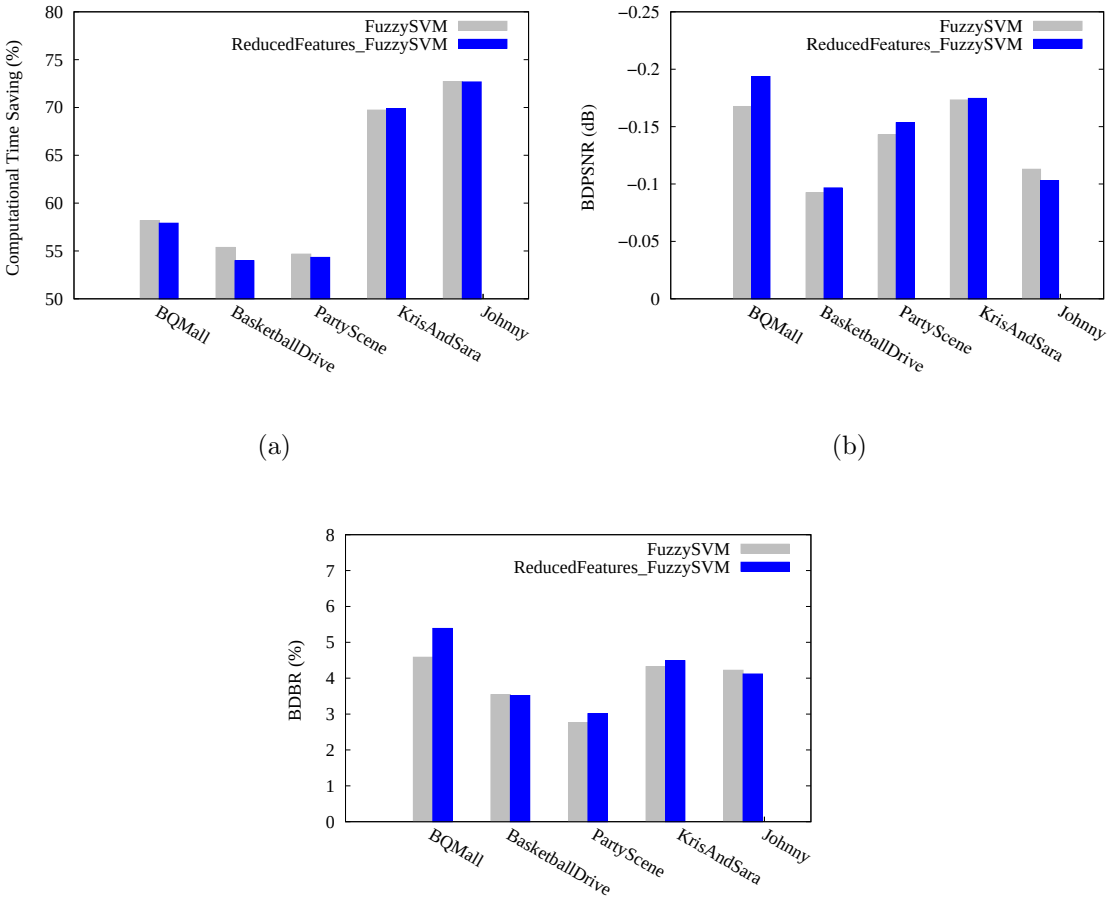



**Figure 4.8: Performance comparison with FuzzySVM [17] under LDP Configuration without RC and 4 QPs. (a) Computational time saving (%). (b) Video quality degradation BDPSNR (dB). (c) Overhead bitrate BDBR (%).**

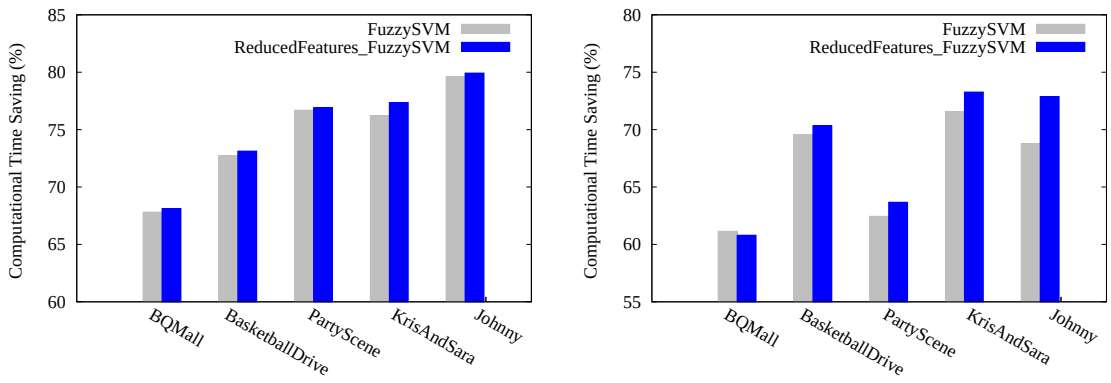



**Figure 4.9: Performance comparison with FuzzySVM [17] under LDP Configuration with RC. (a) Bitrate = 256 kbps.(b) Bitrate = 512 kbps.**

## 4.5 GA-Based Fast CU Partitioning

### 4.5.1 Performance Comparison with Original HM and State-of-the-art Approach

In this subsection, the original HM and FuzzySVM [17] are the benchmarks to compare with the proposed method [26]. FuzzySVM predicts the CU sizes for three depth levels by using three modified SVM classifiers. The source code of FuzzySVM is downloaded from the authors [17].

Table 4.2 and Table 4.3 describe the experimental results of HM16.5, FuzzySVM, and the proposed method [26] under low delay P (LDP) and low delay (LD) configurations with enabled RC, respectively. We use four target bit rates such as 1 Mbps, 2 Mbps, 4 Mbps, and 8 Mbps which are described as 1, 2, 4, and 8, respectively, in Table 4.2 and Table 4.3. The larger TS indicates that the encoding time can be further reduced. Compared with HM 16.5, the average computational time saving of FuzzySVM for all bit rates ranges from 45.8% to 61.1% and 49% to 60.8%, while our proposed method can achieve the saving for all bit rates ranges from 62.5% to 66.6% and 64.1% to 67.4% under LDP and LD configurations, respectively. The average computational time savings of our proposed method over FuzzySVM are thus 5.5%, 8.3%, 11%, and 16.7% for the bit rates of 1, 2,

4, and 8 Mbps, respectively, under LDP while 6.5%, 9.3%, 10.6%, and 15.1% under LD configuration. These trade-offs are with the negligible average PSNR loss of less than 0.5 dB. Our proposed method has a stable improvement in computational time saving and a comparable value in video quality especially at the higher bit rate cases. In special, our proposed method achieves better quality performance for the Class E test sequences. Therefore, our proposed method is more effective than FuzzySVM with comparable video quality and an improved computational time saving, especially for low latency application such as real-time video conversational application.

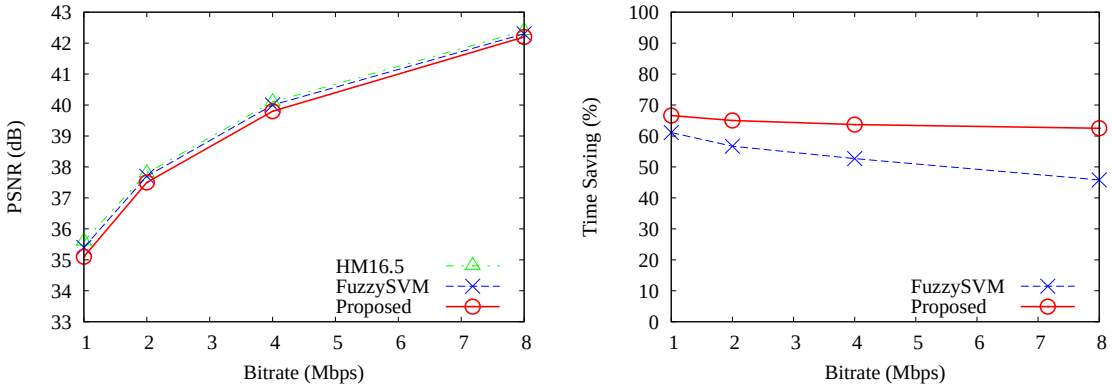
For several different sequences, the proposed algorithm can save the maximum and minimum computational complexity of 68.5% in "ChinaSpeed" and 45.4% in "BlowingBubbles" for LDP, respectively, with a comparable quality loss at 8192 Kbps compared with HM16.5. For 1024 Kbps, the proposed method can reduce the computational complexity at most 74.1% in "Traffic" and at least 47.3% in "BlowingBubbles" with a negligible quality degradation compared with HM16.5. For one example video sequence of class A called "Traffic", the proposed method greatly achieves 18.5% computational complexity reduction over FuzzySVM with a comparable PSNR value at 8192 Kbps. For "KristenAndSara" video sequence of class E, our proposed algorithm significantly reduces 18.9% computation burden more than FuzzySVM at 8192 Kbps under the same video quality. In specific, our proposed method achieves a similar RD performance as the original HM under a notable computational time saving for the test video sequences which have a highly temporal correlation between consecutive frames such as "Johnny", "KristenAndSara", "FourPeople" of class E. The behavior of class E is a scene with only people's faces and upper bodies movements in the foreground and static background. Figure 4.10 and Figure 4.11 describe the experimental results of HM16.5, FuzzySVM, and the proposed method [26] under low delay P (LDP) and low delay (LD) configurations with enabled RC, respectively.

**Table 4.2: Performance comparison with HM16.5 and start-of-the-art fast algorithm, FuzzySVM[17] (LDP).**

| Class | Sequence | Bit rate (Mbps) | Y-PSNR (dB) | | | TS (%) | |
|---|---|---|---|---|---|---|---|
| | | | HM16.5 | FuzzySVM[17] | Proposed[26] | FuzzySVM[17] | Proposed[26] |
| A | Traffic | 1 | 33.3 | 33.2 | 32.2 | 66.6 | 74.1 |
| | | 2 | 35.8 | 35.7 | 35.1 | 56.9 | 71.6 |
| | | 4 | 38.0 | 37.9 | 37.6 | 59.1 | 69.4 |
| | | 8 | 39.9 | 39.8 | 39.6 | 48.6 | 67.1 |
| B | Kimono1 | 1 | 36.2 | 36.1 | 35.7 | 61.4 | 69.4 |
| | | 2 | 38.7 | 38.6 | 38.4 | 54.8 | 68.2 |
| | | 4 | 40.8 | 40.8 | 40.6 | 46.8 | 67.2 |
| | | 8 | 42.2 | 42.1 | 42.1 | 45.8 | 65.9 |
| | ParkScene | 1 | 33.3 | 33.2 | 32.7 | 61.1 | 69.7 |
| | | 2 | 35.4 | 35.3 | 34.9 | 57.3 | 66.8 |
| | | 4 | 37.6 | 37.5 | 37.3 | 51.5 | 65.2 |
| | | 8 | 39.7 | 39.6 | 39.5 | 46.2 | 63.4 |
| | Cactus | 1 | 31.7 | 31.5 | 31.0 | 67.7 | 72.1 |
| | | 2 | 33.8 | 33.7 | 33.4 | 62.1 | 70.6 |
| | | 4 | 35.8 | 35.7 | 35.5 | 61.7 | 69.2 |
| | | 8 | 37.1 | 37.1 | 37.0 | 54.3 | 67.2 |
| C | BQMall | 1 | 34.7 | 34.5 | 34.1 | 60.4 | 65.6 |
| | | 2 | 37.3 | 37.2 | 36.9 | 56.8 | 63.9 |
| | | 4 | 39.6 | 39.5 | 39.3 | 51.0 | 62.4 |
| | | 8 | 41.5 | 41.5 | 41.4 | 47.6 | 62.8 |
| | PartyScene | 1 | 29.6 | 29.5 | 29.0 | 58.9 | 64.6 |
| | | 2 | 32.1 | 32.0 | 31.7 | 54.3 | 62.7 |
| | | 4 | 34.8 | 34.7 | 34.5 | 48.7 | 61.6 |
| | | 8 | 37.7 | 37.6 | 37.5 | 42.6 | 60.4 |
| | BasketballDrillText | 1 | 34.5 | 34.4 | 34.0 | 58.0 | 65.4 |
| | | 2 | 37.4 | 37.3 | 36.9 | 54.1 | 64.3 |
| | | 4 | 40.3 | 40.2 | 39.9 | 52.0 | 62.9 |
| | | 8 | 43.1 | 43.1 | 42.8 | 47.5 | 61.9 |
| D | BQSquare | 1 | 34.7 | 34.6 | 34.4 | 49.5 | 48.3 |
| | | 2 | 37.2 | 37.1 | 37.0 | 47.3 | 47.8 |
| | | 4 | 40.3 | 40.3 | 40.1 | 42.8 | 47.5 |
| | | 8 | 44.4 | 44.4 | 44.3 | 34.0 | 47.1 |
| | BlowingBubbles | 1 | 35.0 | 34.9 | 34.8 | 48.8 | 47.3 |
| | | 2 | 37.8 | 37.8 | 37.7 | 44.3 | 45.9 |
| | | 4 | 40.9 | 40.9 | 40.8 | 41.1 | 45.4 |
| | | 8 | 45.3 | 45.2 | 45.1 | 34.6 | 45.4 |
| E | Johnny | 1 | 42.1 | 42.0 | 41.9 | 69.6 | 72.7 |
| | | 2 | 42.9 | 42.8 | 42.8 | 65.9 | 71.1 |
| | | 4 | 43.6 | 43.6 | 43.5 | 58.6 | 69.5 |
| | | 8 | 44.5 | 44.5 | 44.4 | 50.7 | 67.6 |
| | KristenAndSara | 1 | 41.7 | 41.6 | 41.4 | 69.8 | 73.0 |
| | | 2 | 43.0 | 43.0 | 42.9 | 65.0 | 71.4 |
| | | 4 | 44.0 | 43.9 | 43.9 | 60.3 | 69.5 |
| | | 8 | 45.0 | 44.9 | 44.9 | 48.6 | 67.5 |
| | Vidyo4 | 1 | 41.1 | 40.9 | 40.8 | 62.5 | 72.6 |
| | | 2 | 42.5 | 42.4 | 42.3 | 60.0 | 71.1 |
| | | 4 | 43.7 | 43.7 | 43.6 | 57.4 | 69.5 |
| | | 8 | 45.1 | 45.0 | 45.0 | 51.7 | 67.8 |
| F | ChinaSpeed | 1 | 34.6 | 34.4 | 33.9 | 60.0 | 70.7 |
| | | 2 | 37.9 | 37.7 | 37.4 | 58.2 | 69.9 |
| | | 4 | 41.5 | 41.3 | 41.0 | 54.1 | 69.3 |
| | | 8 | 45.4 | 45.2 | 44.9 | 43.6 | 68.5 |
| **All** | **AVERAGE** | **1** | **35.6** | **35.4** | **35.1** | **61.1** | **66.6** |
| | | **2** | **37.8** | **37.7** | **37.5** | **56.7** | **65.0** |
| | | **4** | **40.1** | **40.0** | **39.8** | **52.7** | **63.7** |
| | | **8** | **42.4** | **42.3** | **42.2** | **45.8** | **62.5** |

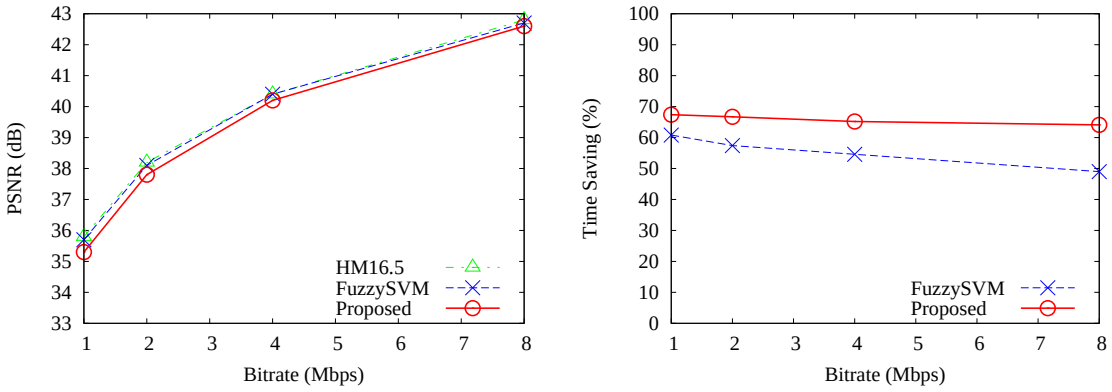
**Table 4.3: Performance comparison with HM16.5 and start-of-the-art fast algorithm, FuzzySVM[17] (LD).**

| Class | Sequence | Bit rate (Mbps) | Y-PSNR (dB) | | | TS (%) | |
|---|---|---|---|---|---|---|---|
| | | | HM16.5 | FuzzySVM[17] | Proposed[26] | FuzzySVM[17] | Proposed[26] |
| A | Traffic | 1 | 33.4 | 33.2 | 32.4 | 65.7 | 74.9 |
| | | 2 | 35.9 | 35.8 | 35.2 | 59.3 | 74.1 |
| | | 4 | 38.2 | 38.1 | 37.8 | 60.6 | 71.9 |
| | | 8 | 40.2 | 40.2 | 39.9 | 49.0 | 70.7 |
| B | Kimono1 | 1 | 36.5 | 36.4 | 36.0 | 61.4 | 69.9 |
| | | 2 | 38.9 | 38.9 | 38.7 | 54.8 | 68.8 |
| | | 4 | 41.0 | 41.0 | 40.9 | 49.4 | 67.9 |
| | | 8 | 42.4 | 42.4 | 42.3 | 40.5 | 66.9 |
| | ParkScene | 1 | 33.3 | 33.2 | 32.8 | 58.2 | 70.0 |
| | | 2 | 35.5 | 35.4 | 35.0 | 59.4 | 68.7 |
| | | 4 | 37.7 | 37.7 | 37.4 | 52.8 | 67.5 |
| | | 8 | 39.9 | 39.9 | 39.7 | 39.6 | 66.4 |
| | Cactus | 1 | 31.8 | 31.7 | 31.2 | 65.9 | 70.0 |
| | | 2 | 34.0 | 33.9 | 33.5 | 61.2 | 68.7 |
| | | 4 | 35.9 | 35.9 | 35.6 | 62.2 | 67.5 |
| | | 8 | 37.4 | 37.3 | 37.2 | 56.7 | 66.4 |
| C | BQMall | 1 | 34.9 | 34.8 | 34.4 | 60.5 | 67.1 |
| | | 2 | 37.6 | 37.5 | 37.2 | 58.0 | 65.8 |
| | | 4 | 40.0 | 39.9 | 39.7 | 54.3 | 64.7 |
| | | 8 | 41.9 | 41.8 | 41.8 | 49.9 | 63.5 |
| | PartyScene | 1 | 29.7 | 29.6 | 29.2 | 59.6 | 66.4 |
| | | 2 | 32.4 | 32.3 | 32.0 | 57.9 | 65.1 |
| | | 4 | 35.3 | 35.3 | 35.0 | 52.2 | 64.1 |
| | | 8 | 38.5 | 38.4 | 38.2 | 47.4 | 63.1 |
| | BasketballDrillText | 1 | 34.8 | 34.7 | 34.3 | 60.2 | 67.1 |
| | | 2 | 37.8 | 37.7 | 37.3 | 55.2 | 65.8 |
| | | 4 | 40.8 | 40.7 | 40.4 | 55.0 | 64.8 |
| | | 8 | 43.7 | 43.6 | 43.4 | 53.4 | 64.2 |
| D | BQSquare | 1 | 35.5 | 35.4 | 35.2 | 50.2 | 50.5 |
| | | 2 | 38.1 | 38.0 | 37.9 | 48.3 | 54.8 |
| | | 4 | 41.2 | 41.2 | 41.1 | 45.1 | 49.3 |
| | | 8 | 45.4 | 45.3 | 45.2 | 42.5 | 48.4 |
| | BlowingBubbles | 1 | 35.3 | 35.2 | 35.1 | 50.4 | 49.7 |
| | | 2 | 38.4 | 38.3 | 38.2 | 49.2 | 48.8 |
| | | 4 | 41.7 | 41.6 | 41.5 | 44.0 | 47.7 |
| | | 8 | 46.0 | 46.0 | 45.9 | 42.2 | 47.6 |
| E | Johnny | 1 | 42.4 | 42.3 | 42.2 | 69.9 | 72.8 |
| | | 2 | 43.3 | 43.2 | 43.1 | 66.1 | 71.9 |
| | | 4 | 44.0 | 44.0 | 43.9 | 62.2 | 70.5 |
| | | 8 | 44.8 | 44.8 | 44.7 | 57.0 | 68.7 |
| | KristenAndSara | 1 | 41.9 | 41.7 | 41.6 | 68.9 | 72.9 |
| | | 2 | 43.3 | 43.2 | 43.1 | 65.1 | 71.7 |
| | | 4 | 44.2 | 44.2 | 44.1 | 60.7 | 70.4 |
| | | 8 | 45.2 | 45.2 | 45.2 | 53.2 | 68.7 |
| | Vidyo4 | 1 | 41.4 | 41.3 | 41.1 | 59.6 | 72.8 |
| | | 2 | 42.9 | 42.8 | 42.7 | 53.4 | 71.6 |
| | | 4 | 44.1 | 44.0 | 43.9 | 56.1 | 70.5 |
| | | 8 | 45.5 | 45.4 | 45.3 | 57.3 | 69.4 |
| F | ChinaSpeed | 1 | 34.7 | 34.5 | 34.0 | 60.2 | 71.7 |
| | | 2 | 38.1 | 37.9 | 37.5 | 58.6 | 71.2 |
| | | 4 | 41.6 | 41.4 | 41.1 | 55.1 | 70.6 |
| | | 8 | 45.5 | 45.4 | 45.1 | 48.1 | 69.9 |
| **All** | **AVERAGE** | **1** | **35.8** | **35.7** | **35.3** | **60.8** | **67.4** |
| | | **2** | **38.2** | **38.1** | **37.8** | **57.4** | **66.7** |
| | | **4** | **40.4** | **40.4** | **40.2** | **54.6** | **65.2** |
| | | **8** | **42.8** | **42.7** | **42.6** | **49.0** | **64.1** |




(a) (b)

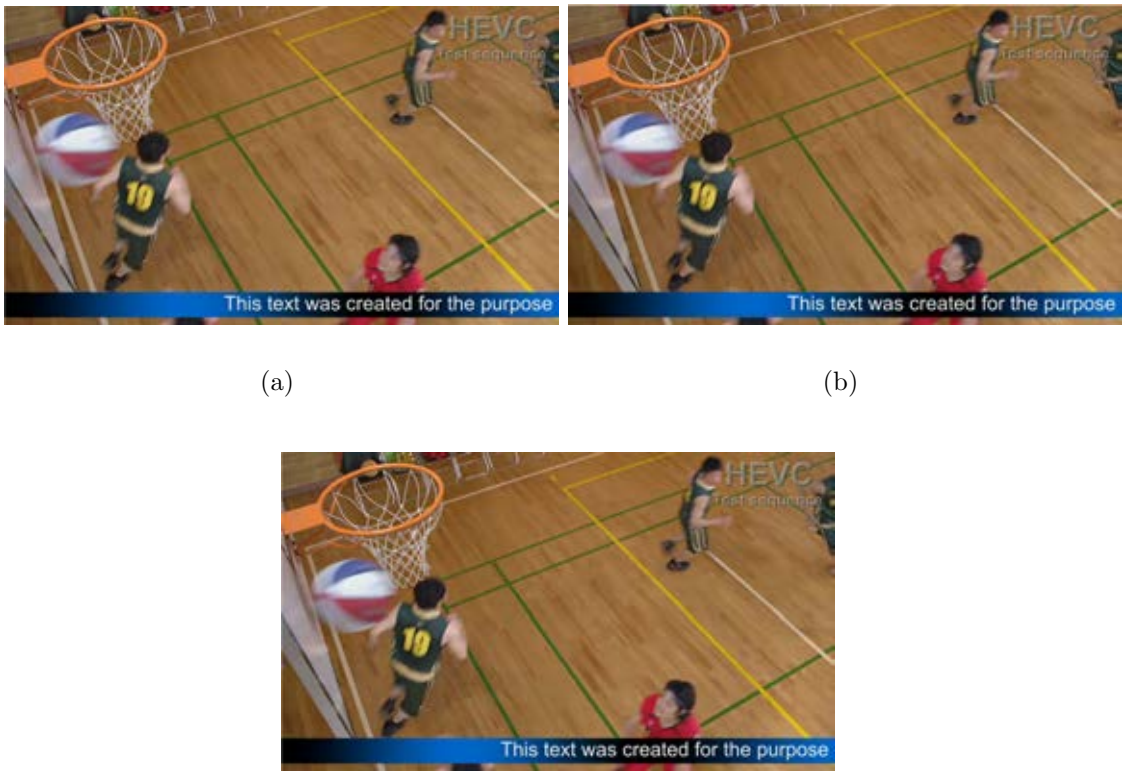
**Figure 4.10: Average performance comparison with FuzzySVM [17] under LDP configuration. (a) Video quality (dB). (b) Computational time saving (%).**




(a) (b)

**Figure 4.11: Average performance comparison with FuzzySVM [17] under LD configuration. (a) Video quality (dB). (b) Computational time saving (%).**

For video quality comparison between original HM16.5, FuzzySVM and the proposed method, Figure 4.12 shows the reconstructed frames decoded by HM16.5, FuzzySVM and the proposed method at 1024 kbps under LDP configuration. The required bit allocations and PSNR values of that frame are 14976 and 34.6 dB for HM16.5, 13936 and 34.4 dB for FuzzySVM, and 14280 and 34.1 dB for the proposed method. Even though the PSNR degradation of the proposed method compared with HM16.5 and FuzzySVM are nearly 1 dB and 0.2 dB, respectively, at 1024 kbps, a similar video quality is perceived by the human




(a) (b)

(c)

**Figure 4.12: Subjective video quality for POC 242 of BasketballDrillText video sequence.(a) Encoded frame of HM16.5. (b) Encoded frame of FuzzySVM. (c) Encoded frame of the proposed method [26].**

visual system.

### 4.5.2 Stability of the Proposed Method

In order to analyze the stability of the proposed method, we use several target bit rates such as 1, 2, 4, 8, 16, and 32 Mbps to encode two different test video sequences such as "BQMall" from class C and "Johnny" from class E under LDP configuration. "BQMall" sequence has many moving objects in foreground and camera movements while "Johnny" has the only face and upper body movements with a stable background. Figure 4.13 shows the stability of the proposed method compared with FuzzySVM. From this figure, it can be observed that our proposed method gives a stable reduction in the computational burden of HEVC from the 1 Mbps to 32 Mbps target bit rate. This happens due to the consideration of temporal correlation and the advantage of effective chromosome structure, fitness

function, and negligible computation time of GA which is not greater than 1% of the total encoding time as shown in Figure 4.14. On the other hand, the computational time saving of FuzzySVM is not stable and is dramatically reducing since FuzzySVM has a higher chance to make a splitting decision when the target bit rate is higher. In details, Figure 4.15 and Figure 4.16 describe the percentage of SVM decision of each depth for "BQMall" and "Johnny" test sequences, respectively. From Figure 4.17, it can be found that the total splitting decision percentage for all three depths is increasing while the target bit rate is increasing. As shown in Figure 4.18, FuzzySVM calculates the RD costs for only one 64×64 CU when the target bit rate is 1 Mbps. For 4 Mbps, there are one 64×64 CU and four 32×32 CUs. The RD costs for one 64×64 CU, four 32×32 CUs, eight 16×16 CUs, and four 8×8 CUs are calculated at the high target bit rate, 8 Mbps. Therefore, the total number of CUs that need to be calculated the RD cost may be increased when the target bit rate is increasing. As a result, FuzzySVM may consume a larger computational complexity while HM16.5 and the proposed method calculate the RD costs for a fixed amount of CUs.

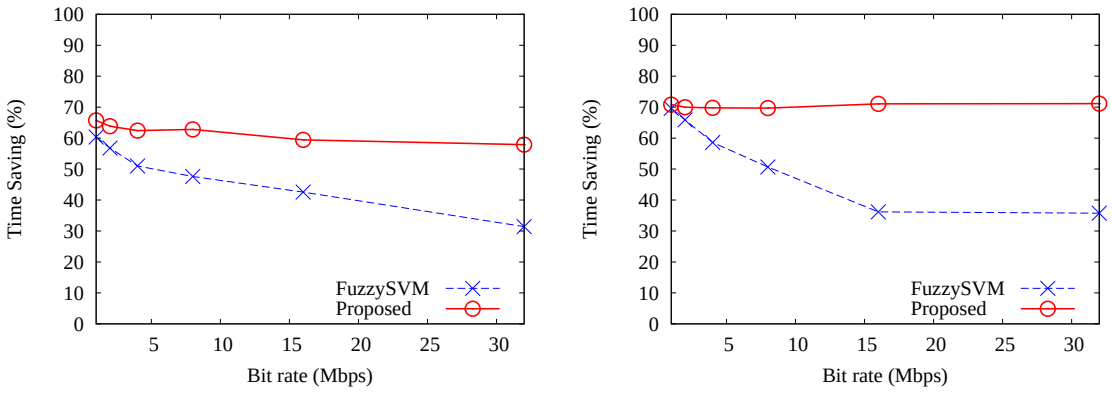



**Figure 4.13: Stabilization of proposed method[26] over FuzzySVM[17]. (a) Time saving of BQMall. (b) Time saving of Johnny.**

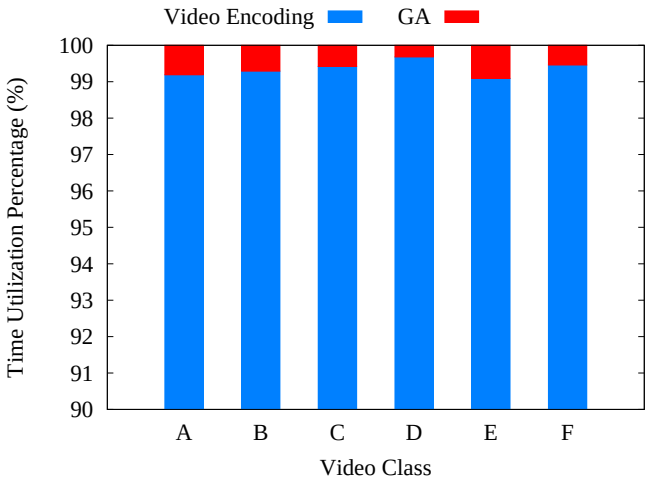



**Figure 4.14: Time utilization percentage of the proposed method [26].**

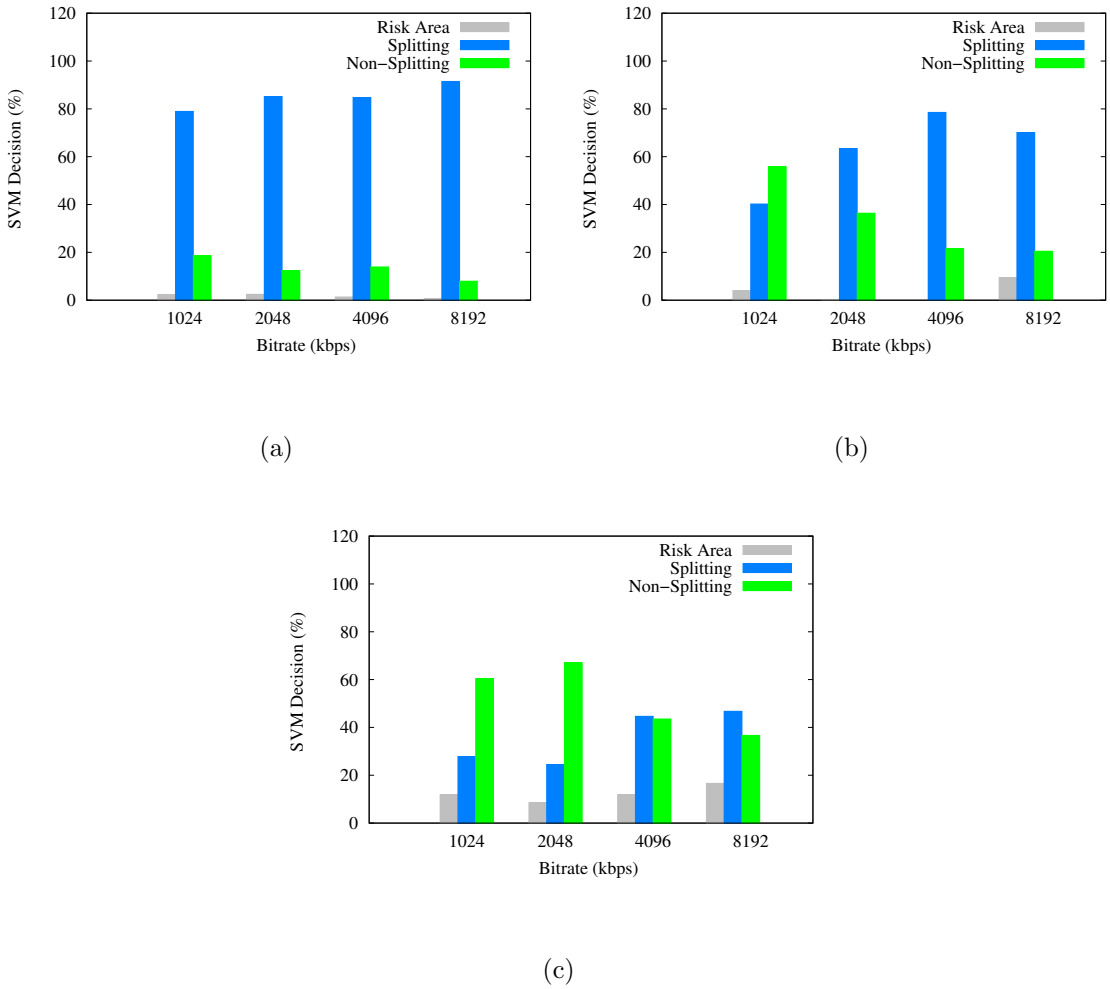



**Figure 4.15: SVM decision percentage of FuzzySVM[17] for BQMall (a) Depth 0 (b) Depth 1 (c) Depth 2.**

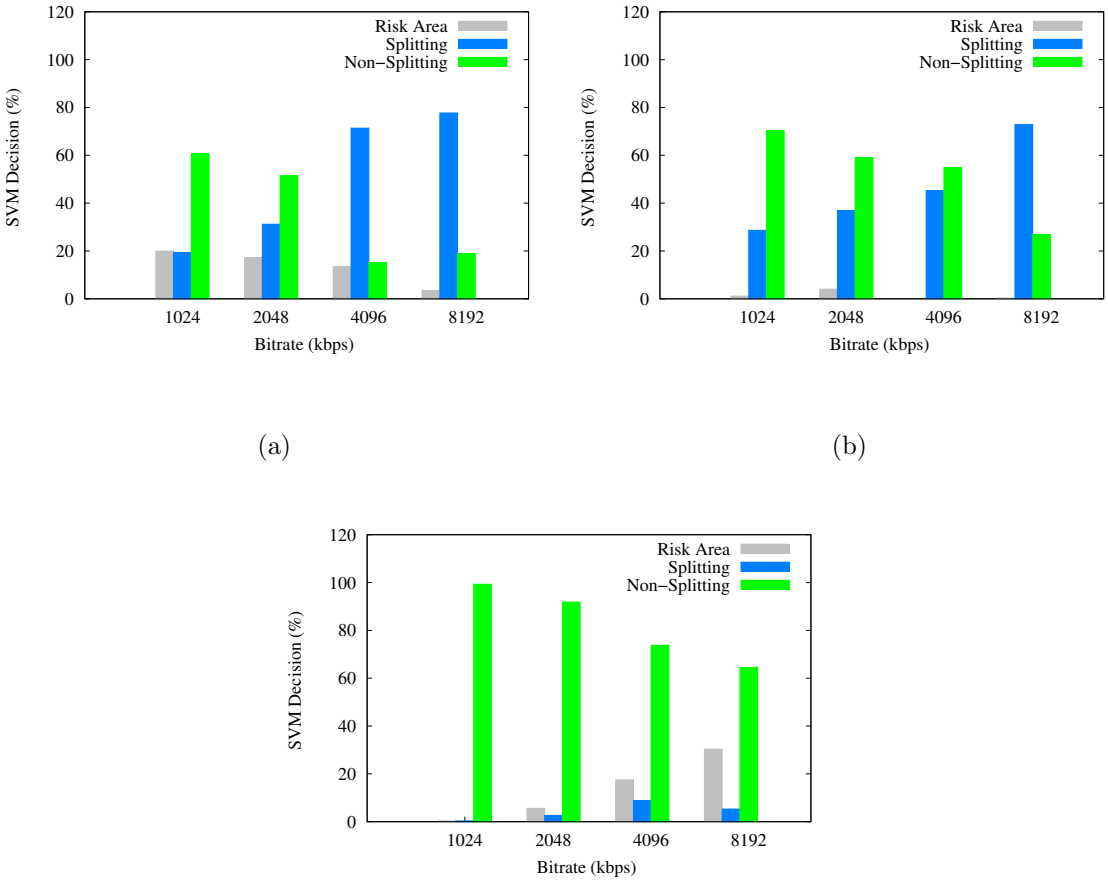



**Figure 4.16: SVM decision percentage of FuzzySVM[17] for Johnny (a) Depth 0 (b) Depth 1 (c) Depth 2.**

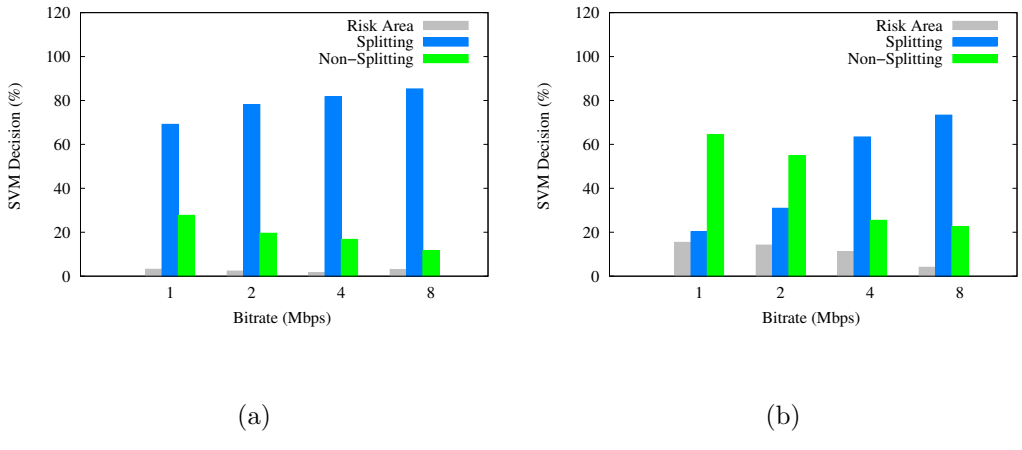



**Figure 4.17: Total SVM decision percentage of FuzzySVM[17] for depth 0, 1 and 2 (a) BQMall (b) Johnny.**

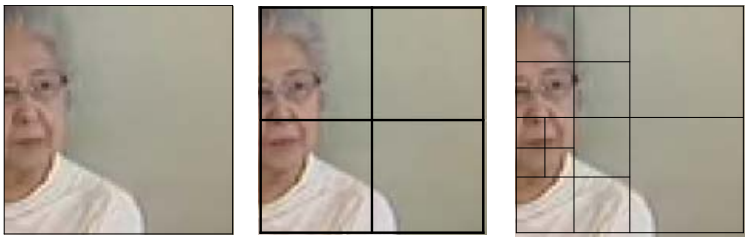


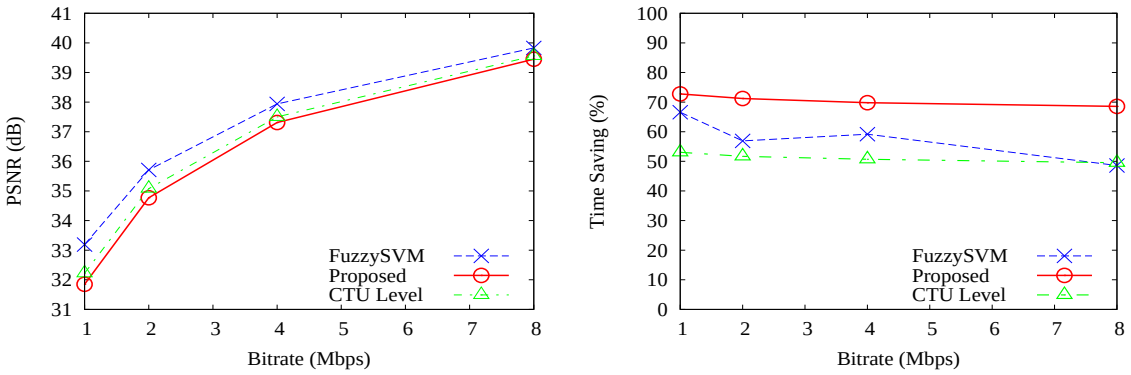
**Figure 4.18: CU Partition of a CTU encoded by FuzzySVM[17] at 1 Mbps, 4 Mbps, and 8 Mbps.**

## 4.6 Discussion

### 4.6.1 Performance Comparison with CTU Level Sharing

To compare with our frame level sharing scheme [26], another sharing scheme such as coding tree unit (CTU) level sharing have been implemented. Firstly, the motion information of each CTU is analyzed by utilizing the frame differencing method. Secondly, the CTU is categorized into two groups: moving region and non-moving region. Thirdly, the PPs from GA and key frame are utilized for CTUs which are in the moving region and non-moving region, respectively.

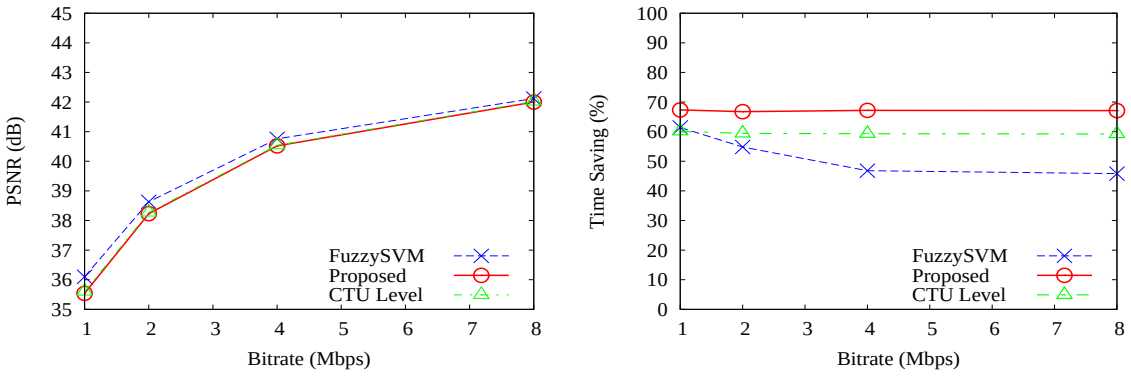
As shown in Figure 4.19, CTU level PP sharing can keep 0.37 dB, 0.31 dB, 0.2 dB, and 0.11 dB more than the proposed frame level at 1, 2, 4, and 8 Mbps, respectively, for class A video sequence called "Traffic" which is the higher resolution video sequence than full HD and is cropped to get frame resolution of 2560×1600 in order to reduce the encoding time. However, the proposed frame level can significantly reduce 19.7%, 19.5%, 19.1%, and 19% computational complexity more than CTU level at 1, 2, 4, and 8 Mbps, respectively. Especially at high bit rate (8Mbps), the proposed method achieves a large time saving with a comparable video quality compared with CTU level approach.




(a) (b)

**Figure 4.19: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Traffic (Class A). (b) Time saving of Traffic (Class A).**

For a "Kimono1" video sequence of class B with only a slow walking movement of a Japanese girl in the foreground and dynamic background, the proposed approach can save 7.2%, 7.3%, 7.9%, and 7.9% computational time saving more than CTU level with the small values of quality drop as 0.06 dB, 0.04 dB, 0.02 dB, and 0.01 dB at 1, 2, 4, and 8 Mbps, respectively, as shown in Figure 4.20.

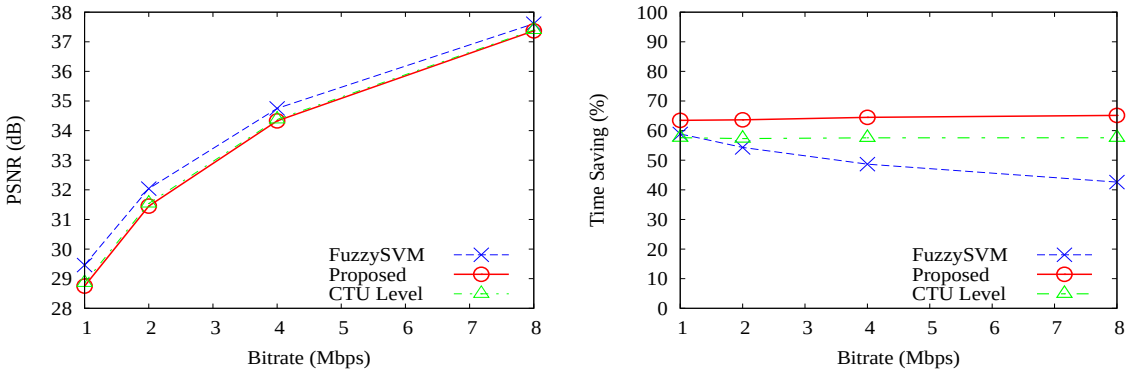



(a) (b)

**Figure 4.20: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Kimono1 (Class B). (b) Time saving of Kimono1 (Class B).**

As shown in Figure 4.21, CTU level PP sharing can keep 0.11 dB, 0.09 dB, 0.05 dB,

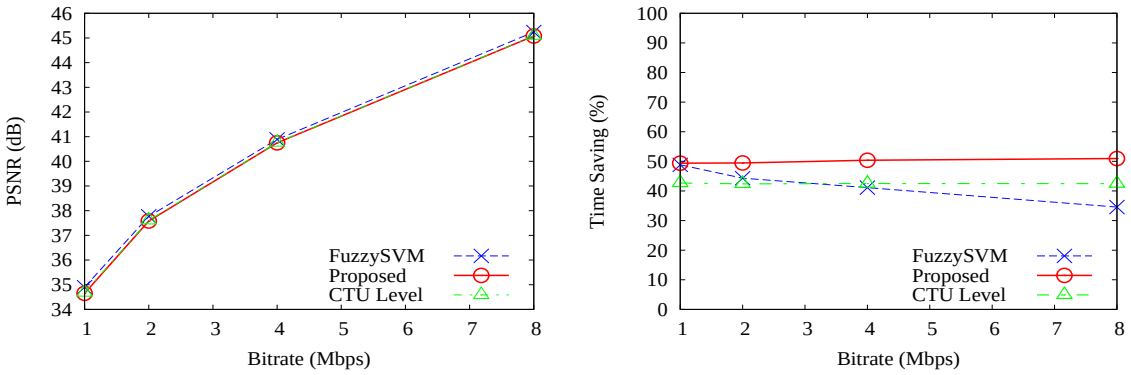
and 0.03 dB more than the proposed frame level at 1, 2, 4, and 8 Mbps, respectively, for "PartyScene" video sequence from class C which has fast moving objects in the foreground and static background. However, the proposed frame level can reduce 5.9%, 6.3%, 6.9%, and 7.5% computational complexity more than CTU level at 1, 2, 4, and 8 Mbps, respectively.




(a) (b)

**Figure 4.21: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of PartyScene (Class C). (b) Time saving of PartyScene (Class C).**

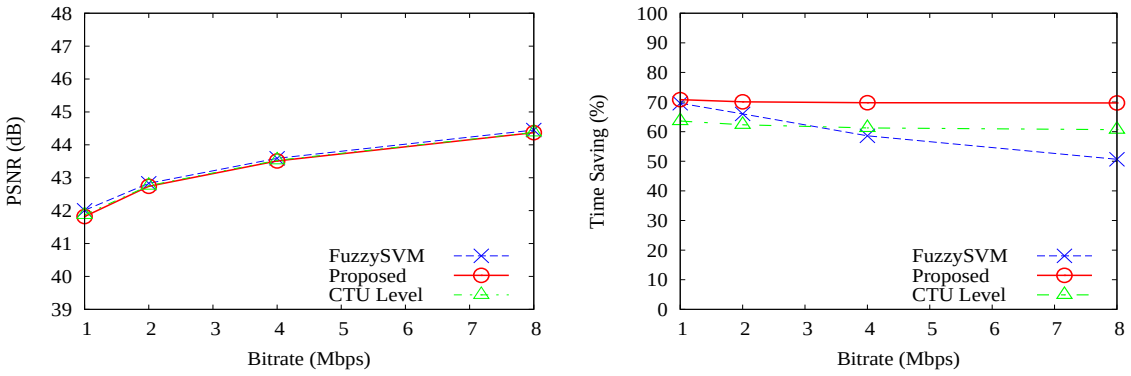
For low resolution video sequence called "BlowingBubbles", the proposed frame level can save more 6.7%, 7.1%, 7.9%, and 8.5% than CTU level while CTU level scheme can only keep 0.03 dB, 0.03 dB, 0.01 dB, and 0.01 dB more than the proposed at 1, 2, 4, and 8 Mbps, respectively, as shown in Figure 4.22.




(a) (b)

**Figure 4.22: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of BlowingBubbles (Class D). (b) Time saving of BlowingBubbles (Class D).**

For a scene with only people's faces and upper bodies movements in the foreground and static background such as "Johnny" from Class E, the proposed approach can save 7.2%, 7.8%, 8.5%, and 9% computational time saving more than CTU level with the small values of quality drop as 0.05 dB, 0.02 dB, 0.01 dB, and 0.01 dB at 1, 2, 4, and 8 Mbps, respectively, as shown in Figure 4.23.

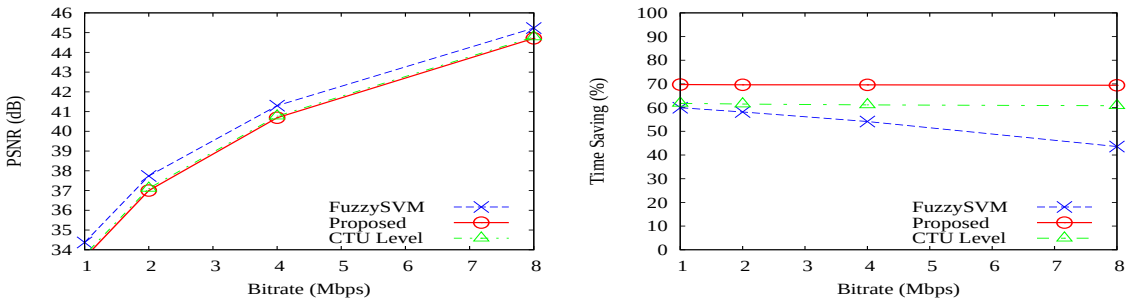



(a) (b)

**Figure 4.23: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of Johnny (Class E). (b) Time saving of Johnny (Class E).**

In addition captured video sequences, there is one different class called class F which

contains video scenes which are not captured by camera and captured by device itself to get the screen content such as "ChinaSpeed". For this video, CTU level PP sharing can maintain a small PSNR such as 0.14 dB, 0.12 dB, 0.09 dB, and 0.07 dB more than the proposed frame level under a large amount of time saving drop such as 8%, 8.1%, 8.4%, and 8.6% at 1, 2, 4, and 8 Mbps, respectively, as shown in Figure 4.24. Due to these experimental results for all classes (A to F) under LDP configuration, we effectively utilized frame level sharing scheme to significantly reduce time saving under a negligible PSNR drop.




(a) (b)

**Figure 4.24: Performance comparison between FuzzySVM [17], CTU level and proposed frame level partitioning pattern sharing scheme [26]. (a) Video quality of ChinaSpeed (Class F). (b) Time saving of ChinaSpeed (Class F).**

### 4.6.2 RD Cost Calculation with Two Most Common Modes and CU Prediction without Two Most common Modes

At the previous version of our proposed method, we have calculated the approximate RD cost after encoding with two most common modes called SKIP/MERGE and Part_2N×2N modes in RD cost calculation part. Actually, SKIP/MERGE mode is enough to calculate the approximate RD according to the empirical results. Additionally, we have not used these two common modes after predicting the CU size to save the computation time of quadtree partitioning. However, there is a small loss in RD at low bit rate. Therfore, we revised the previous version by using only the most common mode and adding two most common modes after CU size prediction to get a better PSNR value. The flowcharts of our previous version and current version of the proposed method [26] are shown in Figure

4.25. Table 4.4 shows the performance comparison of the original HM, FuzzySVM [17], the previous version and the proposed method [26].

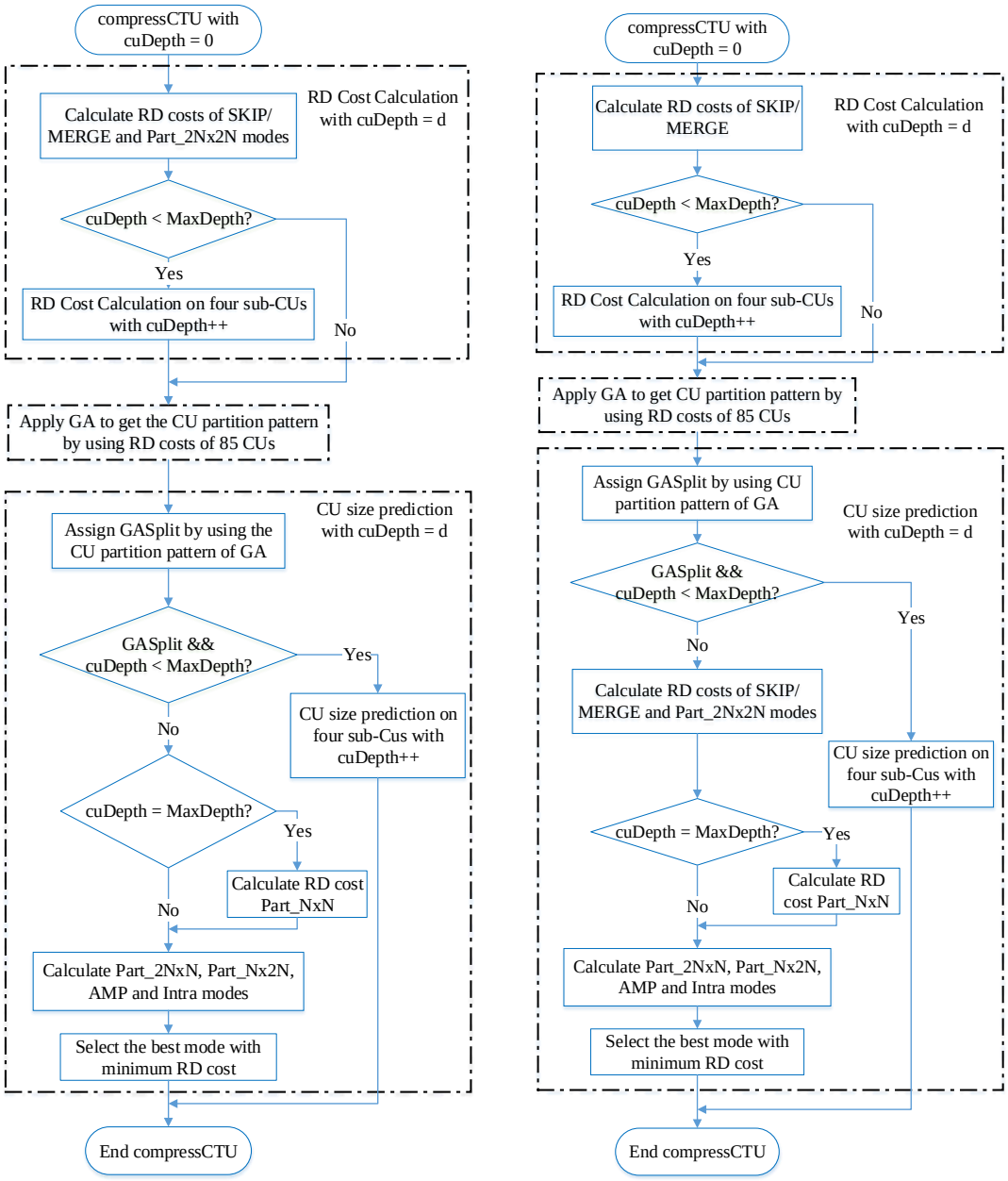



**Figure 4.25: Flowcharts of previous and proposed GA-based fast CU partitioning.**

**Table 4.4: Performance comparison of HM16.5, start-of-the-art fast algorithm [17], previous and proposed method [26]**

| Class | Sequence | Bit rate (Mbps) | Y-PSNR (dB) | | | | TS (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | HM16.5 | FuzzySVM[17] | Previous | Proposed[26] | FuzzySVM[17] | Previous | Proposed[26] |
| B | Kimono1 | 1 | 36.2 | 36.1 | 35.5 | 35.7 | 61.4 | 67.3 | 69.4 |
| | | 2 | 38.7 | 38.6 | 38.2 | 38.4 | 54.8 | 66.8 | 68.2 |
| | | 4 | 40.8 | 40.8 | 40.5 | 40.6 | 46.8 | 68.2 | 67.2 |
| | | 8 | 42.2 | 42.1 | 42.0 | 42.1 | 45.8 | 67.1 | 65.9 |
| | ParkScene | 1 | 33.3 | 33.2 | 32.4 | 32.7 | 61.1 | 67.1 | 69.7 |
| | | 2 | 35.4 | 35.3 | 34.7 | 34.9 | 57.3 | 64.2 | 66.8 |
| | | 4 | 37.6 | 37.5 | 37.1 | 37.3 | 51.5 | 64.6 | 65.2 |
| | | 8 | 39.7 | 39.6 | 39.3 | 39.5 | 46.2 | 65.1 | 63.4 |
| | Cactus | 1 | 31.7 | 31.5 | 30.9 | 31.0 | 67.7 | 69.8 | 72.1 |
| | | 2 | 33.8 | 33.7 | 33.2 | 33.4 | 62.1 | 68.7 | 70.6 |
| | | 4 | 35.8 | 35.7 | 35.3 | 35.5 | 61.7 | 68.0 | 69.2 |
| | | 8 | 37.1 | 37.1 | 36.9 | 37.0 | 54.3 | 67.1 | 67.2 |
| C | BQMall | 1 | 34.7 | 34.5 | 33.8 | 34.1 | 60.4 | 64.0 | 65.6 |
| | | 2 | 37.3 | 37.2 | 36.7 | 36.9 | 56.8 | 63.8 | 63.9 |
| | | 4 | 39.6 | 39.5 | 39.2 | 39.3 | 51.0 | 64.0 | 62.4 |
| | | 8 | 41.5 | 41.5 | 41.3 | 41.4 | 47.6 | 66.1 | 62.8 |
| | PartyScene | 1 | 29.6 | 29.5 | 28.8 | 29.0 | 58.9 | 63.5 | 64.6 |
| | | 2 | 32.1 | 32.0 | 31.4 | 31.7 | 54.3 | 63.6 | 62.7 |
| | | 4 | 34.8 | 34.7 | 34.3 | 34.5 | 48.7 | 64.5 | 61.6 |
| | | 8 | 37.7 | 37.6 | 37.4 | 37.5 | 42.6 | 65.1 | 60.4 |
| | BasketballDrillText | 1 | 34.5 | 34.4 | 33.7 | 34.0 | 58.0 | 65.5 | 65.4 |
| | | 2 | 37.4 | 37.3 | 36.7 | 36.9 | 54.1 | 64.7 | 64.3 |
| | | 4 | 40.3 | 40.2 | 39.7 | 39.9 | 52.0 | 64.7 | 62.9 |
| | | 8 | 43.1 | 43.1 | 42.7 | 42.8 | 47.5 | 65.2 | 61.9 |
| D | BQSquare | 1 | 34.7 | 34.6 | 34.3 | 34.4 | 49.5 | 50.7 | 48.3 |
| | | 2 | 37.2 | 37.1 | 36.9 | 37.0 | 47.3 | 51.3 | 47.8 |
| | | 4 | 40.3 | 40.3 | 40.1 | 40.1 | 42.8 | 51.7 | 47.5 |
| | | 8 | 44.4 | 44.4 | 44.3 | 44.3 | 34.0 | 52.5 | 47.1 |
| | BlowingBubbles | 1 | 35.0 | 34.9 | 34.7 | 34.8 | 48.8 | 49.4 | 47.3 |
| | | 2 | 37.8 | 37.8 | 37.6 | 37.7 | 44.3 | 49.5 | 45.9 |
| | | 4 | 40.9 | 40.9 | 40.7 | 40.8 | 41.1 | 50.4 | 45.4 |
| | | 8 | 45.3 | 45.2 | 45.1 | 45.1 | 34.6 | 51.0 | 45.4 |
| E | Johnny | 1 | 42.1 | 42.0 | 41.8 | 41.9 | 69.6 | 70.8 | 72.7 |
| | | 2 | 42.9 | 42.8 | 42.7 | 42.8 | 65.9 | 70.0 | 71.1 |
| | | 4 | 43.6 | 43.6 | 43.5 | 43.5 | 58.6 | 69.8 | 69.5 |
| | | 8 | 44.5 | 44.5 | 44.4 | 44.4 | 50.7 | 69.7 | 67.6 |
| | KristenAndSara | 1 | 41.7 | 41.6 | 41.3 | 41.4 | 69.8 | 71.0 | 73.0 |
| | | 2 | 43.0 | 43.0 | 42.9 | 42.9 | 65.0 | 70.2 | 71.4 |
| | | 4 | 44.0 | 43.9 | 43.9 | 43.9 | 60.3 | 69.2 | 69.5 |
| | | 8 | 45.0 | 44.9 | 44.9 | 44.9 | 48.6 | 69.1 | 67.5 |
| | Vidyo4 | 1 | 41.1 | 40.9 | 40.7 | 40.8 | 62.5 | 70.5 | 72.6 |
| | | 2 | 42.5 | 42.4 | 42.3 | 42.3 | 60.0 | 69.5 | 71.1 |
| | | 4 | 43.7 | 43.7 | 43.5 | 43.6 | 57.4 | 68.9 | 69.5 |
| | | 8 | 45.1 | 45.0 | 45.0 | 45.0 | 51.7 | 68.6 | 67.8 |
| **All** | **AVERAGE** | **1** | **35.9** | **35.7** | **35.3** | **35.4** | **60.7** | **64.5** | **65.5** |
| | | **2** | **38.0** | **37.9** | **37.6** | **37.7** | **56.5** | **63.8** | **64.0** |
| | | **4** | **40.1** | **40.1** | **39.8** | **39.9** | **52.0** | **63.9** | **62.7** |
| | | **8** | **42.3** | **42.3** | **42.1** | **42.2** | **45.8** | **64.2** | **61.5** |

### 4.6.3 GA-Based Fast CU Partitioning without Utilizing Temporal Correlation and with QP

In order to know the advantages of temporal correlation, we proposes a fast CU depth estimation algorithm based on genetic algorithm (GA)without utilizing temporal correlation [27]. In [27], we focus the main important metric of fast encoding, TS (4.1), where $T_{HM16.5}(QP_n)$ is the encoding time of the original HM 16.5 and $T_P(QP_n)$ is FuzzySVM[17] or GA-based Fast CU partitioning without temporal correlation method $P$ [27] under Low Delay P (LDP) configuration with four common QPs, *i.e.*, 22, 27, 32 and 37. As shown in Table 4.5, the experimental results show 69.2% computational time on average can be reduced by the proposed method compared with HM16.5. Compared with start-of-the-art fast encoding method, the proposed one can achieve 5.2% time saving on average under a comparable BD-PSNR.

**Table 4.5: Performance Analysis of the proposed method [27] with HM16.5 and start-of-the-art fast algorithm, FuzzySVM [17].**

| **Sequence** | **FuzzySVM [17]** | | | **Without temporal correlation [27]** | | |
|---|---|---|---|---|---|---|
| | ***BD-PSNR*** | ***BD-BR*** | ***TS*** | ***BD-PSNR*** | ***BD-BR*** | ***TS*** |
| BQMall | -0.126 | 3.340 | 58.5 | -0.412 | 11.235 | 69.42 |
| PartyScene | -0.094 | 2.408 | 54.64 | -0.405 | 10.664 | 60.47 |
| Johnny | -0.090 | 4.227 | 72.76 | -0.095 | 10.045 | 74.12 |
| KrisAndSara | -0.106 | 3.670 | 69.9 | -0.379 | 14.465 | 72.66 |
| **Average** | **-0.104** | **3.411** | **63.9** | **-0.323** | **11.602** | **69.2** |

The units of BD-PSNR, BD-BR and TS are in dB, % and %, respectively.

# Chapter 5

# Conclusion and Future Works

HEVC is the newest video codec of the Joint Collaborative Team on Video Coding (JCT-VC) which can save a 50% bit rate of H.264 under the same video quality because of its advanced features such as a quadtree-based CU partition, a modified deblocking filter, and 35 prediction modes for intra coding. However, these advanced features make the computational complexity of HEVC to be extremely high. Among these features, quadtree-based CU partition is the most expensive computational cost (over 80% in the HEVC test model) due to a recursively exhaustive RDO search. Therefore, most of the fast algorithms have focused on the CU partition by utilizing a statistical approach or learning approach in order to save the encoding time of HEVC.

In this thesis, we propose a feature reduction approach on a fuzzy SVM method to reduce the time consumed by some correlated features of the optimal feature sets. The experimental results confirm that a feature reduction on a conventional method can save the computational time with the same RD performance under LDP configuration with RC. As we know, machine learning prediction can achieve 50% reduction for complexity and that is a good approach to investigate more. After reduction some correlated features from the best feature set, it subject that it is a good tendency to decrease the computational complexity more than before. It may be possible to apply the feature reduction approach to other conventional CU size decision algorithms.

However, according to our knowledge, all fast algorithms have not searched the whole splitting pattern of each CTU by utilizing an optimization approach. Therefore, we utilize a simple optimizer with a meaningful chromosome pattern and a reasonable fitness function to find a good splitting pattern for each CTU. In this thesis, we propose a CU size decision method based on GA to reduce the computation complexity of the quadtree-based CU partitioning. To the best of our knowledge, we are the one who firstly introduces CU partitioning as an optimization problem which is solved by GA with an effective chromosome structure. In order to quickly find the fitness function of GA, the RD costs for each CU are calculated by encoding CUs with the most common modes for PU. To further save

the computational complexity of CU partitioning, the temporal redundancy is considered to share CU partitioning pattern within consecutive frames. Compared with the state-of-the-art SVM based fast algorithm, the proposed method can reduce a higher computational complexity at higher target bit rate under a negligible quality loss. In current communication networks including 5G, the available bandwidth is going up. Apparently, we can increase the bit rate to be high and we can consider high bit rate in video coding. Our method can get a comparable PSNR at high target bit rate. For the calculation cost for small equipment, the reduction for calculation cost is an important issue.

In order to further extend this work, we can measure the subjective evaluation of our proposed method. Additionally, We can optimize GA especially for creating a better initial population than random population and using another less time-consumption input value than RD cost to calculate the fitness function.

# References


[1] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[3] JCT-VC, HM Software, [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.5/. Accessed on: Nov 5, 2016.

[4] S. Cho and M. Kim, "Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 9, pp. 1555–1564, Sep. 2013.

[5] B. Min and R. C. C. Cheung, "A fast CU size decision algorithm for the HEVC intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, May 2015.

[6] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2017, pp. 1255–1260.

[7] T. Zhang, M. Sun, D. Zhao, and W. Gao, "Fast intra-mode and CU size decision for HEVC," *Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1714–1726, Aug. 2017.

[8] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465–470, Feb. 2013.

[9] X. Hou and Y. Xue, "Fast coding unit partitioning algorithm for HEVC," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 7–10, Jan. 2014.

[10] I. Zupancic, S. G. Blasi, E. Peixoto, and E. Izquierdo, "Inter-prediction optimizations for video coding using adaptive coding unit visiting order," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1677–1690, Sep. 2016.

[11] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on bayesian decision rule," in *2012 Picture Coding Symposium*, pp. 453–456, May 2012.

[12] K. Duan, P. Liu, K. Jia, and Z. Feng, "An adaptive quad-tree depth range prediction mechanism for HEVC," *IEEE Access*, vol. 6, pp. 54 195–54 206, 2018.

[13] X. Shen and L. Yu, "CU splitting early termination based on weighted SVM," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 4, Jan. 2013.

[14] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC encoding decisions using data mining," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 4, pp. 660–673, Apr. 2015.

[15] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu, "Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2225–2238, Jul. 2015.

[16] A. Heindel, T. Haubner, and A. Kaup, "Fast CU split decisions for HEVC inter coding using support vector machines," in *2016 Picture Coding Symposium (PCS)*, pp. 1–5, Dec. 2016.

[17] L. Zhu, Y. Zhang, S. Kwong, X. Wang, and T. Zhao, "Fuzzy SVM-based coding unit decision in HEVC," *IEEE Transactions on Broadcasting*, vol. 64, no. 3, pp. 681–694, 2018.

[18] H. Kim and R. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based bayesian decision rule," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 130–138, Jan. 2016.

[19] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 547–561, Sep. 2017.

[20] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, Oct. 2018.

[21] Z. Pan, S. Kwong, M. Sun, and J. Lei, "Early merge mode decision based on motion estimation and hierarchical depth correlation for HEVC," *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 405–412, Jun. 2014.

[22] Z. Pan, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast motion estimation based on content property for low-complexity H.265/HEVC encoder," *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 675–684, Sep. 2016.

[23] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996.

[24] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Berlin, Heidelberg: Springer-Verlag, 1996.

[25] E. E. Tun, S. Aramvith, and Y. Miyanaga, "Feature Reduction on Fuzzy Svm-based Coding Unit Decision in HEVC," *2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, pp. 1–4.

[26] E. E. Tun, S. Aramvith, and Y. Miyanaga, "Fast Coding Unit Encoding Scheme for HEVC Using Genetic Algorithm," *IEEE Access*, pp. 68010–68021, May 2019.

[27] E. E. Tun, S. Aramvith, and Y. Miyanaga, "A Fast CU Depth Estimation Algorithm for HEVC Inter Coding," *The 4th IEEE International Conference on Consumer Electronics Asia 2019.*

[28] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document M33, ITU-T Video Coding Experts Group, Austin, TX, USA, 2001.

# VITA

Ei Ei Tun was born in 1986 in Taungoo, Myanmar. She received the B.C.Tech. degree in computer technology from Computer University (Taungoo), Taungoo, Myanmar, in 2005, and M.C.Tech. degree with flying colors in computer technology from the University of Computer Studies, Yangon (UCSY), Myanmar, in 2010. She is currently a lecturer at UCSY, Myanmar and pursuing the Ph.D. degree with Department of Electrical Engineering, Chulalongkorn University, Thailand under a grant from AUN/SEED-Net, JICA. Her research interests are in the areas of video coding, especially focus on the complexity reduction of the newest video coding standards, HEVC. Ms. Ei Ei Tun was one of the Best Paper and Poster candidates in the 2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST) and Asian Universities Alliance Postgraduate Academic Forum (AUAPAF), Tsinghua University, Beijing, China, respectively.

**List of Publications**

[1] E. E. Tun, S. Aramvith, and Y. Miyanaga, "Feature Reduction on Fuzzy Svm-based Coding Unit Decision in HEVC," in 2018 International Conference on Engineering, Applied Sciences, and Technology (ICEAST), July 2018, pp. 1–4. DOI: 10.1109/ICEAST.2018.8434467.

[2] E. E. Tun, S. Aramvith, and Y. Miyanaga, "Fast Coding Unit Encoding Scheme for HEVC Using Genetic Algorithm," in IEEE Access, May 2019, pp. 68010–68021. DOI: 10.1109/ACCESS.2019.2918508.

[3] E. E. Tun, S. Aramvith, and Y. Miyanaga, "A Fast CU Depth Estimation Algorithm for HEVC Inter Coding," in The 4th IEEE International Conference on Consumer Electronics Asia 2019, 2019.