Efficient HEVC-Based Video Adaptation Using Transcoding

Efficiënte HEVC-gebaseerde videoadaptatie met behulp van transcodering

Luong Pham Van

Promotor: prof. dr. P. Lambert
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. R. Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

ISBN 978-90-8578-979-6
NUR 965
Wettelijk depot: D/2017/10.500/14

Promotor: prof. dr. Peter Lambert (Universiteit Gent)

Examencommissie: prof. Gert de Cooman (Universiteit Gent)
prof. Bart Dhoedt (Universiteit Gent)
prof. dr. ir. Adrian Munteanu (Vrije Universiteit Brussel)
dr. ir. Patrice Rondao Alface (Nokia)
dr. ir. Glenn Van Wallendael (Universiteit Gent)
prof. dr. Peter Lambert (Universiteit Gent)

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Elektronica en Informatiesystemen

Campus Ufo,
Sint-Pietersnieuwstraat 41,
9000 Gent, België

Tel.: +32-9-331.49.93

# Acknowledgments

My Ph.D. path has been a wonderful and challenging experience to me. On this adventure, I had the opportunity to do scientific research in a stimulating research group in which I have learned and worked with many excellent individuals. Their tips and assistance have directly or indirectly influenced the work presented in this dissertation. As such, I would like to take this opportunity to pay all of my tributes to the people whose insurmountable contributions have helped to bring the paper to its completion.

First of all, I would like to send a special thank to my promoter, Prof. Peter Lambert, for the great chance to perform my research within DataScience Lab (formerly Multimedia Lab). Without his unconditional support, guidance, and comments, this dissertation would not have been accomplished. In addition, I would like to express my gratefulness to Prof. Rik Van de Walle for his support during the first years of my Ph.D. journey.

I would like to thank my colleagues and ex-colleagues in Multimedia Lab (newly IDLab). I am indebted to Prof. Jan De Cock, Dr. Glenn Van Wallendael, Dr. Sebastiaan Van Leuven, and Johan De Praeter. Their collaboration in scientific research and supports after 'business' hours has certainly helped me to improve many professional skills. Appreciations go to Mr. Ruben Verhack, Mr. Niels Van Kets and others for making an exciting and enriching atmosphere in our office. Ellen Lammens, Laura Smekens, and Kristof Van Damme deserve many great thanks for taking care of the administrative work related to my Ph.D.

During the Ph.D. journey, I had a great chance to work with experts in other research groups. For very fruitful collaborations and discussions, I am grateful to Dr. Patrice Rondao Alface and Mr. Raf Huysegems in Nokia Bell Labs, Mr. Stefano Petrangeli in Internet Based Communication Networks and Services research group, and Mr. Antonio Jesus Diaz-Honrubia in Albacete Research Institute of Informatics, University of Castilla-La Mancha.

Moreover, I want to thank everyone who reviewed the chapters of this Ph.D. text, and also the members of my jury who gave me many helpful comments and suggestions that better this thesis. Furthermore, I would like to thank the anony-

mous reviewers whose comments and feedbacks certainly improve my papers that are presented in this book.

I also would like to thank my friends for all the great times. I cannot forget to thank Steffie and Johan for being with me for their unflagging faith in me during hard times that has lengthened my efforts.

Finally, words fail me to say how thankful I am to my parents and family for the possibilities they gave me to develop myself, as well as for their constant support and encouragement during the years.

*Ghent, September 2016*
*Luong Pham Van*

# Table of Contents

# List of Tables

# List of Acronyms

## A

| | |
|---|---|
| AMVP | Advance Motion Vector Prediction |
| ARFF | Attribute-Relation File Format |
| AVC | Advanced Video Coding |

## B

| | |
|---|---|
| BDBR | Bjøntegaard Delta Bit Rate |

## C

| | |
|---|---|
| CABAC | Context-Adaptive Binary Arithmetic Coding |
| CART | Classification and Regression Trees |
| CB | Coding Block |
| CBR | Constant Bit Rate |
| CFG | Configuration |
| CPDT | Cascaded Pixel-Domain Transcoder |
| CPU | Central Processing Unit |
| CU | Coding Unit |
| CRA | Clean Random Access |
| CTB | Coding Tree Block |
| CTU | Coding Tree Unit |

## D

| | |
|---|---|
| DC | Direct Current |

| | |
|---|---|
| DCT | Discrete Cosine Transform |
| DF | Deblocking Filter |
| DRS | Dynamic Rate Shaping |
| DST | Discrete Sine Transform |

## G

| | |
|---|---|
| GOP | Group Of Picture |

## H

| | |
|---|---|
| HD | High Definition |
| HEVC | High Efficiency Video Coding |

## I

| | |
|---|---|
| IDCT | Inverse Discrete Cosine Transform |
| IDR | Instantaneous Decoding Refresh |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Intra Prediction |
| ISO | International Organization for Standardization |
| ITU | International Telecommunication Union |
| ITU-T | ITU Telecommunication Standardization Sector |

## J

| | |
|---|---|
| JCT-VC | Joint Collaborative Team on Video Coding |
| JTC | Joint Technical Committee |

## M

| | |
|---|---|
| ME | Motion Estimation |

| | |
|---|---|
| MER | Misclassification Error Rate |
| ML | Machine Learning |
| MPEG | Moving Picture Experts Group |
| MV | Motion Vector |

## N

| | |
|---|---|
| NAL) | Network Abstraction Layer |

## O

| | |
|---|---|
| OL | Open Loop |

## P

| | |
|---|---|
| PPS | Picture Parameter Set |
| PSNR | Peak Signal to Noise Ratio |
| PU | Prediction Unit |

## Q

| | |
|---|---|
| QP | Quantization Parameter |

## R

| | |
|---|---|
| RA | Random Access |
| RD | Rate Distortion |
| RDO | Rate Distortion Optimization |
| RF | Random Forest |
| RQT | Residual Quad-Tree |

## S

| | |
|---|---|
| SAD | Sum of Absolute Differences |
| SAO | Sample Adaptive Offset |
| SNR | Signal to Noise Ratio |
| SPS | Sequence Parameter Set |
| SVC | Scalable Video coding |
| SVM | Support Vector Machines |

## T

| | |
|---|---|
| TB | Transform Block |
| TS | Time Saving |
| TU | Transform Unit |
| TZSearch | Test Zone Search Algorithm |

## U

| | |
|---|---|
| UHD | Ultra High Definition |

## V

| | |
|---|---|
| VCEG | Video Coding Experts Group |
| VPR | Variable Bit Rate |
| VPS | Video Parameter Set |

# Nederlandse samenvatting –Summary in Dutch–

Met een geleidelijke toename van het gebruik van multimediatoepassingen en -diensten speelt digitale video een onmisbare rol in de samenleving. Gezien de huidige ontwikkeling op het gebied van elektronica, wordt video-inhoud eenvoudig gegenereerd. Daarnaast maakt de hoge snelheid van breedbandverbindingen het mogelijk om enorme hoeveelheden video te verzenden. Dit heeft video populairder gemaakt dan ooit tevoren.

Aangezien digitale video nu eenvoudig en op grote schaal toegankelijk is geworden, is de inhoud van websites ook aangepast. Daarnaast heeft de toegenomen verspreiding van video ook geleid tot de creatie van volledig nieuwe businessmodellen en nieuwe manieren voor het delen van content. Bovendien heeft de opkomst van internettelevisie bijgedragen tot de wijdverspreide populariteit van video. In dit opzicht is het dringend nodig om de kwaliteit van de geleverde video en de gebruikerservaring te verbeteren.

Het proces van het opslaan en delen van video content is mogelijk gemaakt met behulp van geavanceerde compressietechnieken. Conceptueel gezien elimineren deze technieken de redundante informatie binnen een video met het doel om de video voor te stellen met een efficiënte representatie die merkbaar minder data vereist vergeleken met het oorspronkelijke formaat. Er is veel moeite gedaan om compressie-efficiënte te verbeteren, wat heeft geleid tot de introductie van internationale videocompressiestandaarden: H.262/MPEG-2, H.263, MPEG-4 visual en H.264/MPEG-4 AVC (H.264/AVC). Deze videocompressiestandaarden zijn voornamelijk ontwikkeld door de ISO/IEC MPEG, de ITU-T VCEG of de samenwerking tussen deze twee groepen. Recent heeft de joint collaborative team on video coding (JCT-VC) een nieuwe compressiestandaard gefinaliseerd: high efficiency video coding (HEVC) die in staat is om high-definition video te comprimeren met een bitsnelheid van 40% tot 50% vergeleken met H.264/AVC voor een equivalente perceptuele kwaliteit. Met deze superieure coderingsprestaties wordt verwacht dat HEVC H.264/AVC de volgende jaren zal vervangen voor de meeste opkomende toepassingen. Daarom is de recent gefinaliseerde HEVC-standaard gekozen in dit proefschrift.

In echte toepassingen wordt video content doorgaans geleverd over een heterogeen multimedialandschap dat gekenmerkt wordt door een grote verscheidenheid aan netwerk- en eindgebruikersbeperkingen. Het is belangrijk om rekening te houden met deze beperkingen om een efficiënte distributie te verkrijgen. In dit

opzicht zijn dus efficiënte videoadaptatietechnieken nodig. Aan de ene kant kan video gestreamd worden over een gemengd netwerk dat gekarakteriseerd wordt door verschillende bandbreedtecapaciteiten. In veel gevallen is de bandbreedte onvoldoende om de videostroom aan zijn originele kwaliteit over te zetten. Daarom is een adaptatieproces nodig dat de bitsnelheid van de video aanpast aan de bandbreedtebeperking. Anderzijds heeft de dramatische explosie binnen de consumentenelektronicamarkt geleid tot een grote verscheidenheid van multimediatoestellen. In zulke scenarios wordt een enkele video vaak afgespeeld op meerdere toestellen die gekenmerkt worden door een grote variatie van beperkingen van middelen zoals schermresolutie, processorsnelheid en geheugen. Uiteraard kan een enkel videobestand niet voldoen aan de beperkingen van elk apparaat, dus moet een adaptatie zoals bijvoorbeeld de verlaging van de bitsnelheid of de spatiale resolutie uitgevoerd worden tijdens videotransmissie naar de toestellen van eindgebruikers.

Drie gebruikelijke technieken voor videoadaptatie zijn ontwikkeld tijdens de laatste decennia: simulcast, schaalbare videocodering en transcoderen. Simulcast gebruikt verschillende kwaliteitsversies van dezelfde videostromen voor adaptatie. De andere technieken converteren daarentegen een voorgecodeerde video naar een nieuwe videostroom die voldoet aan de beperkingen van het netwerk en/of het toestel.

Conceptueel betekent dit alles dat een eenvoudige simulcast verschillende gecodeerde versies van een video genereert, waarbij elk van deze versies gericht is op een bepaald netwerk of eindgebruikerstoestel. Een passende videostroom wordt geselecteerd en doorgestuurd naar de gebruiker. Deze aanpak heeft een lage latency. Deze techniek resulteert echter in een enorme rekenkundige encodeercomplexiteit en een aanzienlijke toename van de bandbreedte-overhead in het backbone netwerk.

Schaalbare codering genereert een meerlaagse representatie van de video tijdens het coderen. In deze gelaagde stromen vertegenwoordigt de onderste laag (basislaag) de laagste kwaliteit. De extra lagen (verbeteringslagen) bestaan dan uit verbeteringen van de kwaliteit (kwaliteitsschaalbaarheid), spatiale resolutie (spatiale schaalbaarheid) of beeldsnelheid (temporele schaalbaarheid). Er zijn veel inspanningen gedaan om de prestaties van schaalbare coderingen te verbeteren, wat geresulteerd heeft in schaalbare profielen van videocompressiestandaarden zoals MPEG-2, MPEG-4 visual en HEVC. Echter, de schaalbare uitbreidingen van die standaarden worden zelden in de praktijk gebruikt. In plaats daarvan wordt voor de meeste applicaties single-layer codering gebruikt.

Vanwege het gebrek aan schaalbare stromen is de ontwikkeling van alternatieve videoadaptatietechnieken noodzakelijk. Transcodering is ingevoerd als een flexibele en efficiënte manier van adaptatie. Daarom maakt het werk in dit proefschrift gebruik van transcoderen voor het adapteren van single-layer videostromen. Een transcoder past eigenschappen van de video efficiënte aan, zoals bijvoorbeeld de quantisatieparameter die de kwaliteit en bitsnelheid van een video bepaalt, en de spatiale resolutie. Dit resulteert in een stroom met een lagere bitsnelheid die voldoet aan de beperkingen van het netwerk en de eindgebruiker.

Videoadaptatie door transcodering kan bereikt worden door ofwel een open-lus

ofwel een gesloten-lus pixeldomeinconversie via een cascade. Een open-lus transcoder implementeert de adaptatie in het gecomprimeerd domein zonder de video te hercoderen. Dit mechanisme maakt de transcoder rekenkundig efficiënte. Deze architectuur is echter wel gevoelig voor foutpropagatie, wat resulteert in significante dalingen van de codeerprestaties. Een gesloten-lus encoder in het pixeldomein door middel van een cascade daarentegen kan de foutpropagatie reduceren via het decoderen en hercoderen van de videobitstroom om de gewenste bitsnelheid te verkrijgen. Het belangrijkste probleem van een gesloten-lus transcoder is de enorme rekenkundige complexiteit geassocieerd met de hercodeerstap. Het is belangrijk om hierbij op te merken dat de transcodeeroperatie meerdere malen wordt uitgevoerd on-the-fly, wat leidt tot een groot energieverbruik in het netwerk. Deze belangrijke kwestie wordt in dit proefschrift aangepakt door efficiënte oplossingen voor het reduceren van de rekenkundige complexiteit van de gesloten-lus transcodeeraanpak te onderzoeken en voor te stellen.

Van de nieuw-ontwikkelde videocompressiestandaard HEVC wordt verwacht dat deze met betrekking tot zijn hoge compressie-efficiëntie in de komende jaren de de facto standaard zal worden in het domein van videocodering. Derhalve is onderzoek over adaptatie van deze standaard nodig. In dit kader concentreren de inspanningen in Hoofdstuk 2 en Hoofdstuk 3 van dit proefschrift zich op het optimaliseren van het transcoderen van de bitsnelheid van HEVC-stromen. Onderzoek over spatiaal transcoderen van HEVC-stromen wordt vervolgens gepresenteerd in Hoofdstuk 4.

Een kort overzicht van de HEVC-standaard wordt geleverd in Hoofdstuk 1. In HEVC is een grote blokgrootte, samen met een complex mechanisme voor blokpartitionering (quad-tree partitionering), ingevoerd voor het verbeteren van de codeerprestaties van hoge-resolutie videos. Deze verbetering gaat gepaard met een significante stijging van codeercomplexiteit, wat resulteert in een hoge rekenkundige complexiteit van een transcoder. Om dit effect tegen te gaan stelt Hoofdstuk 2 verschillende technieken voor om het blokpartitioneringsproces in het encodergedeelte van de gesloten-lus bitsnelheidtranscoder te optimaliseren. Deze voorgestelde technieken maken intelligent gebruik van de codeerinformatie van de invoervideo om het blokpartitioneringsproces vroegtijdig te beindigen en het aantal coderingsmoduskandidaten te reduceren. Deze voorgestelde aanpakken bestaan uit een methode die gebaseerd is op machinaal leren, en een nieuwe procedure voor evaluatie van partities.

De techniek gebaseerd op machinaal leren voorspelt het gedrag voor het splitsen van een blok in de gebruikelijke evaluatieprocedure in HEVC. Volgens de gebruikelijke procedure worden eerst de grote blokken (ouders) gevalueerd voordat de kleine blokken (kinderen) aan de beurt komen. Met ondersteuning door machinaal leren wordt deze partitionering vereenvoudigd tot een vroegtijdige beindiging van het splitsen. Verder is de methode gebaseerd op machinaal leren ook in staat om een complexiteitsschaalbare transrating te voorzien voor praktische gebruikersscenario's.

Het belangrijkste nadeel van de techniek gebaseerd op machinaal leren is de nood aan training en voorspelling tijdens het transcoderen. Bovendien evalueren

de huidige implementaties van HEVC blokken van grote blokgroottes naar kleinere. Hierdoor kan het splitsgedrag van kleine blokken niet worden benut voor het voorspellen van splitsingen in grotere blokken. Daarom wordt in Hoofdstuk 2 een nieuwe procedure voor het evalueren van partities voorgesteld voor het encodeergedeelte van de gesloten-lus bitsnelheidtranscoder. Vertrekkend van de blokstructuur in de invoervideostroom evalueert de nieuwe aanpak eerst de kleine blokken. Vervolgens wordt de evaluatie voorwaardelijk uitgevoerd op de ouderblokken op basis van het splitsgedrag van de kleine blokken. Simulaties tonen aan dat de voorgestelde technieken superieure transcoderingsprestaties voorzien in vergelijking met de bestaande aanpakken. Bovendien kunnen de voorgestelde methoden een waaier aan afwegingen maken tussen transratingcomplexiteit en codeerprestaties. Gebruikmakend van de voorgestelde technieken wordt de transcodeercomplexiteit gereduceerd met 82%, terwijl voor de snelste aanpak de overhead van de bitsnelheid onder de 3% blijven.

In Hoofdstuk 2 wordt het partitioneringsproces van gesloten-lus transrating adaptief beindigd en wordt het aantal coderingsmoduskandidaten beperkt. Echter, het bewegingsschattingsproces dat verantwoordelijk is voor het merendeel van de rekenkundige complexiteit van het evalueren van de coderingsmodus van een blok is nog niet geoptimaliseerd. In de encoder wordt bewegingsschatting uitgevoerd om de beste overeenkomst te vinden van een blok in referentiebeelden. Deze zoekactie is beperkt tot een vast venster. Een groot zoekvenster resulteert in het evalueren van onnodige bewegingsvectoren, wat leidt tot een hoge rekenkundige overhead voor het evalueren van de coderingsmodus van een blok. Hoofdstuk 3 reduceert de complexiteit van het bewegingsschattingsproces door gebruik te maken van de correlatie tussen de bewegingsinformatie van het uitvoerblok en het overeenkomstige blok in de invoervideo. Een adaptieve zoekvenster gebaseerd op beslissingstheorie en snelle zoekalgoritmen wordt grondig besproken in dit hoofdstuk. Een evaluatie van de prestaties van de voorgestelde methode voor bewegingsestimatie toont dat de complexiteit van de geoptimaliseerde transcoder in Hoofdstuk 2 verder kan worden verlaagd met gemiddeld 16% en tot 20% voor dezelfde rate-distortion prestaties. Dit leidt tot een aanzienlijke vermindering van de totale transcodeercomplexiteit van gemiddeld 84% tot 87%.

Spatiaal transcoderen wordt vaak gebruikt wanneer de bitsnelheidsverlaging meer dan 50% is of wanneer er behoefte is aan adaptatie van de resolutie. Hoofdstuk 4 behandelt de problemen die voorkomen bij spatiaal transcoderen van HEVC. In de praktijk is de verhouding van spatiale verkleining arbitrair wanneer een spatiale transcoder gebruikt wordt. Een arbitraire verkleiningsfactor kan resulteren in een verschillende uitlijning tussen het verkleinde blok en de overeenkomende blokken in de originele video. Bovendien is er een significant verschil in de transformerende structuur tussen HEVC en de bestaande standaarden. Deze twee zaken maken het efficiënt om bestaande verkleiningstechnieken rechtstreeks toe te passen op een HEVC-videostroom met een arbitraire factor. Om deze problemen aan te pakken wordt een nieuwe aanpak die gebruikmaakt van machinaal leren besproken in Hoofdstuk 4.

In Hoofdstuk 4 wordt de correlatie tussen de coderingsinformatie van de invoer-

en uitvoervoervideo op effectieve wijze gebruikt voor het reduceren van de rekenkundige complexiteit van de transcoder. Dit hoofdstuk onderzoekt hoe modellen gebaseerd op machinaal leren gegenereerd worden om het splitsgedrag van blokken in de uitvoervideo van een gesloten-lus spatiale HEVC-transcoder te voorspellen. Met zo een aanpak van machinaal leren moeten de voorspellingsprestaties zo hoog mogelijk zijn terwijl de complexiteit van training en predictie beperkt moeten zijn. Twee technieken worden voorgesteld om zulke hoge voorspellingsprestaties te bereiken. Ten eerste zijn er verschillende algoritmen van machinaal leren gevalueerd om te bepalen welke in de beste prestaties resulteert. Onder de gevalueerde algoritmen vertoont het random forest algoritme de beste voorspellingsprestaties. Ten tweede worden de parameters van het machinaal leren algoritme adaptief geoptimaliseerd door gebruik te maken van een content-adaptief selectiecriterium. De trainingstijd en de overfitting van het machinaal leren algoritme zijn beperkt door adaptieve feature-dimension-reduction technieken. Deze optimalisaties zijn duidelijke evoluties van de voorgestelde technieken ten opzichte van de bestaande transcodeeralgoritmen gebaseerd op machinaal leren, waar er geen rekening gehouden werd met de problemen van machinaal leren. Zoals vermeld in de experimentele resultaten in Hoofdstuk 4, toont de voorgestelde techniek superieure prestaties in vergelijking met de huidige state-of-the-art methoden. Zo kan de voorgestelde aanpak bijvoorbeeld 65% van de transcodeercomplexiteit reduceren terwijl de state-of-the-art slechts een vermindering van 50% bereikt met dezelfde rate-distortion prestaties. Bovendien zorgen de voorgestelde technieken voor een complexiteitsschaalbare transcodering die de rekenkundige complexiteit vermindert met 72% terwijl de verliezen van bitsnelheid beperkt blijven tot 5%.

Samenvattend pakt het werk dat beschreven wordt in dit proefschrift, het belangrijkste probleem van videoadaptatie aan: de hoge rekenkundige complexiteit. Verschillende technieken zijn hiervoor voorgesteld die de transcodeercomplexiteit van een HEVC-stroom kunnen reduceren. Dit leidt tot een vermindering van het energieverbruik van transcodering in het netwerk. In het algemeen voorspelt de voorgestelde methode de codeerinformatie in het encodergedeelte van de transcoder. Resultaten van simulaties tonen dat het werk op effectieve wijze de transcodeercomplexiteit van een HEVC-stroom reduceert. Hoewel het voorgestelde werk gevalueerd is in de context van het transcoderen van een reeds gecodeerde HEVC-stroom, gelooft de auteur dat het werk ook toepasbaar is op bredere applicaties in andere contexten, zoals het optimaliseren van een schaalbare HEVC-encoder, heterogene transcoders van voorgaande standaarden naar HEVC, multi-bitsnelheidcodering, of de HEVC-encoder.

# English summary

With a steady increase in the use of multimedia-related applications and services, digital video plays an indispensable role in society. Given the current development in the field of electronics, video content is easily generated. Additionally, the high speed of the broadband connection allows transmitting a huge amount of video. This has made video more popular than ever before.

Since digital video has now become easily accessible and widely available, the content of websites has changed accordingly. In addition, the increased spread of video has led to the creation of completely new business models and new ways of sharing content. Furthermore, the emergence of internet television has helped video content gain its widespread popularity. In this regard, improving the quality of delivered videos and the user experience is a crucial need.

The process of storing and sharing video content has been made feasible by means of advanced compression technologies. Conceptually, these techniques eliminate the redundant information existing within a video to present the video in an efficient representation which requires notably less data than the original format. A great deal of effort has been put into improving compression efficiency, leading to the introduction of international video coding standards: H.262/MPEG-2, H.263, MPEG-4 visual, and H.264/MPEG-4 AVC (H.264/AVC). These video coding specifications have primarily been developed by the ISO/IEC MPEG, ITU-T VCEG group, or the collaboration between these two groups. Recently, the joint collaborative team on video coding (JCT-VC) has finalized a new coding standard: high efficiency video coding (HEVC) - which is capable of compressing high definition video with a bit rate reduction of 40% to 50% compared to H.264/AVC for an equivalent perceptual quality. With this superior coding performance, HEVC is expected to be widely used in most emerging applications in the next couple of years. Therefore, the latest video coding standard HEVC has been chosen in this dissertation.

In real applications, video content is typically delivered in a heterogeneous multimedia landscape featured by a great diversity of network/end-user constraints. It is important to take into account these constraints to obtain an efficient distribution. To this extent, efficient video adaptation techniques are required. On the one hand, video content might be streamed over a miscellaneous network that is characterized by different bandwidth capacities. In many cases, the bandwidth is insufficient to transfer the video stream at its original quality. Therefore, an adaptation process is needed to match the video bit rate to the bandwidth limitation. On the other hand, the dramatic explosion within the consumer electronic market

has resulted in a great diversity of multimedia devices. As such, a single video is often played by multiple devices characterized by a huge variation of resource constraints including resolution of display screen, CPU speed, and memory. Obviously, a single video file would not optimally satisfy every device constraint; thus, an adaptation, e.g., a reduction of the bit rate or spatial resolution, should be performed during a video transmission to end user's devices.

Three usual video adaptation techniques have been developed during the last decades, i.e., simulcast, scalable video coding, and transcoding. The simulcast solution uses multiple different quality video streams for adaptation. In contrasts, the other techniques convert a pre-encoded video into a new video stream that satisfies the network or/and device constraints.

Conceptually, the simple simulcast approach generates different encoded versions of a video, each of which is able to satisfy a certain condition of the network or user's devices. A proper video stream is selected to transmit to the user. The simulcast approach is low-latency. However, this technique leads to a huge computational complexity of encoding and to a notable increase in bandwidth overhead in the backbone network.

Scalable coding generates a multi-layer representation of the video during encoding. In these layered streams, the lower layer (base layer) represents the lowest quality. Meanwhile, additional layers (enhancement layers) consist of refinements of the quality (quality scalability), the spatial resolution (spatial scalability), or possibly frame rate (temporal scalability). Many efforts have been put into improving the performance of scalable coding, resulting in the scalable profiles of many video coding standards such as MPEG-2, MPEG-4 visual, and HEVC. However, the scalable extensions of those standards have rarely been used in practice. In contrast, the single-layer coding is deployed in most applications.

Due to the lack of scalable streams, developing alternative video adaptation techniques is necessary. Transcoding has been introduced as a flexible and efficient adaptation solution. Therefore, the work in this dissertation makes use of transcoding for adapting single-layer video streams. The adaptation ability of a transcoder is made by modifying a property of the video, e.g., quantization parameter that decides the quality and bit rate level of a video, frame rate, and spatial resolution in an efficient manner. This results in a lower bit rate stream that satisfies the network and user-device constraints.

Video adaptation using transcoding can be achieved by either an open-loop or a closed-loop cascaded pixel domain conversion. An open-loop transcoder implements the adaptation in the compressed domain without re-encoding the video. This mechanism makes the transcoder computationally effective. However, this architecture is subject to error propagation, resulting in significant losses in the coding performance. In contrast, a closed-loop cascaded pixel domain transcoder can reduce the error propagation via decoding and re-encoding the video bitstream to obtain the desired bit rate. The main issue of a closed-loop transcoder is the huge computational complexity associated with the encoder part. It is important to note that the transcoding operation is performed multiple times on-the-fly, leading to huge energy consumption in the network. In order to address this critical issue,

the work in this dissertation investigates and proposes efficient solutions to reduce the computational complexity of the closed-loop transcoding approach.

The newly developed video coding standard HEVC is expected to be the de facto benchmark in the domain of video coding in the next couple of years with respect to its high compression efficiency. Consequently, investigation on adaptation for this standard is needed. To that extent, the effort in Chapter 2 and Chapter 3 of this dissertation optimizes the bit rate transcoding of HEVC streams. Research on spatial transcoding of HEVC streams is presented in Chapter 4.

A brief overview of the HEVC standard is provided in Chapter 1. In HEVC, a large block size, together with a complex block partitioning mechanism (i.e., quad-tree partitioning), has been adopted to improve coding performance of high-resolution videos. This improvement came along with a significant increase in encoder complexity, resulting in a high computational complexity of a transcoder. To deal with this, Chapter 2 proposes several techniques to optimize the block partitioning process in the encoder part of the closed-loop HEVC bit rate transcoder. These proposed techniques intelligently make use of the coding information of the input video to early terminate the block partitioning process and reduce the number of coding mode candidates. These proposed approaches consist of a machine-learning-based method and a novel partition-evaluation flow.

The machine-learning-based approach predicts the splitting behavior of a block in the use of the conventional evaluation flow that has been adopted in HEVC. In this flow, the evaluation is applied to large blocks (parents) first before being applied to small blocks (children). With the support of machine learning, the partitioning is simplified with an early termination of splitting. Also, the machine-learning-based method is capable of providing a complexity-scalable transrating scheme for practical use cases.

The main downside of the machine-learning-based technique is the need of training and prediction during transcoding. Moreover, the current implementations of HEVC evaluate blocks from larger sizes to smaller ones. As such, the splitting behaviour of small blocks cannot be exploited for predicting splits of larger blocks. Therefore, a novel partition-evaluation flow is proposed for the encoder part of the closed-loop bit rate transcoder in Chapter 2. Starting from the block structure in the input video stream, the novel approach evaluates small blocks first. Then, the evaluation is conditionally carried out on the parent blocks given the splitting behavior of the small blocks. Simulations show that the proposed techniques clearly provide a superior transcoding performance compared to the state-of-the-art approaches. Additionally, the proposed methods can achieve a range of trade-offs between the transrating complexity and coding performance. From the proposed schemes, the transcoding complexity is reduced by 82% with the fastest approach while bit rate overheads remain below 3%.

In Chapter 2, the partitioning process of closed-loop transrating is adaptively terminated together with a reduced number of coding mode candidates. However, the motion estimation process - which is responsible for the majority of the computational complexity of evaluating the coding mode of a block, has not yet been optimized. In the encoder, motion estimation is performed to find the best match

of a block in reference frames. The search is limited to a fixed window. A large search window results in evaluating unnecessary motion vectors, which leads to a high complexity overhead for evaluating the coding mode of a block. Chapter 3 reduces the complexity of the motion estimation process by exploiting the correlation between the motion information of the output block and the co-located blocks in the input video. An adaptive search window based on decision theory and fast search algorithms are discussed in-depth in this chapter. A performance evaluation of the proposed motion estimation scheme shows that the complexity of the optimized transcoder in Chapter 2 can be further reduced by 16% on average and up to 20% for the same rate-distortion performance. This leads to a notable reduction of total transcoder complexity of 84% on average and up to 87%.

Spatial transcoding is often used when the bit rate reduction is higher than 50% or when there is a need for resolution adaptation. Chapter 4 deals with problems raised in spatial transcoding of HEVC. In practice, the spatial reduction ratio is arbitrary when a spatial transcoder is used. An arbitrary downsizing factor may result in a misalignment between the downsized block and its collocated blocks in the original video. Moreover, there is a significant difference in the transforming structure between HEVC and the existing standards. These two issues have made the straightforward existing downsizing techniques inefficient to be applied for downsizing an HEVC video stream by an arbitrary factor. In order to address these issues, a novel approach that makes use of machine learning will be discussed in Chapter 4.

In Chapter 4, the correlation between the coding information of the input and output video is effectively utilized for reducing the computational complexity of the transcoder. This chapter examines how machine-learning models are generated to predict the splitting behaviour of blocks in the output video of a closed-loop spatial HEVC transcoder. In such a machine-learning-based approach, the prediction performance should be as high as possible whereas training and prediction complexity should be limited, and two techniques have been proposed in order to achieve a high prediction performance. First, different machine learning algorithms have been evaluated to figure out which one provides the best performance. Among the evaluated algorithms, the random forest algorithm shows the highest prediction performance. Second, the parameters of machine learning algorithms are adaptively optimized using a content-adaptive selection criterion. The training time and over-fitting of the machine learning algorithms have been limited by adaptive feature-dimension-reduction techniques. These optimizations are clearly the advances of the proposed techniques compared to the existing machine-learning-based transcoding algorithms where the issues of machine learning have not been taken into account. As reported in the experimental results in Chapter 4, the proposed technique demonstrates superior performance compared to the state-of-the-art methods, e.g., the proposed approach is able to reduce 65% of the transcoding complexity while the state-of-the-art can achieve a 50% reduction with the same rate-distortion performance. Moreover, the proposed techniques provide a complexity-scalable transcoding scheme, which can reduce the computational complexity by 72% while maintaining bit rate losses below 5%.

In conclusion, the work described in this dissertation tackled the main problem of video adaptation, i.e., its high computational complexity. Several techniques have been proposed that are able to reduce the transcoding complexity of an HEVC stream. This leads to a reduction of energy consumption of transcoding in the network. Generally, the proposed method predicts the coding information in the encoder part of the transcoder. Simulation results show that the work effectively reduces the transcoding complexity of an HEVC stream. Although the proposed work has been evaluated in the context of transcoding a pre-encoded HEVC stream, it is the author's belief that it has broader applications in different contexts such as in optimizing the scalable high efficiency video encoder, heterogeneous transcoders from former standards to HEVC, multi-bit rate coding, or the HEVC encoder.

# 1 Introduction

## 1.1 Context

The last decades have witnessed an explosive growth in the use of digital video in a wide range of applications: from multimedia messaging, video telephony and conferencing over mobile TV or streaming video over the Internet to digital television broadcasting. This increase is reflected in an impressive number of 1.1 zettabytes video streamed over IP in 2014 that accounts for a large portion of 64 percent of the global IP traffic [1]. Forecasts indicate that IP video traffic will occupy 80 percent of all consumers' Internet traffic in 2019. Three important aspects are considered in order to obtain an efficient mechanism for video delivery: first, the bit rate of the video with its significant impact on network data traffic; second, the visual quality of the video that affects user experience; and finally, the complexity of video processing that results in the network latency as well as the energy required for network operations.

Among the three aforesaid aspects, video bit rate can be efficiently optimized by compression techniques, purposely to present the video in a form that requires as less data as possible, while it can be reconstructed with a high visual quality. This efficient presentation of video is obtained by eliminating spatial and temporal redundancies in uncompressed videos. A lot of effort has been put into improving the compression ratio, leading to different video coding standards during the last two decades. These coding standards are primarily developed by the ITU Telecommunication standardization sector video coding experts group (ITU-T VCEG) and

the ISO's moving picture experts group (MPEG).

During the transmission of video streams to users, it is important to take into account the network and device constraints. Video content is typically delivered in a heterogeneous multimedia environment. In such a ubiquitous landscape, a single video may not satisfy the network and device constraints. On the one hand, networks often bear fluctuating bandwidth that usually brings about temporary capacity problems. Having a video bit rate higher than the network bandwidth results in visual artifacts as a consequence of packet losses. On the other hand, at the users' side, playback devices are featured by a great diversity of characteristics including display resolution, processing power, battery lifetime, network connectivity, etc. Streaming a high-resolution video to low resolution screens or low processing power devices is not efficient, since these devices are not capable of displaying entire pixels of such a high-resolution video. Moreover, the high-resolution video requires high energy and buffer storage to decode and display the video.

These aforementioned constraints lead to the need for video adaptation techniques and optimized video distribution approaches. These approaches include simulcast, scalable encoding and transcoding - three video adaptation approaches which have been introduced since a long time.

The remainder of this chapter is organized as follows. Video coding standards are briefly introduced in Section 1.2. In this section, the latest high efficiency video coding standard is elaborated on. The introduction is necessary to understand the core standard which is used throughout the story of this thesis. Thereafter, an introduction to the video adaptation approaches is presented in Section 1.3. Then, the outline of the thesis that describes the author research on the optimization of video transcoding for HEVC video streams is given in Section 1.4. Finally, the publications of the author's research achievement are listed in Section 1.5.

## 1.2 High Efficiency Video Coding

In the very early moment of digital video era, ITU-T introduced the first international video coding standard H.120 that has then been developed to form H.261, H.262, and H.263 [2]. In 1993, MPEG introduced MPEG-1 that demonstrated a superior quality to H.261 when operated at higher bit rates. Afterwards, the Joint Video Team of the ISO/IEC MPEG and ITU-T VCEG groups standardized MPEG- 2/H.262 [3] in 1994. Based on this successful collaboration, these teams have then jointly developed a new coding standard fully completed in 2003 as MPEGH.264/AVC [4]. Currently, H.264/AVC is being adopted in a wide range of applications such as digital television, video telephony and video conferencing over mobile TV, video streaming over wireless and internet, video storage.

Nowadays, high definition (HD) video and beyond HD (*e.g.*, $4k\times2k$, $8k\times4k$ resolution) formats are being more widely used than ever before. In addition, the

stereo or multi-view presentation of such high-resolution video is popular. Moreover, the use of these video is conspicuously increased in a great diversity of services including multimedia messaging, video telephone, and video conferencing over TV, wireless, and video surveillance, *etc*. The growing popularity of high-resolution video and the rising diversity of video services have led to a strong demand for coding efficiency superior to H.264/AVC performance. To satisfy this crucial need, the emergent high efficiency video coding standard (HEVC) has been designed by the Joint Collaborative Team on Video Coding (JCT-VC) [5]. By deploying many advanced coding tools, HEVC demonstrates a significantly improved compression performance as compared to H.264/AVCin the range of 40% to 50% bit rate reduction at the same reproduction quality.

The following sections demonstrate key features of the HEVC specification. First, the coding design including high-level syntax and block partitioning of an HEVC stream is described in Section 1.2.1. Thereafter, the coding tools yielding most of the compression gains for HEVC are presented in 1.2.2.

## 1.2.1 HEVC coding design

### 1.2.1.1 High level syntax

The high-level syntax architecture of HEVC is designed such that the flexibility is improved as much as possible for operation over a variety of applications and network environments. The shape of the high-level syntax architecture of H.264/MPEG-4 AVC has generally been retained to HEVC, in which advanced coding tools have been added or replaced to achieve higher compression efficiency.

An HEVC bitstream is made up of a sequence of so-called network abstraction layer (NAL) units. These units are classified into two types including video coding layer (VCL) and non-VCL NAL units.

Non-VCL NAL units contain parameter sets specifying high-level information of the entire coded video sequence or a subset of pictures. By modifying and extending Sequence Parameter Set (SPS) and Picture Parameter Set (SPS) in NAL units of H.264/AVC, an additional set called Video Parameter Set (VPS) has been included to non-NAL units of HEVC. VPS contains information regarding the overall video stream including the number of layers, the dependencies between layers, and their associated profile and level. This design of VPS enables the compatible extensibility of the standard of which multiple scalable layers and multiple views are supported. The information in SPS specifies characteristics of a single sequence containing multiple pictures, which consists of the width, the height, cropping parameters, bit depth, and so on.

VCL NAL units represent coded pictures. A picture may be partitioned into one or more slices. Doing so, the encoder and decoder can be accelerated with parallel processing devices since the cross prediction among slices is not enabled.

#### 1.2.1.2 Block partitioning

**A. Coding unit partitioning**

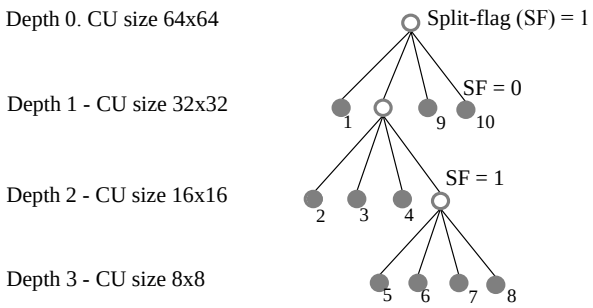
HEVC provides a novel partitioning mechanism which divides a picture into small blocks. In its first division, each picture is split into a sequence of same-size square coding tree units (CTUs), each of which consists of a luma coding tree block (CTB), the corresponding chroma CTBs and syntax elements. While H.264/AVC fixes the size of marcroblocks by $16\times16$ samples for all configurations, larger sizes of luma CTB (i.e., powers of two) are enabled in HEVC. It should be noted that larger CTU size brings better compression and higher encoding complexity. Having been indicated as a good rate-distortion-complexity trade-off point, a $64\times64$ size is used default for every configuration.

In the second step of the division, each CTU serves as a root of a block partitioning quad-tree structure where the CTU can be further divided along the coding tree structure into coding units (CUs), each of which has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs). A CU can be encoded by either skip, inter, or intra prediction modes. To obtain the most efficient mode for a CU, all PU partitions and all Residual Quad-Tree (RQT) configurations are evaluated during the rate distortion optimization (RDO) process. This RD-evaluation decides an optimal mode by minimizing the RD-cost function ($J$) given by $J = D + \lambda R$, where $D$ is the distortion of the reconstructed CU, $\lambda$ the Lagrangian multiplier and $R$ the rate required to signal the prediction and the residual information of the CU. Since any combination of quad-tree splitting is allowed, in worse case the encoder should perform the RD-cost optimization process for 83522 ($(2^4 + 1)^4 + 1$) CUs to find the optimal structure of a CTU. Figure 1.1 illustrates an example of splitting a CTU into sub CUs.

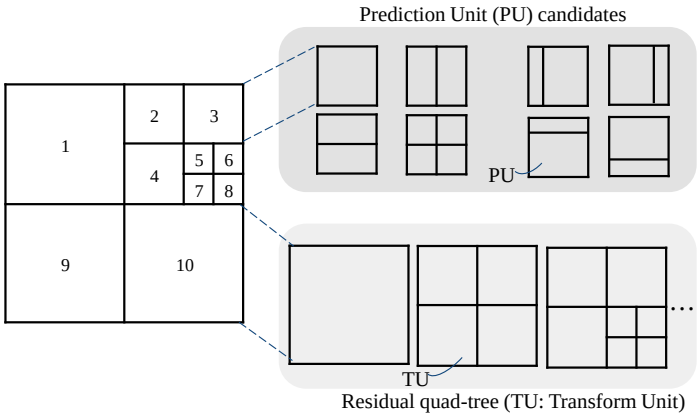
HEVC encoder processes CTUs in a slice using raster scan order. Figure 1.2 represents the division of a frame into CUs. As soon as the optimal structure of a CTU is derived, the coding information (e.g., split flag, coding mode, PU mode, TU structure, residual information, and so on) of this CTU and that of every CU within this CTU is signalled to the decoder in Z-scan order (Figure 1.1).

**B. Prediction unit partitioning**

A CU can be further divided into prediction units, each of which is made up of a luma prediction block (PB) and their corresponding chroma PBs. Variable PU sizes are enabled from $64\times64$ down to $4\times4$ samples. However, the size of PU must not exceed the size of its root CU. Every intra predicted CU is associated with one PU of the same size except for the smallest CU that is allowed in the bitstream. The smallest CUs can be split into 4 same-sized squared PUs. An inter-coded CU can remain as one block, or be divided into two or four sub PUs. Let $2N\times2N$ be the size of a CU, then possible PUs of inter coded CUs can be categorized into two groups including symmetric partitions (i.e., $2N\times2N$, $N\times2N$, $2N\times N$, and




(a) The CTU is split into coding units (CUs) using the quadtree structure.




(b) CU servers a root of PU and TU partitioning.

**Figure 1.1:** Example for the partitioning of a $64 \times 64$ coding tree unit (CTU).

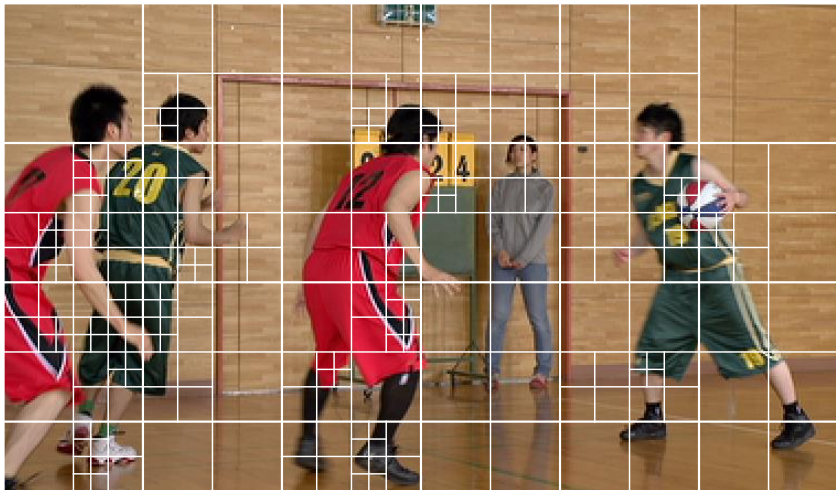


**Figure 1.2:** Example of splitting a picture into coding units.

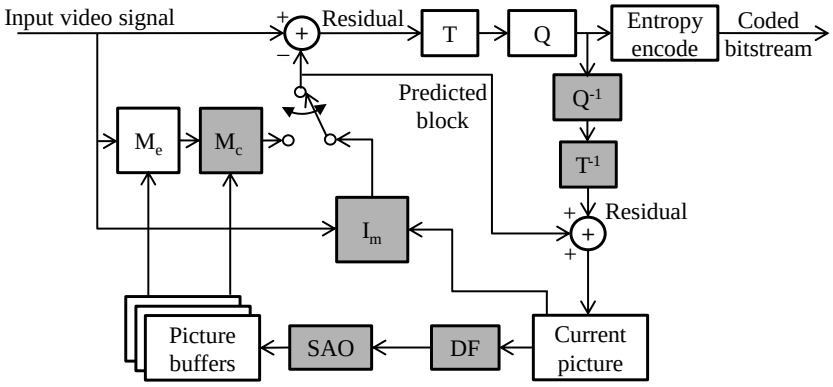



**Figure 1.3:** Typical HEVC encoder (with decoder components shaded in light gray) [5].

$N \times N$) and asymmetric partitions ($nL \times 2N$, $nR \times 2N$, $2N \times nU$, and $2N \times nD$) as illustrated in Figure 1.2. The $N \times N$ partition is applied for splitting of the smallest CUs that are allowed in the bitstream.

The optimal coding mode of a CU and its partitioning in PU as well as TU levels are derived by performing encoding loop (Figure 1.3) which optimizes the rate-distortion cost ($J$). The prediction mode of a CU is signalled to indicate whether it is an intra or an inter block.

### C. Transform unit partitioning

As the last step of partitioning, a CU can be split into transform units (TUs)

which associates with a luma transform block (TB) and their corresponding chroma CBs, and syntax. The splitting is performed recursively on a quadtree approach as depicted in Figure 1.1 where the corresponding CU serves as the root. The size of a TU can vary from $4\times4$ up to $32\times32$ samples.

Within a TU, residual information is derived by subtracting the predicted samples and the samples in the original image. The residual information is then transformed (T) and quantized (Q) as depicted in Figure 1.3. The transformation procedure is carried out by either an integer derivation from the discrete cosine transform (DCT) or the discrete sine transform (DST). While H.264/AVC fixes the size of transform matrix by $4\times4$, larger sizes of transformation matrix corresponding to the size of TU are considered in HEVC.

After quantizing, the quantized coefficients together with all other signalled parameters from the CU, PU, and TU levels are entropy encoded with the use of a context-based adaptive binary arithmetic (CABAC) engine [6]. It should be noted that when the CU is signalled as skip, no residual information is encoded.

#### 1.2.1.3 Random Access

The introduction of random access concept allows users to start a video at a specific point or to switch between different channels in broadcasting or streaming applications. H.264/AVC offers the random access capability by inserting instantaneous decoding refresh (IDR) pictures in interval into the video stream. The decoder is reset to the original configuration of an IDR picture before decoding this picture. This mechanism enables the decoder to start at any IDR point. Unfortunately, the pictures reconstructed prior to the IDR point cannot be used as temporal references for the pictures following the IDR position in decoding order. As a result, the coding performance with the use of IDR is significantly lowered.

To solve the aforementioned downside of using IDR scheme while retaining the random access functionality, HEVC introduces clean random access (CRA) pictures. A CRA picture allows the decoder to reset after all preceding pictures in display order have been decoded. Consequently, pictures following the CRA picture in decoding order but preceding it in display order can use pictures decoded before the CRA picture as reference.

### 1.2.2 HEVC coding tools

The typical diagram of an HEVC encoder is depicted in 1.3. The encoder performs coding tools, which are supported in HEVC in order to find optimal coding strategies for a coding unit. Basically, HEVC is a hybrid coding algorithm that involves several coding tools including inter, intra predictions to respectively exploit temporal, and spatial statistical dependences, and transform coding of the prediction residual information to further exploit spatial statistical dependency. These coding

tools have been implemented in prior block-based coding standards; however, in the new HEVC coding standards, each is notably improved to contribute to the great compression efficiency of HEVC.

#### 1.2.2.1 Slices and tiles

A picture is divided into one or more slices. A slice consists of a sequence of CTUs processed in the order of a raster scan. Slices are independently encoded. This means the cross prediction (e.g., spatial signal prediction or prediction of motion vectors) between slices are not allowed. This facilitates decoding multiple slices at the same time. Hence, playback devices, which support parallel processing, can accelerate the reconstruction process.

In terms of possible prediction modes that allow encoding CUs within a slice, there are three types of slices:

1. *I slice*: All CUs of the slice are coded using only intra prediction.

2. *P slice*: A CU in this slice is encoded using either intra prediction or inter prediction with at most one motion-compensated prediction signal per PU. Only reference pictures in list 0 are used for P slices.

3. *B slice*: All prediction modes that are available in P slice can be used in B slice. The difference between B and P slices is that inter predicted blocks in B slice can use at most two motion compensated prediction signals. Both reference picture list 0 and list 1 are available for B slices.

While slices have already been introduced in H.264/AVC, HEVC defines a new concept of division of a picture known as tiles. Each tile is an independently decodable rectangular region of the picture. The design of tiles is mainly to enable the use of parallel processing architectures for decoding and encoding. Alternative to slices, a single tile may consist of multiple slices.

#### 1.2.2.2 Intra prediction

The purpose of intra prediction is to exploit the spatial correlation among pixels within a frame. Conceptually, the predicted samples of a PU are extrapolating samples in spatially neighboring reconstructed blocks using one of supported prediction modes. The residual information is derived by subtracting the predicted block from original samples in the picture. Subsequently, the residual block get transformed, quantized, and entropy encoded. The selected intra prediction mode is also encoded and signalled to decoder together with encoded residual information.

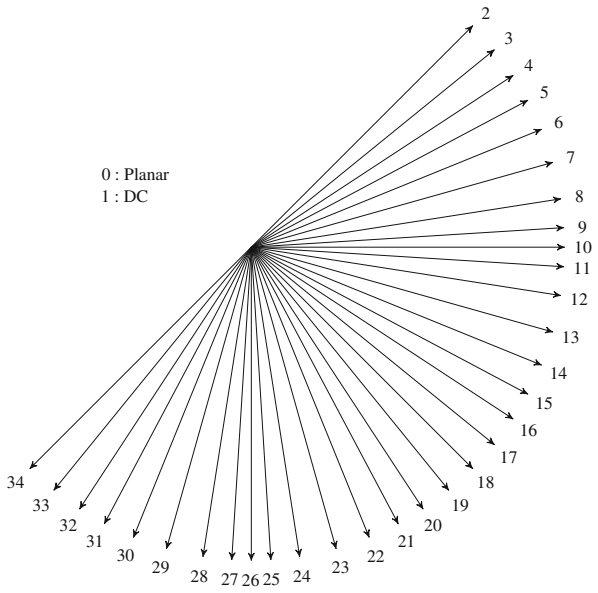



**Figure 1.4:** Angular intra prediction modes (2 - 32), non-angular modes (planar 0, DC 1).

H.264/AVC supports nine intra prediction modes, while HEVC improves the efficiency of intra coding by extending the number of supported modes to 35. Supported intra prediction modes are categorized into angular and non-angular groups as depicted in Figure 1.4. The optimal prediction mode, which is used to encode a PU, is selected from a set of available modes by a rate-distortion optimization process of encoding loop as illustrated in Figure 1.3.

The angular intra prediction modes are numbered from 2 to 34. When using an angular intra mode, each PU is predicted directionally from spatially neighboring reconstructed samples known as reference samples. The reference samples of a sample in the predicted block are indicated by projecting the location of the sample to the reference sample array following the angle of the selected prediction mode. Then, the predicted sample value is derived by performing bilinear interpolation at 1/32 sample accuracy using two closest reference samples in the direction of prediction.

While angular prediction modes target regions with direction edges, DC mode and planar mode are designed for smooth image areas. In the DC mode, all samples in the predicted block share the average value of reference samples. On the other hand, the predicted samples in planar mode are the average values of two linear prediction using four corner reference samples.

In order to improve the prediction accuracy of intra prediction, the reference samples might be filtered by a three-tap [1 2 1]/4 smoothing filter. The filtering

process is performed adaptively, according to the amount of discontinuity, the direction of prediction mode, and the block size. In addition to smoothing reference samples, predicted samples of using certain three modes (i.e., DC, exactly vertical mode 10, and exactly horizontal mode 26) are filtered to remove discontinuity along block boundaries.

The use of a large number of intra prediction modes results in a notable increase of overhead for signalling the index of selected mode to the decoder. To reduce the signalling overhead, the mode is encoded before being sent to the decoder. The detail of intra mode coding is presented in [7].

#### 1.2.2.3 Inter prediction

Inter prediction serves as a key tool in hybrid block-based coding. This tool is to eliminate the temporal redundancy between successive pictures in time domain, resulting in a significant reduction of bit rate of video stream.

In HEVC, inter prediction operates on the prediction unit (PU) level, of which eight types of PU partitioning are available. The inter prediction process searches for the best match block of a PU in previously reconstructed pictures (known as reference pictures) by performing a so-called motion estimation procedure. The best match is defined as the block with the least mathematical distortion compared to the input block. The displacement of the current PU relative to its best match is named motion vector (MV) that is represented by two motion components (i.e., horizontal and vertical directions). The residual information, which is obtained by subtracting the best matched block from the original block, together with motion vectors are encoded and sent to the decoder.

HEVC supports two types of inter prediction including uni-prediction and bi-prediction as depicted in Figure 1.5. While uni-prediction can use only one previously reconstructed picture as a reference frame, bi-prediction can use two reference frames. Uni-prediction is applied for PUs in P slices, where only one reference list is available. In contrast, the latter type of prediction is applied for PUs in B slices with two reference lists. The concept of this weighted prediction, which assigns a weight factor to each reference picture, is also integrated in HEVC.

In order to improve the performance of inter coding, HEVC supports subsample interpolation in two levels, i.e., half and quarter pel samples.

Based on the residual information in PU and the way that motion vectors are encoded, coding mode of an inter PU can be classified into three types including merge mode, skip mode, and predictive motion vector coding (PMVC) mode. The motion estimation process performs minimization of the rate-distortion cost to decide the optimal coding mode of a PU as depicted in Figure 1.3. More insights into the motion estimation is presented in Section 1.2.2.4.

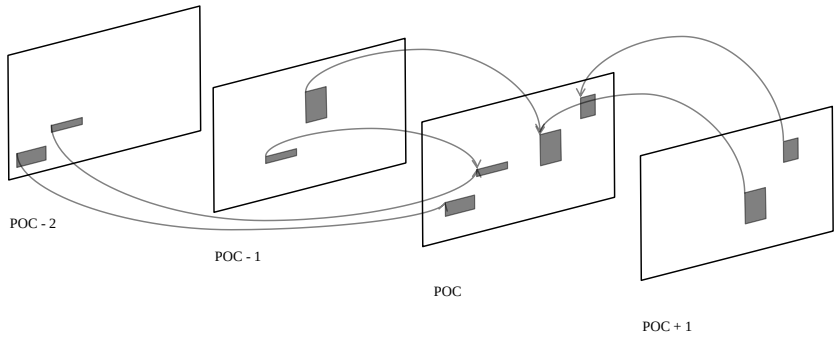



**Figure 1.5:** Uni- and bi-prediction for the current picture using adjacent reference pictures.

#### 1.2.2.4 Motion estimation

The motion estimation process evaluates the RD costs of three coding modes to select the optimal one for a PU.

**Merge mode:** For merge mode, the motion vector of the CU is derived from a candidate set. The candidate motion vectors are selected from the motion vectors of spatially neighboring encoded PUs (e.g., top, left, top-let, top-right, and left-bottom PU) and the motion vector of the collocated location in a reference picture. All of candidates are evaluated. The motion vector having the minimum RD cost is selected to encode the current PU. For the merge mode, the index of the selected motion vector and residual information is signalled to the decoder.

**Skip mode:** If the CU is signalled as skip mode, it has only one PU with the size of the CU. The optimal motion vector of the CU is derived using the same criterion of merge mode. However, there is no residual information being coded. Therefore, the encoder only signals the index for the merge candidate to the decoder if the number of merge candidates is larger than one. At the decoder, a skip CU is reconstructed by copying the block in the reference frame presented by the motion vector which derived by the described merge operation.

**PMVC mode:** Similar to the merge mode, a predictor motion vector is selected among multiple predictor candidates that are obtained by a so-called advance motion vector prediction (AMVP). The difference between the actual motion vector and the predictor together with the index of the predictor in the candidate list are encoded and transmitted to the decoder.

In the reference software HM [8], the actual integer motion vector of a PU is searched in a window centered by the predictor in reference frames. The TZSearch

algorithm is applied to reduce the complexity of the motion estimation process. The integer motion vector is then refined at half- and quarter-pel sample levels.

#### 1.2.2.5 In-Loop filters

HEVC performs two filter steps, namely a deblocking filter (DBF) followed by a sample adaptive offset (SAO) filter, to improve quality of the reconstructed images. While the deblocking filter of HEVC is similar to that of H.264/AVC, SAO is newly proposed in HEVC.

The purpose of deblocking filter is to reduce the blocking artifacts along block boundaries including both PU and TU boundaries. Strength of the filter is assigned to a block boundary based on the coding information of the two blocks that share this boundary. If the strength is decided to be 0, the filter is not performed. Otherwise, a low-pass filter is adaptively applied. Since both the encoder and decoder carry out the same filter procedure, there is no information of filter, which needs to be signalled.

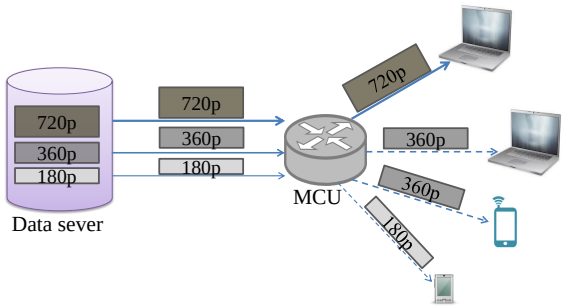
SAO conditionally adds an offset to each sample of a CTU after deblocking filtering. The offset values are defined in look-up tables, which are transmitted by the encoder. In use of SAO, an SAO type and offset are decided and signalled to the decoder. By performing a fixed interpolation filters on the full-sample values, the sample values for sub-sample locations are obtained based on the current sub-sample location. Compared to H.264/AVC, the interpolation is more complex with the use of more taps filter.

## 1.3 Video adaptation techniques

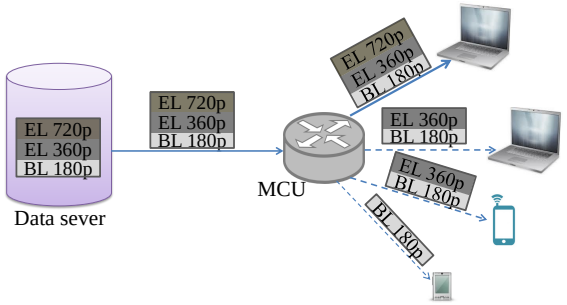
An example of streaming video to a heterogeneous multimedia environment using various adaptation techniques is illustrated in Figure 1.6, and is explained one by one in the next sections.

**Simulcast** This approach includes separate encoding processes of video at different bit rates and possibly at different resolutions, or frame rates. These encoders generate different versions of the same video source [9]. The version which satisfies the constraints of the network or device constraints is transmitted to the user. This technique offers a very low adaptation complexity, since only a simple selection of existing videos is performed. However, this suffers from high complexity because of encoding multiple videos, high storage requirements, and high bandwidth usage in the backbone network for transmitting multiple videos.

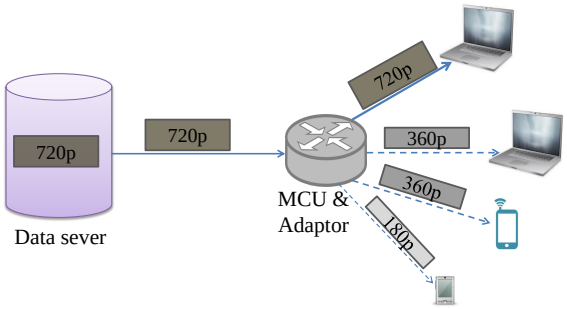
**Scalable video coding** In order to reduce the overhead of simulcast due to independent encoding and transmission of different versions of a video, scalable video




(a) Simulcast streaming solution: Different single-layer versions of the video are independently encoded.




(b) Scalable video coding architecture: A multi-layer representation of the video is generated.




(c) Transcoding approach: A single-layer video is generated.

**Figure 1.6:** Example of delivering video content to various user devices featured by different characteristics. The top laptop constitutes of a high power resolution and a high bandwidth network connection. The other laptop uses wifi connection with lower bandwidth. The PDA has a very low screen resolution and processing power. The mobile phone is integrated with a higher resolution and processing power. The video is transmitted to a multi-control unit through core network. At the multipoint control unit (MCU), the video is adaptively delivered to clients over the access network.

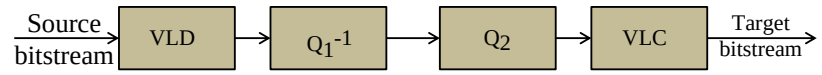
coding has been developed. This effort results in extensions of various video coding standards [10, 11]. A scalable stream is generated to be constituted of various quality video layers that differently chartered in quality, resolutions, or frame rate. Within a scalable stream, the base layer presents the lowest quality level while one or more enhancement layers provide improved qualities. To reduce the overhead of such a scalable video, redundant information between layers is being exploited.

The adaptation of a scalable stream to the capabilities of the receiving device or the network conditions is made by simply discarding certain layers. The resulting bitstream retains conforming to the used video coding standard. Since this adaptation mechanism is deployed without the need of recoding, this is considered as a low complexity technique.

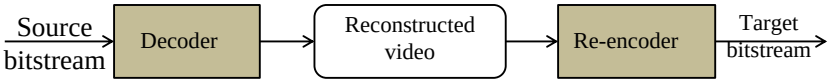
Although a scalable video is capable of providing scalability through a simple operation, such a multi-layer presentation comes along with various drawbacks in some applications. Firstly, the coding efficiency of scalable coding is typically lower than the one of single-layer coding. Secondly, this technique leads to a large increase in encoding and decoding complexity compared to single-layer encoding. Thirdly, the scalability is restricted due to the limited number of layers, and finally, a change at endpoint devices is needed to decode such a multilayer video presentation. So far, these fundamental disadvantages have made scalable coding, particularly the scalable extension of H.264/AVC, less attractive to be adopted in emerging applications; despite many efforts have been made on the development of the scalable extension for video coding standards.

**Transcoding** Alternatively to simulcast and scalable coding, transcoding generates a single-layer video encoded at a high quality. When adaptation is needed, transcoding will be used to change the characteristics, e.g., resolution, frame rate, or quantization parameter, of the video to satisfy the limitations of transmission networks or display devices [12]. Transcoding overcomes the network overhead and the huge amount of computations required for encoding and decoding of the aforementioned adaptation techniques. Moreover, the output video of transcoding is able to exactly match any constraints. In addition to adaptation ability, efficient transcoding may benefit other post-processing applications such as information insertion, watermarking, and interactive video. Due to these advantages, transcoding has been widely used during the last decades. Video research community has put a lot of effort into improving transcoding to reduce the computational complexity while retaining coding efficiency as high as possible [13–21].

Transcoding can be classified into three groups, i.e., bit rate transcoding, temporal transcoding, and spatial transcoding, each of which is applied for a certain reduction of bit rate. Transcoding by modifying the quantization parameter, which is also referred to as bit rate transcoding or transrating, is often used when the required bit rate reduction is less than 50%. When transrating is adopted for a higher




(a) Open-loop approach.




(b) Closed-loop approach.

**Figure 1.7:** High level architecture of a transcoder.

bit rate reduction, the subjective quality of video is significantly decreased with blurred frames. This decline is due to dropping out the residual information since high quantization parameters are applied. To avoid the blurred content, adjusting frame rate (temporal transcoding) or spatial resolution (spatial transcoding) has been recommended for higher bit rate reductions. However, these two techniques result in loss of information since some frames or pixels are discarded. Therefore, they are not suited for small bit rate reductions where transrating provides good performance without any loss of information.

Transcoding can be performed by using either an open-loop or a closed-loop transcoder (Figure 1.7). In an open-loop transcoder (e.g., [14, 20]), the source bitstream is directly translated to the target bitstream by applying variable length decoding, de-quantizing to obtain the DCT coefficients, re-quantizing these coefficients to meet the desired bit rate, remapping the motion vectors, and applying variable length encoding. This approach is a computationally efficient adaptation since it operates directly on the DCT coefficients without the need of fully decoding and re-encoding the entire bitstream. However, such a simple adaptation operation suffers from a significant quality loss due to error propagation caused by mismatch between encoder and decoder reference frames. This quality loss, fortunately, can be reduced by using a cascaded decoder-encoder loop transcoding [13, 15, 18].

In a closed-loop transcoder, the video is decoded. The reconstructed video is subsequently encoded to generate a network/device-friendly stream. The main drawback of a closed-loop transcoder is the huge computational complexity associated with the encoder part, which increases energy consumption of the intermediate network devices. This high complexity needs to be taken into account when developing closed-loop transcoding algorithms.

With the finalization of the high efficiency video coding standard in 2013, it is expected that HEVC will be used in a large scale of emerging applications in the next couple of years. Extensions of this standards, i.e., scalable coding, screen-

content coding, and 3D video coding have been introduced, with scalable coding being capable of providing adaptation property [22]. However, due to the fact that the scalable extension of the former standard H.264/AVC has rarely been used, it is unclear whether the scalable coding will dominate the single-layer coding when HEVC appears in the market. Therefore, examining how video transcoding drawbacks can be overcome is still an attractive research topic for the latest video coding standard.

The remainder of this work will only cover the transcoding process of single-layer video for adaptation of HEVC. Since temporal transcoding can be easily achieved by controlling the prediction architecture between frames in the video, this approach has not been examined in this dissertation. In this work, the author optimizes bit rate transcoding and spatial transcoding of HEVC video, in which transcoding computational complexity is reduced. Motivated by the fact that block partitioning and motion estimation account for a large amount of computational complexity of the HEVC encoder, the author believes these processes can be optimized. The methodology of this work is to utilize the correlation between coding information of the transcoder input and output stream to predict the coding information of the output video. This prediction will allow unnecessary evaluation in the encoder to be discarded, resulting in complexity reduction.

## 1.4 Outline

The optimization of bit rate and spatial transcoding of HEVC video is covered by the three main chapters in this dissertation. As Chapter 2 describes the optimization of a bit rate transcoder in block partitioning level, Chapter 3 describes the optimization in the motion estimation levels. Afterwards, Chapter 4 focuses on improving the performance of spatial transcoding. An overview of each chapter is given as follows:

**Chapter 2:** In this chapter, the complexity of a bit rate transcoder is reduced. The correlation between the blocking partitioning behaviour of the input and output video of a transcoder is analysed. This correlation is then utilized in optimizing the encoding process of the output video via different approaches. A complexity-scalable transcoding is also proposed in this chapter.

**Chapter 3:** The computational complexity of a bit rate transcoder has been optimized at the block partitioning level in Chapter 2. However, motion estimation has not been optimized, even though it accounts for a large portion in evaluating coding mode of a block. Thus, this chapter focuses on reducing the complexity of the motion estimation process. An evaluation on the motion vectors of the input video is conducted first. Then, the motion estimation is optimized by two approaches:

selecting an adaptive initial motion vector by using decision theory, and proposing several fast motion search patterns.

**Chapter 4:** While transrating is often used for a low bit rate reduction of a video stream, spatial transcoding is usually applied for a high-required bit rate reduction or a resolution adaptation. Chapter 4 focuses on reducing the complexity of a spatial transcoding. First, problems on using the state-of-the-art spatial transcoding for an arbitrary resolution reduction of HEVC video are described. Then, a machine-learning-based approach is proposed to address these problems.

Finally, Chapter 5 presents the conclusions of this dissertation.

## 1.5 Publications

The research activity of the author has generated 3 SCI-indexed journals, in all of which the author stands the lead author (in *IEEE Transactions on Multimedia*, *IEEE Transactions on Consumer Electronics*, and *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*). Additionally, the author's work has been presented in 7 international conferences listed in the ISI Proceedings database (all of which the author stands lead). In order to make a concrete story, not all of these publications are adopted in this dissertation.

**Publications in international journals**

1. Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, and Rik Van de Walle. Performance analysis of machine learning for arbitrary downsizing of pre-encoded HEVC video. *IEEE Transactions on Consumer Electronics*. 2015. November 2015.

2. Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Sebastiaan Van Leuven, Jan De Cock, and Rik Van de Walle. Efficient bit rate transcoding for high efficiency video coding. *IEEE Transactions on Multimedia*. 2016. March 2016.

3. Luong Pham Van, Hoyoung Lee, Jaehwan Kim, and Byeungwoo Jeon. A low complexity H.264/AVC deblocking filter with simplified filtering boundary strength decision. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*. 2013. February 2013.

**Publications in international conferences**

1. Luong Pham Van, Jan De Cock, Glenn Van Wallendael, Sebastiaan Van Leuven, Rafael Rodriguez-Sanchez, Jose L. Martinez, Peter Lambert, and Rik Van de Walle. Fast transrating for high efficiency video coding based

on machine learning. *IEEE International Conference on Image Processing (ICIP)*. September 2013.

2. Luong Pham Van, Jan De Cock, Glenn Van Wallendael, Byeungwoo Jeon, and Rik Van de Walle. Adaptive deblocking filtering scheme for intra-coded slices in H.264/AVC. *IEEE International Workshop on Multimedia Signal Processing (MMSP)*. September 2013.

3. Luong Pham Van, Jan De Cock, Antonio J. Diaz-Honrubia, Glenn Van Wallendael, Sebastiaan Van Leuven, and Rik Van de Walle. Fast motion estimation for closed-loop HEVC transrating. *IEEE International Conference on Image Processing (ICIP)*. October 2014.

4. Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, and Rik Van de Walle. Machine learning for arbitrary downsizing of pre-encoded video in HEVC. *IEEE International Conference on Consumer Electronics (ICCE)*. January 2015.

5. Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, and Rik Van de Walle. Out-of-the-Loop information hiding for HEVC video. *IEEE International Conference on Image Processing (ICIP)*. September 2015.

6. Luong Pham Van, Johan De Praeter, Glenn Van Wallendael, Patrice Rondao Alface, and Peter Lambert, Intra-frame sharing for low-complexity decoding of SHVC video. *IEEE International Conference on Consumer Electronics (ICCE)*. January 2016.

7. Luong Pham Van and Byeungwoo Jeon. Complexity reduction of IDCT by using CBP and adaptive pruning in H.264/AVC decoder. In *Proceeding of Computer Graphics and Imaging: Signal Processing, Pattern Recognition and Applications*. February 2013.

# References

[1] Cisco Systems. *Cisco Visual Networking Index: Forecast and Methodology, 20142019*. White Paper, May 2015.

[2] ITU-T. *Video coding for low bitrate communication*. Technical report, ITU-T Rec. H.263, version 1: 1995, version 2: 1998, version 3: 2000.

[3] ITU-T and ISO/IEC JCT 1. *Generic coding of moving pictures and associated audio information - Part 2: Video*. Technical report, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), 1994.

[4] 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, 13(7):560–576, July 2003.

[5] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) Standard*. IEEE Transactions on Circuits and Systems for Video Technology, 22(12):1649–1668, December 2012.

[6] V. Sze and M. Budagavi. *High Throughput CABAC Entropy Coding in HEVC*. IEEE Transactions on Circuits and Systems for Video Technology, 22(12):1778–1791, December 2012.

[7] V. Sze, M. Budagavi, and G. J. Sullivan. *High Efficiency Video Coding (HEVC) - Algorithms and Architectures*. Springer, 2014.

[8] I. Kim, K. McCann, K. Sugimoto, B. Bross, and W. Han. *High Efficiency Video Coding (HEVC) Test Model 7 Encoder Description*. JCTVC-1002, Geneva, CH, May 2012.

[9] B. Li and J. Liu. *Multirate video multicast over the Internet: an overview*. IEEE Network, 17(1):24–29, January 2003.

[10] H. Schwarz, D. Marpe, and T. Wiegand. *Overview of the Scalable Video Coding Extension of the H.264/AVC Standard*. IEEE Transactions on Circuits and Systems for Video Technology, 17(9):1103–1120, September 2007.

[11] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian. *Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard*. IEEE Transactions on Circuits and Systems for Video Technology, 26(1):20–34, January 2016.

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

[13] G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman. *Transcoding of MPEG bitstreams*. Signal Processing: Image Communication, 8(6):481–500, September 1996.

[14] P. A. A. Assuncao and M. Ghanbari. *A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams*. IEEE Transactions on Circuits and Systems for Video Technology, 8(8):953–967, December 1998.

[15] M.-J. Chen, M.-C. Chu, and C.-W. Pan. *Efficient motion-estimation algorithm for reduced frame-rate video transcoder*. IEEE Transactions on Circuits and Systems for Video Technology, 12(4):269–275, April 2002.

[16] J. Xin, C.-W. Lin, and M.-T. Sun. *Digital Video Transcoding*. Proceedings of the IEEE, 93(1):84–97, January 2005.

[17] I. Ahmad, Xiaohui Wei, Yu Sun, and Ya-Qin Zhang. *Video transcoding: an overview of various techniques and research issues*. IEEE Transactions on Multimedia, 7(5):793–804, October 2005.

[18] V. Patil, R. Kumar, and J. Mukherjee. *A Fast Arbitrary Factor Video Resizing Algorithm*. IEEE Transactions on Circuits and Systems for Video Technology, 16(9):1164–1171, September 2006.

[19] H. Shen, X. Sun, and F. Wu. *Fast H.264/MPEG-4 AVC Transcoding Using Power-Spectrum Based Rate-Distortion Optimization*. IEEE Transactions on Circuits and Systems for Video Technology, 18(6):746–755, June 2008.

[20] J. De Cock, S. Notebaert, P. Lambert, and R. Van de Walle. *Requantization transcoding for H.264/AVC video coding*. Signal Processing: Image Communication, 25(4):235–254, April 2010.

[21] V. A. Nguyen and M. N. Do. *Efficient coding unit size selection for HEVC downsizing transcoding*. In IEEE International Symposium on Circuits and Systems (ISCAS), pages 1286–1289, May 2015.

[22] G. J. Sullivan, J. M. Boyce, Y. Chen, J. R. Ohm, C. A. Segall, and A. Vetro. *Standardized Extensions of High Efficiency Video Coding (HEVC)*. IEEE Journal of Selected Topics in Signal Processing, 7(6):1001–1016, December 2013.

# 2

# Fast block partitioning for bit rate transcoding of HEVC video

## 2.1 Rationale and related work

When video content is delivered to users, the network and device constraints lead to the need for an adaptation. In many scenarios, this adaptation can be performed by reducing the bit rate of video streams. A bit rate reduction would be made by changing a property of the video (spatial, temporal resolution, or quantization parameter). Spatial and temporal resolution are often used for a high bit rate reduction (i.e., higher than 50%) while modifying the quantization parameter (bit rate transcoding) is carried out for a smaller bit rate reduction. This chapter focuses on optimizing a bit rate transcoder (or *transrating*). Spatial transcoder is investigated in Chapter 4. Since a temporal resolution modification can be made by designing the temporal referencing architecture when the video is encoded, this solution has not been investigated in the scope of this research.

Transrating allows a video bitstream to (i) adjust to network bandwidth constraints, and/or (ii) meet constraints of the receiver terminal. In the former case, when video is transmitted over networks with fluctuating bandwidth, temporary capacity problems can occur. Having a video bit rate higher than the network bandwidth results in visual artifacts due to packet loss. To reduce such visual distortions, the video stream has to be scaled to a lower rate in a controlled manner at the intermediate devices such as gateways, multipoint control units, or servers. In the use of transrating, bit rate adaptations can follow the constraints of the network in

an optimal way. In the latter case, a video stream might be stored and/or streamed to various devices with different playback capabilities. With such a diversity of devices, it is impossible for a single copy of encoded video to completely match the requirements of all devices. A possible solution is to store several copies of the video on the server and to send the bitstream that best satisfies the requirements of the user. However, this entails storage cost of the server while the pre-encoded video stream may, in one way or another, still mismatch the user requirements. To tackle this problem, the video may be encoded at a high bit rate followed by an online transrating step to meet the requirements of the end-user devices.

Transrating operations can be categorized into either open-loop transcoding or closed-loop cascaded pixel domain transcoding. In the open-loop transcoder, typically only transformed coefficients in the frequency domain are requantized or directly discarded, while the motion parameters as well as other coding information are not re-evaluated. Therefore, a mismatch between encoder and decoder reference frames occurs, which leads to error propagation - known as drift error. Several solutions for bit rate reduction using open-loop transcoding have been investigated for MPEG-1/2 and H.264/AVC bitstreams [1–6]. In one of the earliest works on transrating [2], the variable-length code words corresponding to the quantized DCT coefficients are extracted from the video bitstream. These quantized coefficients are inversely quantized and subsequently requantized to satisfy the new output bit rate. An alternative to requantization, which is called DCT coefficient dropping or dynamic rate shaping, directly cuts high-frequency coefficients from each macroblock [3]. More recently, various bit rate adaptation methods have been proposed for H.264/AVC bitstreams [4–7]. An efficient mixed transrating architecture containing a low complexity scheme combined with a drift cancelling closed-loop scheme was proposed in [5]. A model-based transrating scheme using requantization of the transform coefficients, which is integrated in a rate control mechanism, has been examined in [6]. Despite being a computationally efficient adaptation approach, an open-loop transcoder is subject to drift error resulting in significant losses of visual quality. This drift can be reduced in a closed-loop cascaded pixel domain transcoder where the video bitstream is decoded and re-encoded to match the target bit rate.

Although a closed-loop transcoder for bit rate adaptation can achieve a high rate-distortion performance relative to an open-loop transcoder, the use of this approach is limited for real-time applications due to the high complexity of extensive re-encoding computations. It is important to note that many advanced coding tools (e.g., large coding block sizes with quad-tree coding unit, advanced motion vector prediction) have been integrated into the HEVC standard [8]. This significantly increase the computation complexity of an HEVC transcoder. Therefore, this chapter explores the reduction of computational complexity of a closed-loop bit rate transcoder for HEVC videos.

**Related work**

The complexity of a closed-loop transcoder is essentially caused by huge computations in the encoder part. In order to reduce computational complexity of the transcoder, many approaches have been proposed to optimize the encoding process. Most of these techniques focus on predicting the coding modes in the output video to early terminate encoding. Optimizing the encoder can be by either 1) fast encoding without considering the coding information of the input video, or 2) utilizing the coding information in the input video to accelerate the encoding.

In the first approach, various techniques have been proposed to accelerate the HEVC encoder. These techniques use the texture characteristics of video and/or utilize the temporal/spatial correlation in the video to predict coding information of a CU. In [9], the splitting of a CU in intra-coded frames is decided on the texture homogeneity of the video in the pixel domain and the splitting of its neighboring CUs. A method based on $k$ nearest neighbors has been proposed to determine the CU splitting [10]. In [11], the depth range of a CU is determined to achieve early termination of CU evaluations. On the other hand, early skip mode and merge mode detection methods have been proposed in [12]. Currently, some researchers have started using machine learning to accelerate the HEVC encoder. Support vector machines have been used in [13]. More recently, a decision tree based method has been proposed to reduce the complexity of the HEVC encoder [14]. Although it is reported that the complexity of the HEVC encoder is reduced by the aforementioned techniques, the reduction remains limited. Most of these techniques achieves an encoding time reduction less than 50%. For instances, the method in [9] reduces the encoding time by 47% while the algorithms in [10] and [11] are able to lower the encoding time by 42%.

An alternative approach which can provide a higher complexity reduction for transcoding is to utilize the correlation between coding information of the input and output video. In [15], the statistical properties of the mode distribution are utilized for fast mode refinement of intra prediction. Similarly, in [16] the statistical properties of the mode distribution and motion vector refinement were exploited to reduce the complexity of inter prediction. Given the differences between HEVC and previous video coding standards in their block structure, motion estimation and residual information coding, traditional transrating techniques cannot be directly applied to HEVC. Therefore, new transcoding strategies should be investigated for transcoding of HEVC video streams. Several techniques using machine learning for fast transcoding from MPEG-2 or H.264/AVC to HEVC have been proposed in [17, 18]. Notice that the machine learning based techniques in [17, 18] are used in heterogeneous video transcoding, in which the input and output video are encoded under different standards. More recently, a fast machine learning based transcoding technique has been proposed for transcoding in HEVC [19]. This transcoding technique is used for video composition of multiple HEVC bitstreams.

In [20], an alternative to fast transcoding for HEVC has been proposed by means of a control stream. In other words, the sender encodes a video at different qualities. The high quality encoded version is sent to the user. In case of an adaptation process is required, one of the other streams without residual information will be sent to the encoder to guide the transcoding process. The major downside of this technique is its limited flexibility since the number of encoded versions is fixed. Moreover, this technique results in network overhead for transmitting the extra stream.

In order to upgrade the performance of a closed-loop bit rate transcoding for HEVC video streams, the work presented in this chapter has made the following contributions:

- Firstly, in order to figure out an efficient method for transrating, we propose several techniques which exploit the coding information correlation to speed up transrating. The correlation can be utilized by using machine learning or non-machine learning approaches to predict the CU structure of the output video. Besides that, the spatial and temporal correlation is also used to support this prediction. The complexity of the transcoder can be reduced at the CU and/or PU evaluation level.

- Secondly, we propose a complexity-scalable transrating scheme for particular practical use cases. Different proposed algorithms provide different complexity reductions and bit rate losses. Moreover, the machine learning based algorithm efficiently controls the complexity of the transcoder with two thresholds. The performance of these proposed techniques is evaluated and compared to the state-of-the-art fast HEVC transcoding approaches in terms of complexity reduction and bit rate penalty. Based on this analysis, our proposed methods outperform these algorithms.

- Finally, we propose a novel approach for CU evaluation. This approach recursively merges CUs from smaller CUs to larger CUs. In the traditional evaluation flow, the CUs are evaluated from lower depths to higher depths. In contrast, our proposed method evaluates CU in a reverse way. By using this approach, the optimal splitting behavior of CUs at higher depth is known before evaluating the RD cost at the current depth. This prior-known information may be used for an early termination of CU evaluation. Hence, the complexity of the transcoder is reduced.

The remainder of this chapter is organized as follows: The proposed transrating architecture and methodology are described in Section 2.2. This section provides insights into the correlation between input and output video is elaborated. In Section 2.3, we propose different transcoding techniques which are capable of reducing the transcoding complexity in the CU partitioning level. At the CU

level, CUs can be evaluated in top-to-bottom or bottom-to-top flows, in which the coding information of the input video stream is utilized to reduce the number of evaluations or to early terminate certain evaluations. Furthermore, an adaptive PU selection technique is presented in this section. Evaluations on performance of the proposed techniques are given in Section 2.3.6. Conclusions and original contribution complete this chapter with Section 2.4.

## 2.2 Transrating architecture and methodology

### 2.2.1 Transrating architecture

The typical architecture of a closed-loop bit rate transcoder (also known as a cascaded pixel-domain transcoder (CPDT) is illustrated in Figure 2.1. This architecture consists of two coding loops (i.e., a decoding loop followed by an encoding loop). In the decoding part, the incoming video stream at high quality is fully decoded by performing entropy decoding, inverse quantization ($Q_1^{-1}$), inverse transform ($T^{-1}$), intra prediction ($I_m$) and motion compensation ($M_c$). The decoding process results in the reconstructed video in pixel domain. This reconstructed video is subsequently re-encoded at the target bit rate. In the encoding loop, motion estimation (($M_e$)), motion compensation and intra prediction are carried out to find the optimal coding mode, which results in the prediction residual between the original decoded frames and the predicted frames. This residual is subsequently encoded by performing transform (T), quantization ($Q_2$) and entropy coding to obtain an output video stream with a desirable bit rate. The re-estimation of the residual significantly eliminates drift error. However, this process puts the transcoder under a huge computational complexity. In order to reduce such high complexity of the closed-loop bit rate transcoder, we herein proposed an optimized architecture.

The general concept of the proposed cascaded pixel-domain bit rate transcoder is shown in Figure 2.2. The re-encoding process is optimized by exploiting coding information from the input bit stream. During the decoding part, coding information of the input stream is extracted. This information is utilized by several proposed techniques to limit the number of CUs and PUs, along with motion vectors that are evaluated by the encoder. Note that the complexity of inter prediction is much higher than intra prediction, the focus of the proposed techniques is to optimize inter coding whereas intra prediction stays intact.

### 2.2.2 Methodology

The proposed transrating system focuses on reducing the bit rate of the input video. The relative bit rate reduction is limited to 50%. As recommended in [21], further reductions in bit rate should be achieved by other transcoding methods (e.g., tem-

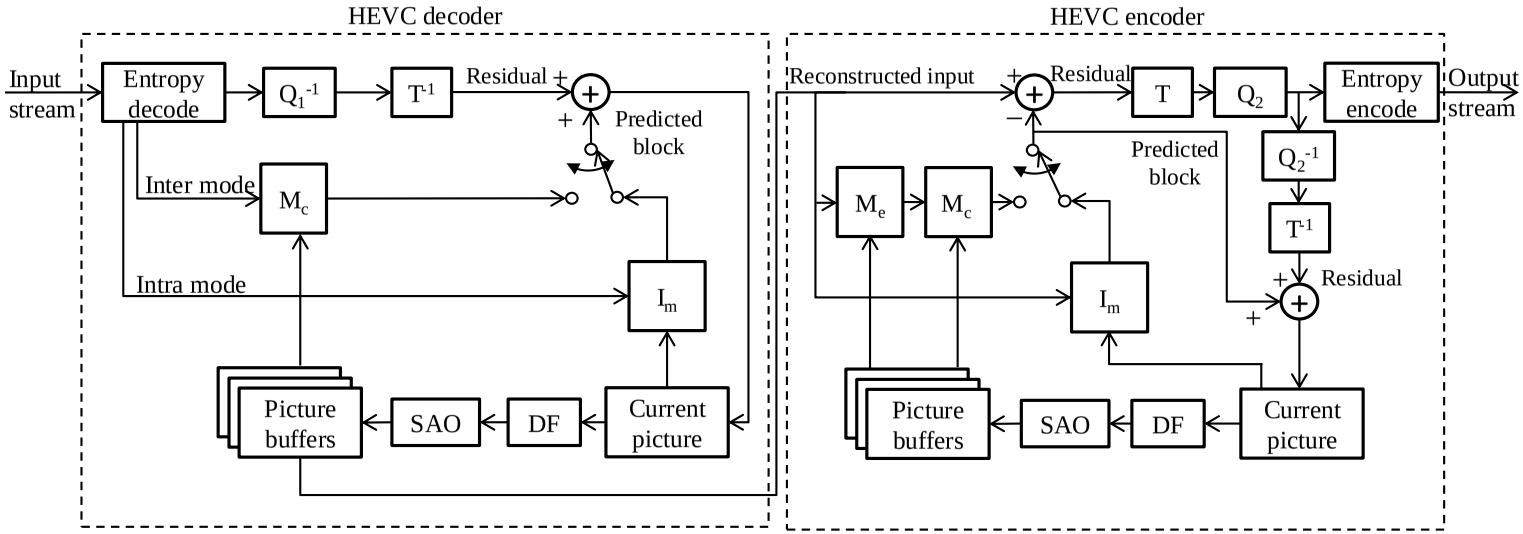



**Figure 2.1:** Cascaded pixel-domain transcoder architecture (CPDT)).

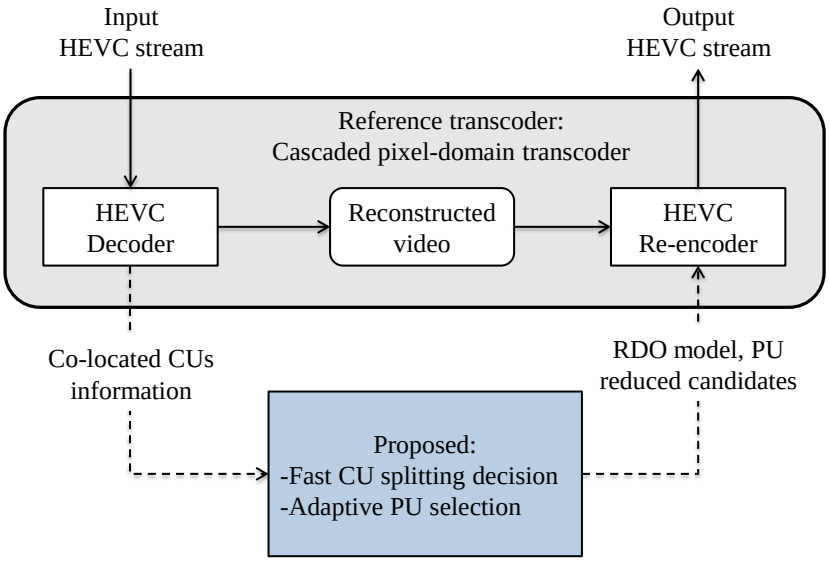



**Figure 2.2:** Proposed pixel domain transcoder.

poral transcoding and/or spatial transcoding). The proposed transrating architecture supports both variable bit rate (VBR) and constant bit rate (CBR) schemes.

In the VBR scenario, input video is encoded under a constant quantization parameter ($QP_1$). The video is re-encoded using the new quantization parameter ($QP_2$) that is higher than the original ($QP_1$). Therefore, a difference $\Delta$QP is introduced as in Equation 2.1.

$$\Delta QP = QP_2 - QP_1 \tag{2.1}$$

With the use of the CBR configuration, the input video is originally encoded at a constant bit rate of $R_i$(kbps). Afterwards, the video is reconstructed and transcoded to a lower bit rate $R_o$(kbps) given by Equation 2.2. In this equation, the rate reduction factor $\alpha$ is limited in the range from 0 to 0.5.

$$R_o = (1 - \alpha) * R_i \tag{2.2}$$

In order to analyze the correlation of bit rate, CU structure, PU partition size, and motion information between the input and output video streams, conditions for evaluation are specified as follows. The HEVC test model (HM) 7 reference software [22] is used for encoding and decoding. Default common test conditions as defined in [23] are used with the low delay P main configuration (LP) and the VBR scheme. Both input and output video streams are encoded under a coding

**Table 2.1:** Bit rate reduction by changing QP

| Sequence | [kbps] ($QP_1$) | Bit rate reduction[%] | | |
|---|---|---|---|---|
| | | $\Delta$QP=2 | $\Delta$QP=4 | $\Delta$QP=6 |
| ParkScene 1080p | 8365 (22) | -35.16 | -54.92 | -68.24 |
| | 3264 (27) | -32.96 | -52.37 | -65.65 |
| | 1380 (32) | -33.90 | -51.63 | -64.84 |
| | 604 (37) | -34.50 | -52.30 | -64.62 |
| BasketballPass WQVGA (240p) | 1788 (22) | -27.11 | -44.55 | -57.63 |
| | 899 (27) | -27.93 | -44.90 | -57.19 |
| | 453 (32) | -27.29 | -42.30 | -54.14 |
| | 243 (37) | -25.81 | -40.20 | -50.28 |
| BlowingBubbles WQVGA (240p) | 2149 (22) | -30.92 | -51.25 | -65.25 |
| | 880 (27) | -28.71 | -48.87 | -63.04 |
| | 386 (32) | -29.44 | -47.91 | -61.04 |
| | 179 (37) | -28.68 | -45.84 | -56.59 |
| FourPeople 720p | 2465 (22) | -38.46 | -56.51 | -67.87 |
| | 935 (27) | -27.75 | -44.34 | -56.54 |
| | 469 (32) | -25.15 | -39.80 | -51.60 |
| | 260 (37) | -23.93 | -37.93 | -48.84 |
| **Average** | | **-29.86** | **-47.23** | **-59.59** |

structure of IPPP... with 4 reference frames. The quantization parameter is chosen such that $QP_1 \in \{22, 27, 32, 37\}$ while $\Delta$QP$\in \{2, 4, 6\}$.

The correlation of CU structure, PU partition size and motion vectors is statically obtained by comparing these information of an output block and that of its collocated blocks in the input video. The collocated blocks of an output block refers to the blocks that cover the area in the input video with the same size and index of the output block.

### 2.2.3 The impact of $\Delta$QP on the bit rate reduction

The impact of different values of $\Delta$QP on the bit rate of video stream is obtained in order to determine the possible $\Delta$QP for a transrater. The results shown in Table 2.1 have showcased the following conclusion.

Firstly, the relative bit rate reduction increases when $\Delta$QP values get high. On average, these reductions are about 30%, 47%, and 60% for $\Delta$QP of 2, 4, and 6 respectively. In the proposed transrating scheme, we limit the $\Delta$QP to 6, resulting in a bit rate reduction by a factor of 2 to 3.

**Table 2.2:** Probability of CU depth in the output video stream given the CU depth of the input video stream ($P\{d_o|d_i\}$[%]). Dark color denotes high probability

| $d_i$ | ΔQP = 2 | | | | ΔQP = 6 | | | |
|---|---|---|---|---|---|---|---|---|
| | $d_o$=0 | $d_o$=1 | $d_o$=2 | $d_o$=3 | $d_o$=0 | $d_o$=1 | $d_o$=2 | $d_o$=3 |
| 0 | 97 | 3 | 0 | 0 | 98 | 2 | 0 | 0 |
| 1 | 20 | 76 | 4 | 0 | 43 | 53 | 3 | 0 |
| 2 | 4 | 20 | 71 | 5 | 16 | 35 | 45 | 4 |
| 3 | 1 | 7 | 26 | 65 | 6 | 21 | 36 | 36 |

## 2.3 Fast block partitioning decision

Computational complexity of the HEVC encoder is mainly induced by recursively splitting a coding tree unit (CTU) into coding units (CUs). This splitting process is performed for CUs from depth 0 ($64\times64$ pixel CU) to depth 3 ($8\times8$ pixel CU). Each CU is the root for further evaluation of the prediction unit (PU), and transform units. Depending on the mode, for each 2Nx2N block, eight PU sizes can be chosen (four symmetric partitions: $2N\times2N$, $N\times2N$, $2N\times N$, and $N\times N$) and four asymmetric partitions: $nL\times2N$, $nR\times2N$, $2N\times nU$, and $2N\times nD$). To obtain the most efficient mode for a CU at depth d, all PU partitions and all Residual Quad-Tree (RQT) configurations are evaluated during the rate-distortion optimization (RDO) process.

Given the high computational complexity of the splitting process, in this chapter, we propose several techniques to reduce the number of candidates being evaluated and to early terminate the splitting process. In the following subsection, the correlation between the coding block sizes (represented by the CU depth) of the input and output video stream in a cascaded decoder-encoder is evaluated. Section 2.3.2 that follows exploits this analysis as a starting point for the proposed fast CU splitting techniques. The correlation between input and output PU sizes is then analyzed in Section 2.3.3; whilst Section 2.3.4 proposes further accelerations upon adaptive PU evaluation using this PU correlation. Finally, the prediction performance of the proposed methods is estimated in Section 2.3.5.

### 2.3.1 Correlation between coding block depths

The correlation between the coding block sizes is derived by observing the transrating process of five video sequences (i.e., *ParkScene*, *BasketballDrill*, *BQMall*, *BQSquare*, and *FourPeople*). The information of the coding depth of CUs in the output stream and collocated CUs in the input stream is extracted. The conditional probability ($P\{d_o|d_i\}$) that a CU is re-encoded using depth $d_o$, while originally encoded at depth $d_i$, is then derived.

In order to obtain the conditional probability, a frame is divided into 8x8 (smallest CU size) blocks. The depth of such an $8\times8$ block is defined as that of the CU covering this block. $P\{d_o|d_i\}$ is the probability of the output depth of an $8\times8$ block given the input depth of this $8\times8$ block. The result is shown in Table 2.2.

Experimental result shows that when the $\Delta$QP increases for a given $d_i$, the CU is typically re-encoded at this depth or lower, corresponding to a larger partition. For instance, when a CU in the input bit stream is encoded using depth 1, there is a 76% probability that the CU is re-encoded using the same depth and there is a 20% probability that the CU is merged to depth 0, in case where $\Delta$QP = 2. This observation will be exploited in the rest of this paper to reduce the complexity of transrating. A high value of $\Delta$QP denotes a significant probability where input CUs are merged into CUs at lower depths. On the other hand, a large difference in QP represents a low correlation between coding information of the input and output stream.

### 2.3.2 Fast CU splitting decision

In this section, four fast transrating techniques are proposed to reduce the complexity of the CU (RDO) evaluation. These methods are categorized into top to bottom (T2B) and bottom to top (B2T). The CU evaluation of the T2B approach is performed by a recursive splitting process, where CUs are split from lower depths to higher depths. The trivial T2B applies the CU structure from the input video stream to the output video. An improved T2B approach is a machine learning based method ($T2B_{ML}$) that exploits the correlation of coding information from the collocated CUs to build split-flag decision tree models. In the B2T category, CUs are merged from smaller sizes to bigger sizes. A first bottom-to-top method is B2T in which the CU structure from the input video serves as the initial structure of the CU in the output video. B2T then merges smaller CUs into a larger CU. To further reduce the complexity of the B2T method, $B2T_{TLP}$ is proposed. $B2T_{TLP}$ considers the splitting behavior of the top and the left CUs of the current frame, as well as of the collocated CU in the previously encoded frame.

#### 2.3.2.1 Top to bottom (T2B) CU decision

Motivated by the observation in Table 2.2 in which the CUs in the output video typically have an equal or lower depth compared to the collocated CUs in the input video, top to bottom (T2B) method uses CTU structure of the input stream to determine the maximum depth it should evaluate for each CU in the output stream.

The RD cost of a CU is obtained by 2.3 where $D$, $\lambda$, and $R$ respectively indicate the distortion, Lagrange constant, and the number of bit to signal coding

information to the decoder [24].

$$RD = D + \lambda.R \tag{2.3}$$

T2B technique evaluates the RD cost for CUs from depth 0 to the maximum depth of the initial CTU (iCTU), which is the structure of the collocated CTU in the input video stream. At each depth of iCTU, the RD cost for every CU of iCTU is examined. After obtaining the RD cost $RD_{notsplit}$ of a CU at a depth $d$, the decision to check higher depths is based on the input split flag.

- If input CU is split, the split of the output CU is also performed for further evaluation. This evaluation results in the $RD_{split}$ which is the summation of the RD costs of 4 children CUs at depth $d+1$. If $RD_{notsplit}$ is smaller than $RD_{split}$, the CU remains at size $2N \times 2N$. Otherwise, it is split into smaller CUs.
- When the split-flag of the input CU is 0, further splitting of this CU stopped and the output CU is decided not to be split.

It should be noted that the evaluation of RD cost of a CU in iCTU is always active even when the input collocated CU is split. This RD cost evaluation is performed since there is a notable probability of an input CU being re-encoded using a lower depth in the output stream. Due to this extra evaluation, the complexity reduction of this approach is limited compared to other proposed approaches. The RDO process of T2B is described in Algorithm 1.

#### 2.3.2.2 Machine learning based T2B ($T2B_{ML}$) CU decision

Obviously, T2B method only exploits the splitting behaviour of CUs in the input stream to control the RD evaluation process of CUs in the output stream. In other words, other coding information of the input stream was not utilized. $T2B_{ML}$ is proposed to improve the T2B technique. With the use of the machine learning approach, $T2B_{ML}$ deploys more coding information of the incoming stream to accelerate the transcoder. First, the list of selected features and the data set for training are presented. After that, the training process is elaborated on. Then, the CU splitting process based on the decision of machine learning is described. Finally, the RD cost evaluation is given.

**A. Training data set and features**

In the proposed techniques, an off-line training mechanism has been used. Specifically, decision trees are established on a data set of training sequences. These trees consequently are used for classification when transrating any other sequences. The selection of training sequence should make the decision trees general enough and

**Table 2.3:** Decoded information as input to decision trees.

| Parameters | Domain | Meaning |
|---|---|---|
| split_flag | 0, 1 | Split-flag of CU in $QP_1$ |
| delta_depth | 0-3 | Difference between the current depth and the max depth of CUs |
| sum_depth | Number | Sum of depths of CUs |
| num_pu | Number | Number of PUs |
| cbf | 0, 1 | 0: If none of the CUs (luminance component) are encoded<br>1: Otherwise |

free from overtraining. Herein, four sequences with an activity ranging from low to high have been selected. These can be categorized into: low motion and low complexity (*FourPeople*), medium activity (*ParkScene*, and *BQSquare*) and high motion (*BasketballDrill*). To ensure unbiased conclusions, these training sequences are not going to be used for evaluating the performance of the proposed methods.

Features are defined upon the coding information of the input video. Five features as listed in Table 2.3 have been used. Given the high correlation between coding depths of the input and output streams, the coding depth information of the input video including split_flag, delta_depth and sum_depth have been taken into account. Furthermore, the splitting of a CU in the PU level has been used. Finally, coded block flag - which represents the texture information of the video, has also been engaged. Originally, the motion vector and residual information in the input video (the variance of the input motion vectors, the means and variances of the residual) have been taken into consideration during the construction of these decision trees. As appearing in end-nodes of these trees, these features are of minor importance. That aside, these features complicate the trees; thus, they were not included in the final design to simplify these trees.

**B. Training mechanism**

Online training is considered as a content-adaptive approach. In the online training mechanism, several first frames of the video are transcoded without the use of the non-optimized encoder. The coding information of these frames is used to form the training set. At the end of the training phase, the coding information is gathered to build the decision trees. The transcoding process of the remaining frames is accelerated with the use of these decision trees.

Online training may provide a higher classification performance compared to offline training. However, there exists a need of a fully decoding and encoding loop without optimization of the encoding for a certain number of frames to generate training data, requiring another re-training process. Therefore, this approach

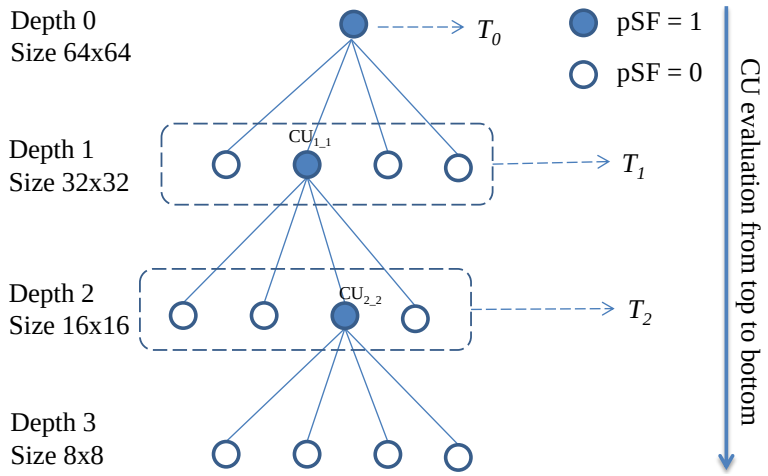



**Figure 2.3:** Splitting of a CTU using decision trees (pSF = Predicted Split Flag).

introduces a high computational complexity that challenges the integration into real-time applications.

Unlike the previously mentioned approach, off-line training helps avoid the re-training process. The decision trees are only generated once in use of the training data set. These decision tree models are used when transcoding other video sequences. Thanks to the low complexity, the off-line training mechanism is used in the proposed transrating technique.

For each $\Delta$QP three decision trees indicated as $T_0, T_1$, and $T_2$ were constructed. These decision trees predict whether splitting the CUs is needed at depth 0, depth 1, and depth 2, respectively. An example of using these decision trees in splitting of a CTU is illustrated in Figure 2.3.

The decision trees have been made with machine learning using decoded information (listed in Table 2.3) from four training sequences. This information is used as the input of the *WEKA* data-mining tool [25]. The tool used for generating decision trees is *J48*, an implementation of the *C4.5* [26] algorithm in *WEKA*. This *C4.5* algorithm is a well-known algorithm in the literature for building decision trees and has widely been used in classification applications. The input of *WEKA* algorithms are datasets representing flat files, known as *ARFF* (Attribute-Relation File Format) files [25]. These files consist of columns representing features and rows samples. An *ARFF* file has two differentiated sections: the header section and the raw data section. The header section embodies the attribute declaration, i.e., the name and possible values to each feature. Figure 2.4 shows the declaration and a part of the data section in an *ARFF* file used in our transrater. It should be noted that output_split_flag in Figure 2.4 is the split flags of the output CUs which

```
@relation Split_flag_Model-dQP6-T1
-----------------------------------------------
@attribute   split_flag_qp1   {0, 1}
@attribute   sum_depth   numeric
@attribute   delta_depth   {0, 1, 2, 3}
@attribute   cbf   {0, 1}
@attribute   num_pu   numeric
-----------------------------------------------
@attribute   output_split_flag {0, 1}
-----------------------------------------------
1, 48, 2, 1, 20, 1
1, 28, 2, 1, 13, 1
1, 18, 2, 1, 9, 1
1, 8, 1, 1, 6, 1
0, 1, 0, 0, 1, 0
        .
        .
        .
```

**Figure 2.4:** *ARFF* file format example.

is obtained by a full decoding/full encoding loop.

An example of decision tree, which is used to predict the split-flag of output CUs at depth 1 with a QP difference of 6, is shown in Figure 2.5. As can be seen from this figure, the split-flag of input CUs appears at the top of the tree. This implies that the partitioning of the input and output CUs are most correlative among the selected coding features.

**C. CU splitting process using the machine learning based classification**

Coding information of collocated CUs from the input bit stream, as listed in Table 2.3, is extracted during transrating. This information is used as the input for machine learning models (decision trees). Out of the decision tree, a split-flag and the corresponding confidence ratio P are given. P is defined by Equation 2.4, wherein $N_s$ and $N_f$ are the numbers of successfully predicted samples and misclassification, respectively.

$$P(\%) = \frac{N_s}{N_s + N_f} \tag{2.4}$$

The RDO process for the CU can be controlled upon these results. An example of splitting a CU from depth 0 to depth 3 based on decision trees is shown in Figure 2.3. The split-flag of the CU at depth 0 ($CU_0$) is predicted by $T_0$. The input for this tree is the decoded information from the CUs within a collocated area of 64x64 pixels of this $CU_0$. The output of this tree is a predicted split-flag (0 or

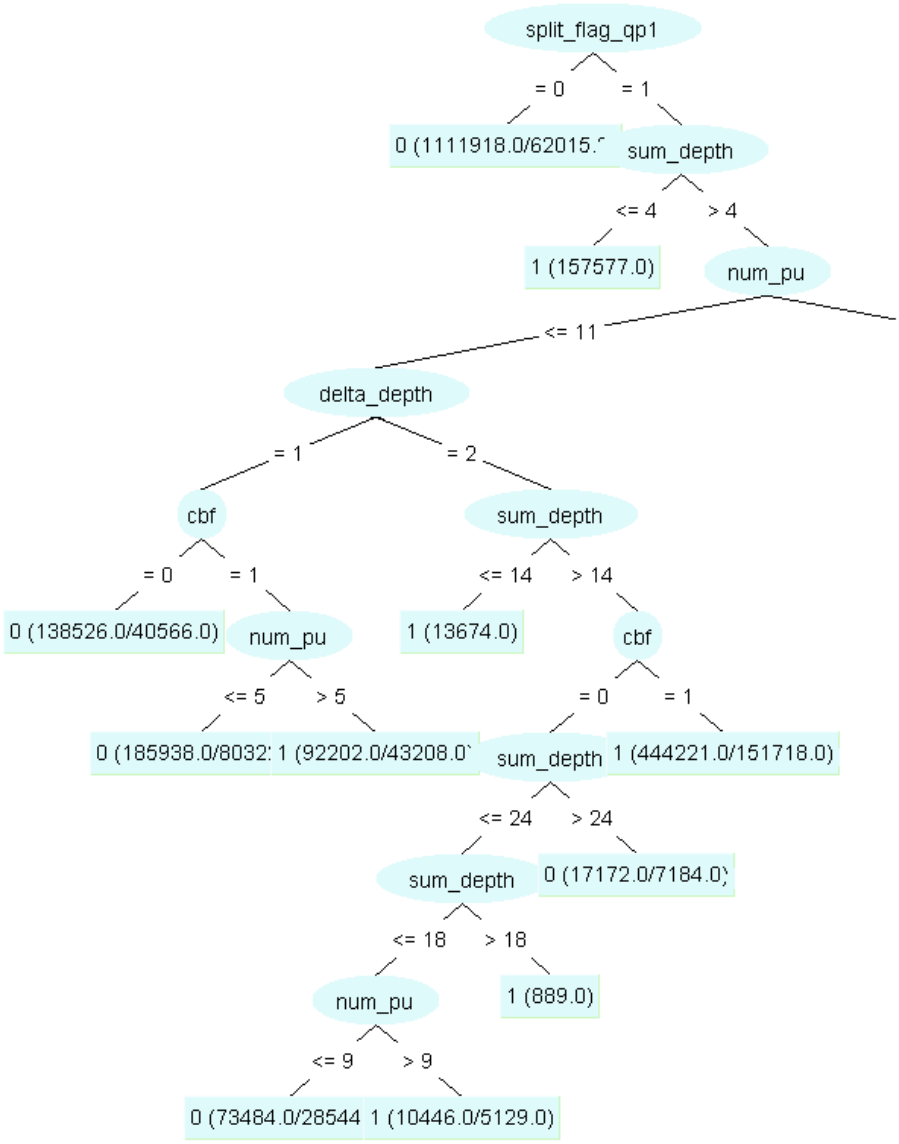



**Figure 2.5:** A part of the decision tree for CUs at depth 1 with a $\Delta$QP of 6.

1), and the probability of the prediction. In this example, a decision that $CU_0$ is split into four CUs has been made. The splitting of the sub-CUs is determined by tree $T_1$ using the coded information from CUs at the collocated $32\times32$ pixels. At depth 1, only $CU_{1\text{-}1}$ is to split further while the others are not. The prediction of the split-flags of the four sub-CUs at depth 2 arising from $CU_{1\text{-}1}$ is performed by

**Table 2.4:** RD evaluation for a CU at depth $d$ (Yes (Y) / No (N))

| Predicted split-flag | $P$ | Check at depth $d$ | Check at depth $d+1$ | End recursion |
|---|---|---|---|---|
| 0 | High | Y | N | Y |
| | Medium | Y | Y | Y |
| | Low | Y | - | N |
| 1 | High | N | - | N |
| | Medium | Y | - | N |
| | Low | Y | Y | Y |

$T_2$. This tree predicts a split-flag of 1 for $CU_{2\_2}$ and 0 for the other CUs.

**D. RDO evaluation for the transrating process**

Based on the decoded syntax information, the decision tree is obtained by offline training results in a predicted split-flag. Probability $P$ of the prediction is used herein to steer the RDO process. The probability is classified as high, medium, or low by comparing it with two proposed thresholds $Thr_1$ and $Thr_2$, in which $Thr_2 > Thr_1$. A summary of classification is given in Equation 2.5.

$$P \text{ is } \begin{cases} High & \text{If } P >= Thr_2 \\ Low & \text{If } P < Thr_1 \\ Medium & \text{Otherwise} \end{cases} \tag{2.5}$$

The RDO evaluation to further refine the predicted split-flag is as follows: A high value of $P$ implies high confidence in the prediction. Therefore, the predicted split-flag is directly used as the optimal splitting behavior for the CU. A low $P$ denotes low level of accuracy of the prediction, leading to the re-evaluating of the predicted split-flag. By adjusting $Thr_1$ and $Thr_2$, the transcoding complexity can be controlled, thus a trade-off between complexity and coding performance can be achieved. The details of the proposed RDO process are depicted in Table 2.4 and the overall algorithm is summarized in Figure 2.6. In this table, 'end recursion' indicates whether the recursive splitting process is terminated after the RD evaluation at depth $d$ and/or $d+1$. When 'end recursion' is 'N', the split flag is predicted again and the RD evaluation process is recursively implemented at depth $d+1$. Otherwise, the RD evaluation process is terminated.

For instance, if the predicted split-flag is 0 and $P$ is high, the CU should not be split and the RD cost evaluation is only performed at the current depth d. When $P$ is medium, the RD cost is evaluated at both depth $d$ and depth $d+1$. However, the CU at depth $d+1$ should not be split, and the recursion is terminated. When

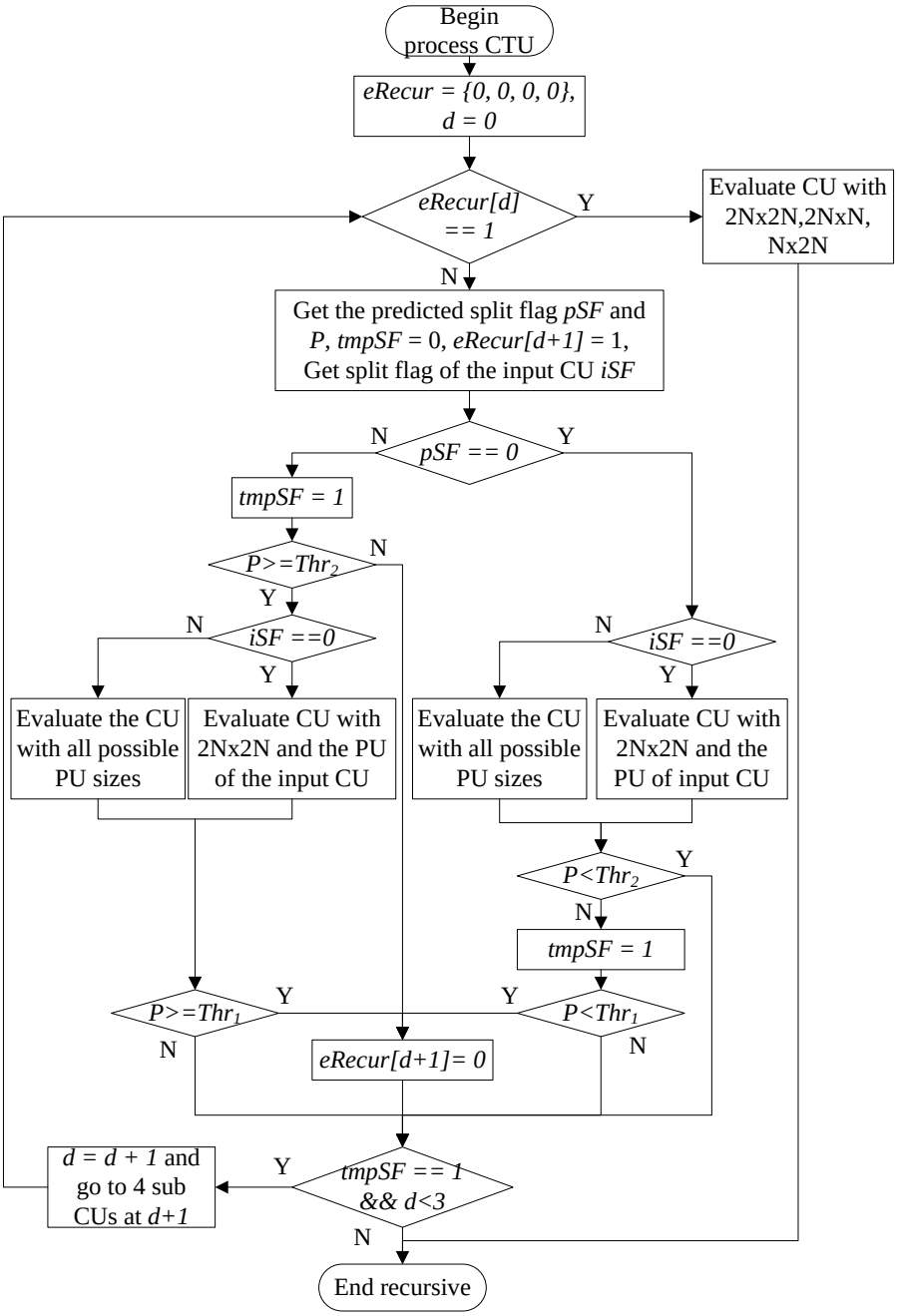



**Figure 2.6:** The flowchart of $T2B_{ML\_PU}$ algorithm. The 'end Recursion' (eRecur) is signalled from depth $d$ to depth $d+1$. The recursive process is terminated until depth d = 3 or $eRecur = 1$.

$P$ is low, the RD cost calculation is performed at depth $d$ and depth $d+1$ without splitting further.

For a predicted split-flag of 1, if $P$ is high, the CU is immediately split to depth $d+1$. Else, RD cost is evaluated at depth $d$ when $P$ is either medium or low. When $P$ is low, the CU is evaluated at depth $d+1$ and the split recursion is terminated; When $P$ is medium, at depth $d+1$, the split-flag prediction is performed and the RD evaluation process is controlled on the new predicted split-flag.

#### 2.3.2.3 Bottom-to-top (B2T) CU decision

In the top-to-bottom (T2B) approach, the evaluation is performed for CUs from lower depths to higher depths. The RD cost of the CU is normally obtained at lower depths first before a derivational RD cost of CUs at higher depths. The splitting of a CU at a lower depth is only decided when its RD cost and those of all CUs at higher depths are obtained. As a result, the RD costs at lower depths are always calculated even if the optimal mode of the CU is split. B2T method is proposed to address this problem as it evaluates CUs from higher depths to lower ones. The optimal splitting behavior of CUs at higher depths is utilized to decide whether the RD cost of CUs at lower depth is obtained or whether the evaluation is terminated.

An example of splitting a CTU under B2T is given in Figure 2.7. At depth 3, RD costs for only 8 CUs are obtained, whereas an unmodified encoder might evaluate the costs of 64 CUs. At depth 2, B2T evaluates RD costs for 8 CUs. After these RD costs are calculated, RD costs of $CU_{2\_1}$ and $CU_{2\_7}$ are compared to the sum of the RD costs of their previously evaluated sub-CUs at depth 3. Assuming that the child nodes of $CU_{2\_1}$ are merged while the best decision for $CU_{2\_7}$ is to split. Then, at depth 1, the RD cost of $CU_{1\_4}$ is not evaluated since the decision has already been made to split $CU_{2\_7}$. Only the RD cost of $CU_{1\_1}$ is evaluated since all of 4 corresponding sub-CUs are not split. The two shaded CUs at this depth are also evaluated. Finally, the RD calculation for the root CU at depth 0 is also skipped since not all sub-CUs can be merged.

The summary of the proposed B2T method for transcoding are described as follow. The collocated CU structure in the input video (iCTU) serves as a starting point for the CU structure in the output video. After all, as seen in Section 2.3.1, the depth of CUs after transrating is usually lower or equal to the depth of the preceding CUs. Apart from the CU evaluation flow of HEVC, the RDO process of B2T is recursively performed by merging sub-CUs from the initial depth $d_{max}$ of iCTU to depth 0. The CUs at the maximum depth ($d_{max}$) are always evaluated, with the merging process performed under particular conditions.

- If all 4 sub-CUs at depth $d+1$ are not split, it might be more optimal to use a larger CU size. Therefore, the RD cost for this CU at depth $d$ is obtained.

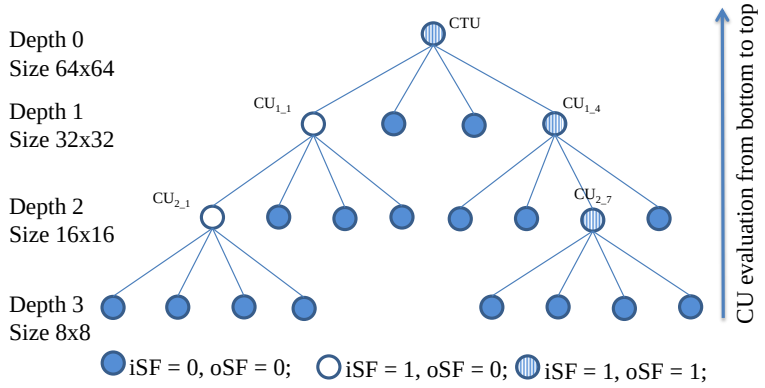



**Figure 2.7:** An example of splitting a CTU based on B2T. iSF and oSF are the split flags of the input and output CUs, respectively.

- Otherwise, the CU is not re-evaluated and considered as split.

Algorithm 2 describes the B2T approach. In this algorithm, $nd$ (*next depth*) represents the depth of the four child CUs of a CU at depth $d$. $SF_{nd\_i}$ and $RD_{nd\_i}$ are respectively the optimal split flag and RD cost of the $i^{th}$ CU of the four child CUs at depth $nd$.

#### 2.3.2.4 B2T CU decision based on top, left CUs, and collocated CU in previous frame

This method ($B2T_{TLP}$) works on a similar basis as to B2T. However, it considers not only the CU structure from the incoming bit stream but also the splitting behavior of neighboring CUs (namely top and left CUs) and the collocated CU in the previously encoded frame. The RD cost of a CU at a lower depth is obtained if all CUs at higher depths can be merged and the top, left, and collocated CU are not split. Otherwise, the RD cost for this CU is not evaluated.

### 2.3.3 Correlation between prediction unit sizes in input and output streams

The complexity of selecting the optimal CU size is reduced by the fast CU splitting techniques proposed in the preceding sections, which exploit tanahe correlation between CU structures of collocated CUs. However, the correlation between PU sizes in the input and output video bit stream has not been employed. This section will analyze the correlation before proposing an adaptive PU selection method that utilizes the PU sizes correlation to reduce the complexity of the optimal PU partition selection (which is a sub-process of CU selection). The number of evaluated

**Algorithm 1** Pseudo-code for T2B CU evaluation algorithm

**Input:** initial CTU $iCTU$ = the CTU structure in the input video stream, $d_{max}$ = maximum depth of $iCTU$
**for** $d$ = 0 to $d_{max}$ **do**
  **for all** $CU_d \in$ CUs at depth $d$ of $iCTU$ **do**
    $RD_{notsplit} = \infty$, $RD_d = \infty$, $SF_d$ = 0
    $RD_{notsplit} \leftarrow GetRD(CU_d)$, $RD_{split} = \infty$
    **if** $CU_d$ is split in $iCTU$ **then**
      Go to 4 CUs at depth $d + 1$
      $RD_{split} \leftarrow \sum_{i=0}^{i=3} RD_{(d+1)i}$
    **end if**
    $SF_d \leftarrow (RD_{notsplit} < RD_{split})?0 : 1$
    $RD_d \leftarrow (SF_d = 0)?RD_{notsplit} : RD_{split}$
  **end for**
**end for**
Process the next CTU

**Algorithm 2** Pseudo-code for B2T CU evaluation algorithm

**Input:** initial CTU $iCTU$ = the CTU structure in the input video stream, $d_{max}$ = maximum depth of $iCTU$
**for** $d$ = $d_{max}$ to 0 **do**
  **for all** $CU_d \in$ CUs at depth $d$ of $iCTU$ **do**
    $RD_{notsplit} = \infty$, $RD_d = \infty$, $SF_d$ = 0
    $RD_{split} = \infty$
    **if** $CU_d$ is not split in $iCTU$ **then**
      $RD_{notsplit} \leftarrow GetRD(CU_d)$
    **else**
      $nd \leftarrow d + 1$
      $Ck_d \leftarrow (SF_{nd\_0}||SF_{nd\_1}||SF_{nd\_2}||SF_{nd\_3})$
      **if** $Ck_d = 0$ **then**
        $RD_{notsplit} \leftarrow GetRD(CU_d)$
        $RD_{split} \leftarrow \sum_{i=0}^{i=3} RD_{nd\_i}$
      **end if**
    **end if**
    $SF_d \leftarrow (RD_{notsplit} < RD_{split})?0 : 1$
    $RD_d \leftarrow (SF_d = 0)?RD_{notsplit} : RD_{split}$
  **end for**
**end for**
Process the next CTU

**Table 2.5:** Probability of PU partitioning mode of the output video stream given the PU partitioning of the input video stream

| $PU_i$ | $P\{PU_i\}[\%]$ | $P\{PU_o \mid PU_i\}[\%]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2Nx2N | 2NxN | Nx2N | NxN | 2NxnU | 2NxnD | nLx2N | nRx2N |
| 2Nx2N | 78.67 | **92.47** | 2.01 | 2.65 | 0.21 | 0.54 | 0.61 | 0.71 | 0.79 |
| 2NxN | 5.57 | **59.40** | **30.54** | 4.38 | 0.20 | 2.34 | 1.26 | 0.99 | 0.90 |
| Nx2N | 7.06 | **54.97** | 2.99 | **36.26** | 0.19 | 0.70 | 0.67 | 2.68 | 1.53 |
| NxN | 1.26 | **45.92** | 2.91 | 4.00 | **44.32** | 0.86 | 0.40 | 1.09 | 0.50 |
| 2NxnU | 1.78 | **61.45** | 7.48 | 3.62 | 0.10 | **23.93** | 1.14 | 1.14 | 1.14 |
| 2NxnD | 1.63 | **61.94** | 5.48 | 3.55 | 0.06 | 1.23 | **25.62** | 1.10 | 1.03 |
| nLx2N | 2.27 | **57.83** | 2.38 | 9.14 | 0.07 | 0.73 | 0.77 | **27.78** | 1.29 |
| nRx2N | 2.07 | **61.27** | 2.37 | 7.39 | 0.05 | 0.74 | 0.75 | 1.58 | **25.84** |

PU partitions will diminish by referring to the PU partition in the input bit stream. Table 2.5 shows the correlation between the PU size of collocated CUs in the input and output bit stream. $\text{PU}_o$ and $\text{PU}_i$ indicate the PU size of an output CU and that of the collocated CU in the input video, respectively. Five sequences indicated in Section 2.3.1 are analyzed. $P\{\text{PU}_o|\text{PU}_i\}$ is the conditional probability that a CU is encoded using $\text{PU}_o$ given $\text{PU}_i$ when both CUs in the input and output bit streams have the same depth. In analysing Table 2.5, for any given $\text{PU}_i$, there is a significant probability that PUo has either the same partitioning or a 2Nx2N partitioning.

### 2.3.4 Predictive PU Selection

The proposed adaptive PU selection only evaluates PU partitions with a high probability given the input prediction unit ($\text{PU}_i$) size. For three methods (T2B, B2T, and $\text{B2T}_{\text{TLP}}$), if the evaluated CU size is equal to the input CU size, only PU sizes 2Nx2N and $\text{PU}_i$ are evaluated. Consequently, the number of PU candidates reduces from 8 down to 2. Otherwise, if the input CU is split, all possible PU sizes are evaluated for the output CU. The selection algorithm for these three approaches is summarized in Algorithm 3.

The $\text{T2B}_{\text{ML}}$ method makes use of the predicted split-flag and the split-flag of the CU in the input to derive PU size candidates. These candidates of CUs at depth $d$ and sub-CUs at depth $d+1$ are jointly controlled by following the RDO model proposed in Table 2.4. At depth $d$, if the predicted split flag ($pSF$) is 0 or $pSF$ = 1 with a low or medium accuracy, the PU selection is controlled by the splitting of the input CU. If the input CU is not split, the output CU is evaluated using the input PU ($iPU$) and $2N\times2N$. Else, the CU is evaluated with all possible PU sizes. At depth $d+1$, if the recursion is decided as to stop, only the three largest PU sizes including $2N\times2N, 2N\times N$, and $N\times2N$ are evaluated. Else, the split flag is newly predicted and the PU selection process is recursively performed for depths $d+1$ and $d+2$. The overall flow chart of $\text{T2B}_{\text{ML\_PU}}$ is given in Figure 2.6.

**Algorithm 3** Pseudo-code for predictive PU evaluation of a CU algorithm using T2B, B2T, and $\text{B2T}_{\text{TLP}}$

```
Input: Split flag iSF of the collocated CU in the input video
if iSF = 0 then
    Get PU iPU of the collocated CU in the input video
    Evaluate the output CU with iPU and 2Nx2N
else
    Evaluate the output CU with all possible PU sizes
end if
Process the next CU
```

### 2.3.5 Prediction accuracy of the proposed techniques

The prediction performance evaluation of the proposed methods has been made for the use of VBR as indicated in Section 2.2 with a $\Delta$QP at 6. The prediction accuracy is measured at three CU depth levels. At each CU depth, the accuracy of prediction is given by the probability where the split-flags predicted by the proposed methods and an anchor transcoder are equivalent. In the anchor transcoder, the CU size is decided by the regular HM reference software. At the PU level, the PU size matching rate is the probability where the PU sizes of the proposed methods and the anchor transcoder are the same, given that the CU sizes of the proposed and anchor method are the same.

The results are shown in Table 2.6 where the trivial method copies the input coding structure to the output video stream. As can be seen in Table 2.6, the proposed methods obtain high prediction accuracy. There is a remarkable improvement in accuracy of about 20% of our proposed methods compared to the trivial transcoder. Among the proposed methods, T2B provides best prediction performance with 83.60% and 88.66% for LP and RA configurations, respectively. The prediction accuracy of T2B is high since the CU is always evaluated whether the input CU is split or not. At the PU evaluation level, the proposed adaptive PU size selection achieves a matching rate of 90% - an approximation of 7% higher than the trivial method.

Table 2.6 demonstrates that the off-line trained model used in $\text{T2B}_{\text{ML}}$ is adequately generalized. Training sequences and test sequences share similar prediction accuracies. The advantage of using the off-line training model is that there is no need of a re-training phase during transcoding. Therefore, the off-line training approach is used in $\text{T2B}_{\text{ML}}$.

**Table 2.6:** The split-flag prediction accuracy and the PU size matching rate

| CFG | Seq | Resolution | Split flag prediction accuracy [%] | | | | PU size matching rate [%] | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Trivial | T2B | $T2B_{ML}$ | $B2T_{TLP}$ | Trivial | T2B | $T2B_{ML}$ | $B2T_{TLP}$ |
| LP | BQSquare* | 416x240 | 67.55 | 84.75 | 82.83 | 83.21 | 78.67 | 87.20 | 87.62 | 87.43 |
| | FourPeople* | 1280x720 | 61.27 | 83.90 | 82.62 | 81.80 | 86.87 | 91.54 | 91.31 | 91.76 |
| | BQMall | 832x480 | 63.46 | 81.56 | 80.46 | 79.55 | 80.92 | 88.12 | 87.99 | 88.43 |
| | PartyScene | 832x480 | 62.83 | 81.13 | 79.48 | 79.64 | 72.34 | 83.37 | 83.65 | 83.62 |
| | Kimono | 1920x1080 | 57.42 | 79.73 | 79.12 | 76.87 | 83.34 | 90.61 | 90.21 | 90.93 |
| | BasketballDrive | 1920x1080 | 60.96 | 81.66 | 80.14 | 79.29 | 83.86 | 89.82 | 89.68 | 90.09 |
| | **Average** | | **64.08** | **83.60** | **81.63** | **81.10** | **82.51** | **89.32** | **89.30** | **89.58** |
| RA | BQSquare* | 416x240 | 64.08 | 90.76 | 88.86 | 87.48 | 85.20 | 94.30 | 94.35 | 93.71 |
| | FourPeople* | 1280x720 | 65.85 | 90.89 | 89.77 | 87.01 | 89.09 | 95.29 | 95.00 | 94.55 |
| | BQMall | 832x480 | 65.09 | 87.47 | 86.33 | 83.42 | 83.51 | 91.94 | 91.71 | 91.45 |
| | PartyScene | 832x480 | 60.94 | 86.99 | 85.02 | 83.07 | 80.24 | 91.21 | 91.23 | 90.19 |
| | Kimono | 1920x1080 | 60.04 | 86.20 | 85.39 | 81.96 | 87.09 | 93.81 | 93.50 | 93.64 |
| | BasketballDrive | 1920x1080 | 63.37 | 85.81 | 84.31 | 81.80 | 86.58 | 92.65 | 92.44 | 92.50 |
| | **Average** | | **65.34** | **88.66** | **86.93** | **84.57** | **85.38** | **93.04** | **92.88** | **92.54** |

* The sequences in the training set for modelling of $T2B_{ML}$.

### 2.3.6 Experimental results of the proposed block partitioning techniques

The proposed methods are evaluated by comparing the performance of several transcoders. These include an unmodified decoder-encoder cascade transrater, trivial methods and various fast encoding and transcoding algorithms. These trivial methods copy the CU size and/or PU size from the input bit stream to the output bit stream. First, the evaluation conditions are described. Thereafter, the experimental results of Trivial, $T2B_{ML}$, and $B2T_{TLP}$ at both the CU and PU levels are analyzed in terms of bit rate increasing and transcoding time. These methods are evaluated in both variable bit rate (VBR) and constant bit rate (CBR). Finally, coding performance of our proposed methods is compared with state-of-the-art fast HEVC encoding and transcoding algorithms.

#### 2.3.6.1 Evaluating conditions

In the experiments, all sequences of classes B, C, D, and E listed in [23] excluding the training sequences have been tested. The experiments are tested on a platform using 64-bit Scientific Linux 6 operating system running on a PC with an integrated Intel dual-socket quad-core 2.27 GHz and 12 GB RAM. The proposed algorithms are implemented in the HEVC test model (HM) 7 reference software [22] under the test conditions defined in [23]. Search mode 'TZSearch' and 'FEN' (fast encoder decision) are enabled. In other words, the proposed algorithms are compared with best speed performance of HM. CU structure is set at a maximum size of $64\times64$ pixels and a maximum depth of 4 ($8\times8$ pixels CU). The performance of the proposed scheme is evaluated in terms of Bjøntegaard Delta Bit rate (BDBR) [27] for both low delay P main (LP) and random access (RA) configurations. For LP, only the first frame is intra-coded while the intra period is set to 32 for RA. In the BDBR measurement, peak signal to noise ratio (PSNR [28]) calculations between the re-encoded and the original sequence are used. Additionally, complexity reduction, which is measured by the time saving (TS), is given by:

$$TS(\%) = \frac{T_{Original}\ (ms) - T_{Proposed}\ (ms)}{T_{Original}\ (ms)} \tag{2.6}$$

Herein, $T_{Proposed}$ represents the total transrating time using the proposed method while TOriginal constitutes the total transrating time using an unmodified cascaded decoder-encoder setup. Since the same code base is used for the original encoder, the trivial methods, and all proposed techniques, the difference in time saving percentage gives an indication of the complexity reductions.

#### 2.3.6.2 Coding performance under the VBR scenario

In the VBR scheme, the input video stream is encoded using a constant $QP_1 \in \{22, 27, 32, 37\}$. This video is reconstructed and coded at a lower bit rate using a higher constant $QP_2$. The difference of the input and output quantization parameter $\Delta$QP is set as $\{2, 4, 6\}$. Firstly, we present the performance of the B2T approach in terms of different quality matrix. Then, an analysis on the flexible transcoding complexity of $T2B_{ML}$ using thresholds is provided, followed by an elaboration of the experimental results of all described algorithms.

**The visual subjective quality performance of the B2T approach**

There are two popular metrics which evaluate the similarity of two pictures. They include the structural similarity (SSIM [29]) index and the peak signal-to-noise ratio (PSNR [28]). In these terms, SSIM also predicts how users perceive video distortions. The experimental results of the B2T approach with the use of SSIM and PSNR is presented in Table 2.7, Table 2.8, and Table 2.9. In this experiments, SSIM and PSNR are obtained by comparing the original video and the reconstructed version of the transcoded video. The degradations of these terms ($\Delta$SSIM and $\Delta$PSNR) of a method indicate the differences of these terms and that obtained by an unmodified cascade transcoding loop.

It is observed from Table 2.8 that when the input QP increases, PSNR and SSIM are both degraded. This is due to the fact that more quantization error are added. When the input QP increases, the encoded video is more smooth. The distortion is easily detected in such a smooth video by human eyes. Therefore, $\Delta$SSIM increases.

Table Average, we can see the degradation of PSNR is notably larger than that of SSIM. Therefore, PSNR will be used to evaluated the performance loss of the proposed techniques. Doing so, the performance loss might be more clear.

**Table 2.7:** The performance of B2T with the use of VBR and the LD configuration with ΔQP of 2 - in terms of SSIM and PSNR

| input QP | $SSIM_{HM}$ | $SSIM_{B2T}$ | $PSNR_{HM}$ | $PSNR_{B2T}$ | ΔSSIM | ΔPSNR | ΔSSIM (%) | ΔPSNR(%) |
|---|---|---|---|---|---|---|---|---|
| 22 | 0.99364 | 0.99360 | 38.18843 | 38.16395 | 0.00003 | 0.02449 | 0.003 | 0.064 |
| 27 | 0.98694 | 0.98684 | 35.13683 | 35.10417 | 0.00010 | 0.03266 | 0.010 | 0.093 |
| 32 | 0.97458 | 0.97444 | 32.21377 | 32.18507 | 0.00015 | 0.02870 | 0.015 | 0.089 |
| 37 | 0.95408 | 0.95379 | 29.48154 | 29.45476 | 0.00029 | 0.02678 | 0.030 | 0.091 |
| Average | 0.97187 | 0.97169 | 32.27738 | 32.24800 | **0.00018** | **0.02938** | **0.018** | **0.091** |

**Table 2.8:** The performance of B2T with the use of VBR and the LD configuration with ΔQP of 6 - in terms of SSIM and PSNR

| input QP | $SSIM_{HM}$ | $SSIM_{B2T}$ | $PSNR_{HM}$ | $PSNR_{B2T}$ | ΔSSIM | ΔPSNR | ΔSSIM (%) | ΔPSNR(%) |
|---|---|---|---|---|---|---|---|---|
| 22 | 0.98988 | 0.98982 | 36.23498 | 36.20797 | 0.00006 | 0.02701 | 0.006 | 0.075 |
| 27 | 0.97979 | 0.97965 | 33.23792 | 33.21193 | 0.00013 | 0.02599 | 0.013 | 0.078 |
| 32 | 0.96256 | 0.96230 | 30.44220 | 30.41355 | 0.00025 | 0.02865 | 0.026 | 0.094 |
| 37 | 0.93481 | 0.93454 | 27.83561 | 27.81399 | 0.00027 | 0.02162 | 0.029 | 0.078 |
| Average | 0.95905 | 0.95883 | 30.50524 | 30.47982 | **0.00022** | **0.02542** | **0.023** | **0.083** |

**Table 2.9:** The average performance of T2B with the use of VBR and the LD configuration - in terms of SSIM and PSNR

| ΔQP | $SSIM_{HM}$ | $SSIM_{B2T}$ | $PSNR_{HM}$ | $PSNR_{B2T}$ | ΔSSIM | ΔPSNR | ΔSSIM (%) | ΔPSNR(%) |
|---|---|---|---|---|---|---|---|---|
| 2 | 0.97731 | 0.97717 | 33.75514 | 33.72699 | 0.00014 | 0.02816 | 0.014 | 0.083 |
| 4 | 0.97251 | 0.97237 | 32.85060 | 32.82664 | 0.00014 | 0.02395 | 0.014 | 0.073 |
| 6 | 0.96676 | 0.96658 | 31.93768 | 31.91186 | 0.00018 | 0.02582 | 0.019 | 0.081 |
| Average | 0.97219 | 0.97204 | 32.84781 | 32.82183 | **0.00015** | **0.02598** | **0.016** | **0.079** |

**Table 2.10:** Coding performance of $T2B_{ML}$ with different thresholds

| CFG | $Thr_1[\%]$ | $Thr_2[\%]$ | BDBR[%] | TS[%] |
|---|---|---|---|---|
| LP | 85 | 90 | 0.77 | 53.00 |
| | 75 | 90 | 1.14 | 54.50 |
| | 75 | 85 | 1.49 | 64.14 |
| | 50 | 75 | 2.78 | 67.36 |
| RA | 85 | 90 | 0.40 | 57.20 |
| | 75 | 90 | 0.65 | 59.40 |
| | 75 | 85 | 0.88 | 63.99 |
| | 50 | 75 | 1.69 | 69.00 |

**Complexity-scalable transcoder using the machine learning based method**

A trade-off between transcoding complexity and bit rate loss can be achieved by the $T2B_{ML}$ method in use of two thresholds. It is clear that when the proposed thresholds increase, the number of RD evaluations increases accordingly. Consequently, the rate-distortion complexity trade-off of the transrating architecture varies among these thresholds. Different pairs of ($Thr_1$, $Thr_2$) have been evaluated to reach a usable trade-off. Experimental results using different relevant values are presented in Table 2.10. When both of these thresholds are high (0.85, 0.90), the bit rate penalty is very small (0.77%). However, transrating complexity reduction is then small (53%) as well. In contrast, when these two thresholds are small (0.50, 0.75), the bit rate penalty is high with an 2.78% increase and the complexity reduction is also high (67.36%). When $Thr_1$ is 0.75 and $Thr_2$ is 0.85, this method achieves 64% complexity reduction with a slight increase of BDBR (1.49%). In the following evaluation, these thresholds (0.75, 0.85) are used as a default for $T2B_{ML}$.

**Coding performance analysis**

The experiment under the LP configuration is analyzed first. Then, the performance with the use of the RA configuration is explored. Results are visually summarized in Figure 2.11. As can be seen in Figure 2.11, a trade-off between coding performance and transrating complexity can be achieved by the proposed methods. Depending on the required complexity reduction, one of the above techniques can be used to guarantee the highest RD.

For LP configuration, detailed experimental results for each class of the Trivial, $T2B_{ML}$ and $B2T_{TLP}$ architectures that optimize the evaluation at both CU and PU level are presented in Table 2.11 and Table 2.12. Comparison of the average performance of these methods is presented in Table 2.13 and visualized in Figure 2.10.

As can be seen in Table 2.11 and Table 2.12, the trivial methods can achieve a low complexity for both CU (Trivial) and PU ($\text{Trivial}_{\text{PU}}$) evaluations. On average, Trivial and $\text{Trivial}_{\text{PU}}$ can reduce transrating complexity by 75.65% and 91.12%, respectively. The low complexity of these methods is achieved by directly copying the CU and PU structures from the input bit stream to the output bit stream. However, the simple re-use of the input CU and PU structures results in BDBR losses, which strongly increase when raising $\Delta$QP values. This could be expected from the probabilities in Table 2.2 and Table 2.5, which indicate that the CU and PU sizes in the output streams typically become larger to increase $\Delta$QP values. The trivial methods, however, only evaluate the CUs at the depths of the input CUs and skip the evaluation at lower depths. As a result, the trivial method increases bit rate on average by 7.49% while an increase by 15.23% is measured in the $\text{Trivial}_{\text{PU}}$ approach.

The complexity reductions of $\text{T2B}_{\text{ML}}$ and $\text{B2T}_{\text{TLP}}$ are smaller than those of the trivial methods. However, these proposed methods significantly outperform the trivial approaches in terms of coding performance. The proposed method $\text{T2B}_{\text{ML}}$ reduces the complexity of transrating by 64.14% with a 1.49% penalty in bit rate. When PU evaluation is optimized, the complexity reduction increases to 76.22% with a negligible bit rate increase by about 2.23%. The complexity reduction of $\text{B2T}_{\text{TLP}}$ is higher than $\text{T2B}_{\text{ML}}$ (66.18% and 79.65% for CU and PU evaluations). $\text{B2T}_{\text{TLP}}$ has a loss of coding performance of 1.93% for CU and 2.65% bit rate increase for PU evaluations.

Figure 2.8 shows examples of the CU size results obtained by applying our proposed algorithms and the trivial method. We defined the difference between coding depths (dCU) of a frame obtained by a method and this frame was obtained by $\text{HEVC}_{\text{Anchor}}$ transrating as the average of absolute depth differences among pixels. As can be observed from Figure 2.8, the CU structures of $\text{B2T}_{\text{TLP}}$ and $\text{T2B}_{\text{ML}}$ are very similar to the CU structure obtained by $\text{HEVC}_{\text{Anchor}}$ transrating. The $\text{Trivial}_{\text{PU}}$ method encodes CUs using higher depths compared to $\text{HEVC}_{\text{Anchor}}$ transrating.

When the $\Delta$QP increases, we see different effects on the complexity reduction and coding performance of the $\text{T2B}_{\text{ML}}$ and B2T methods. The correlation between coding information of CUs in the input and output bit stream gets weaker when $\Delta$QP increases. Consequently, the probability of correct predictions in $\text{T2B}_{\text{ML}}$ is reduced. Therefore, the number of CU re-evaluations increases, resulting in a higher transrating complexity (and a reduction of the bit rate penalty). However, the initial CU structure for evaluating a CU in B2T is unchanged when QP increases. As a result, the transrating complexity reduction of B2T is only slightly reduced (and remains around 63%).

Since the difference between the initial CU structure and the optimized CU structure is larger when $\Delta$QP increases, the bit rate penalty of B2T increases with

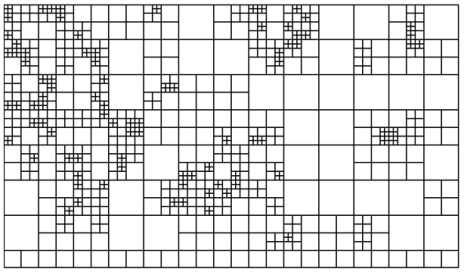
$\Delta$QP. The coding performance comparison of these two methods when $\Delta$QP increases is depicted in Figure 2.9.

The rate-distortion plots for the methods optimizing the PU evaluation are depicted in Figure 2.10. As can be seen, RD performance of the proposed methods is similar to an unmodified HEVC cascaded decoder-encoder ($HEVC_{Anchor}$) and clearly better than the $Trivial_{PU}$ method.

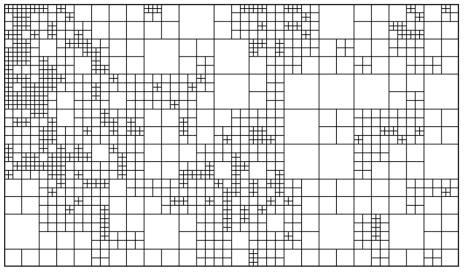
Table 2.13 shows the coding performance of all proposed methods and the trivial methods. Notice that the difference in performance for each class is not remarkable as demonstrated in Table 2.11; therefore, the average performance of all classes is given in the remainder analysis.

Results of the experiment under RA configuration presented in Table 2.15 have demonstrated that the proposed algorithm can significantly reduce the transcoding complexity with a negligible bit rate penalty. With a 51.67% complexity reduction, the T2B method results in a very poor bit rate increase by 0.29%. For a higher complexity reduction of about 64%, B2T and $T2B_{ML}$ show the same coding performance with a bit rate increase by 0.9%. At the PU level, T2B can reduce 66% complexity with a bit rate error of only 0.59%, which is smaller than the error of B2T and $T2B_{ML}$ in the CU evaluation level. Therefore, with the complexity reduction target of 66% (2/3 reduction), $B2T_{PU}$ is the most advisable solution. However, when the transcoding complexity needs to be reduced further, $B2T_{TLP\_PU}$ and $T2B_{ML\_PU}$ are more reliable. These methods are able to reduce a notable reduction of 80% to 82.4% transcoding complexity with about a 2% bit rate loss compared to the non-optimized transcoder. Between these two algorithms, $B2T_{TLP\_PU}$ is preferred over $T2B_{ML\_PU}$ regarding the implementation performance since a model and a complex prediction need to be integrated in the $T2B_{ML\_PU}$ approach.

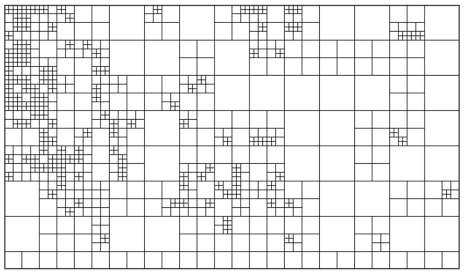
It should be noticed that the proposed algorithms are only applied for inter frames. In doing so, the coding performance loss is recovered at intra frames. As a result, the performance of using RA is better than using LP. For instance, $B2T_{TLP\_PU}$ under RA leads to a higher complexity reduction (82.4%) compared to using LP. However, the bit rate penalty under RA is lower (1.95% compared to 2.65%).



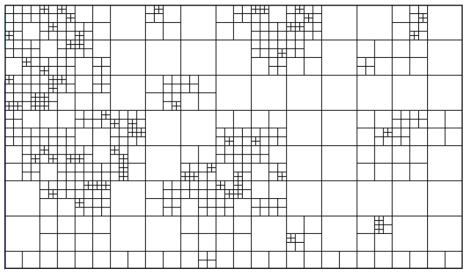
(a) Anchor cascaded decoder-encoder transcoder



(b) $\text{Trivial}_{\text{PU}}$ dCU = 0.52



(c) $\text{T2B}_{\text{ML\_PU}}$ dCU = 0.33



(d) $\text{B2T}_{\text{TLP\_PU}}$ dCU = 0.27

**Figure 2.8:** CU structures generated by the different algorithms for the $200^{th}$ frame in the PartyScene sequence, $\text{QP}_1$ = 32, $\Delta$QP = 6. The VBR and LP configuration are used in this experiment. The CU structures obtained by our proposed methods are similar to the ones obtained by the $\text{HEVC}_{\text{Anchor}}$ transcoder.

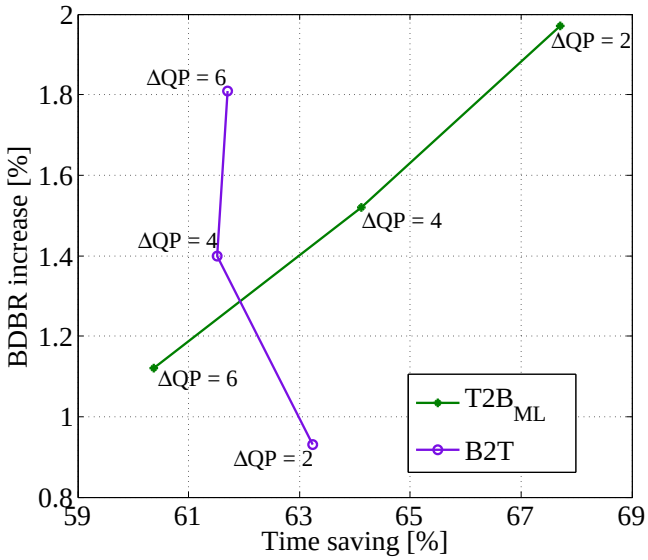
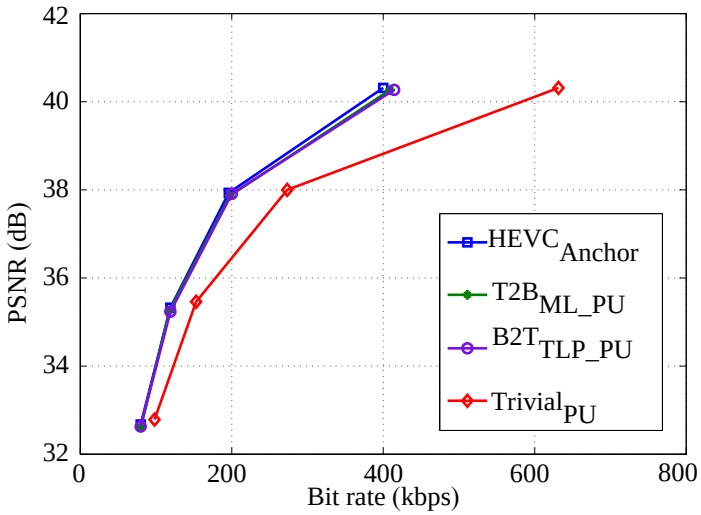
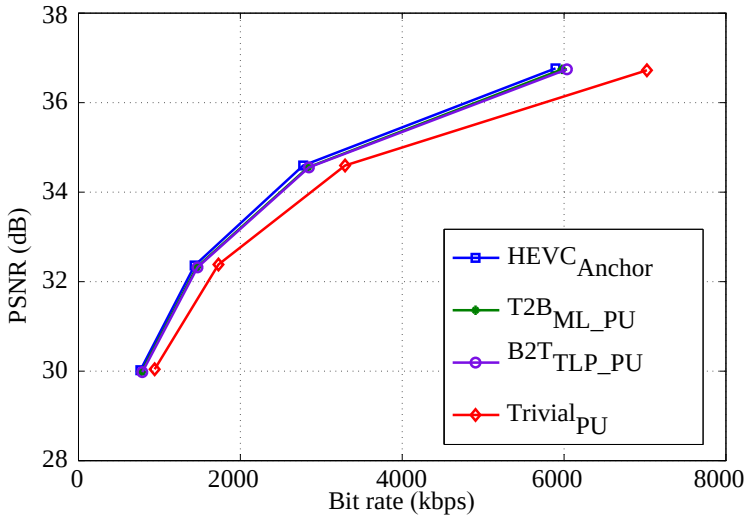



**Figure 2.9:** BDBR increase and time saving of $T2B_{ML}$ and B2T with the use of VBR and the LP configuration.

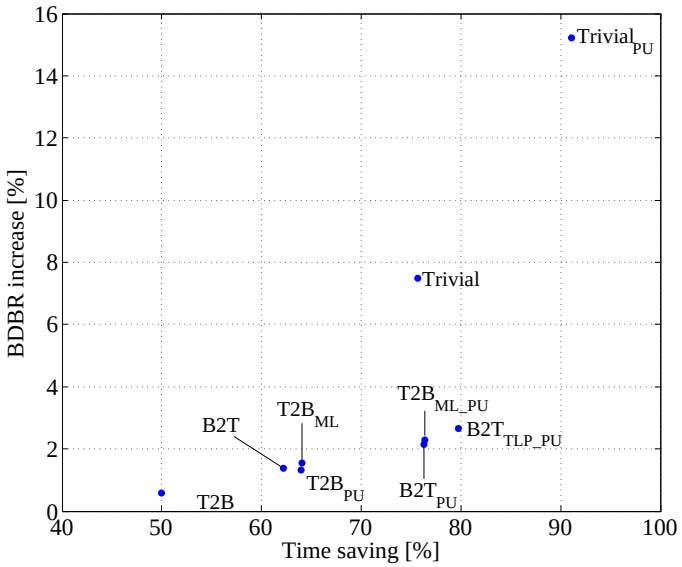



(a) The Johnny sequence

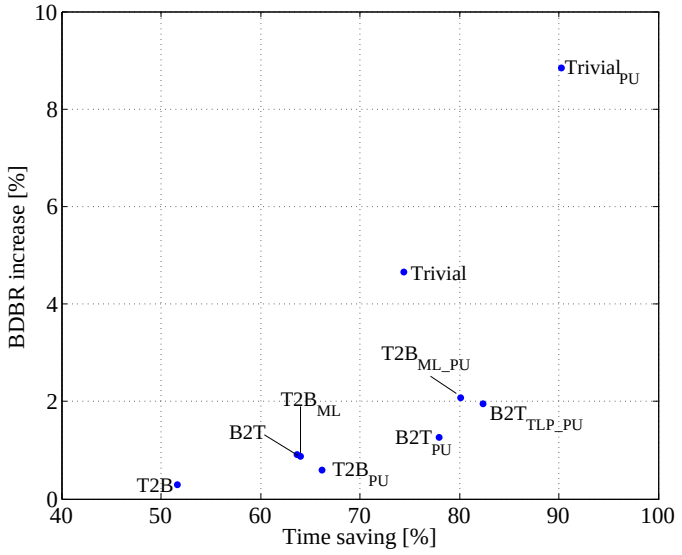



(b) The BasketballDrive sequence

**Figure 2.10:** RD performance for transrating with a $\Delta$QP = 6 using the CBR scheme and the LP configuration. The RD performances of our proposed methods match to the performance of the $\text{HEVC}_{\text{Anchor}}$ transcoder and significantly outperform the performance of the $\text{Trivial}_{\text{PU}}$ transcoder.




(a) Low delay configuration.




(b) Random access configuration.

**Figure 2.11:** Visualization of the performance results of the proposed techniques with the use of the variable bit rate profile.

**Table 2.11:** Performance of different described techniques compared to a decoder-encoder cascade under the LP configuration with the VBR scheme and the CU optimization level.

| Method | Class | BDBR increase[%] | | | Time Saving (TS)[%] | | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | BDBR | TS |
| Trivial | B | 3.74 | 7.01 | 10.94 | 74.51 | 75.10 | 77.01 | 7.23 | 75.54 |
| | C | 2.88 | 6.68 | 11.16 | 71.98 | 73.34 | 73.93 | 6.90 | 73.08 |
| | D | 2.89 | 6.11 | 9.71 | 72.18 | 72.00 | 72.20 | 6.24 | 72.13 |
| | E | 5.64 | 9.64 | 13.89 | 76.41 | 78.90 | 78.69 | 9.72 | 78.00 |
| | **Avg.** | **3.78** | **7.32** | **11.37** | **74.52** | **75.48** | **76.94** | **7.49** | **75.65** |
| $T2B_{ML}$ | B | 2.11 | 1.73 | 1.38 | 67.54 | 63.30 | 59.38 | 1.74 | 63.41 |
| | C | 1.36 | 1.57 | 1.19 | 63.95 | 59.34 | 53.16 | 1.27 | 58.62 |
| | D | 1.12 | 1.29 | 1.04 | 61.19 | 55.72 | 48.98 | 1.22 | 55.30 |
| | E | 3.28 | 1.46 | 0.71 | 72.94 | 72.84 | 71.43 | 1.82 | 72.40 |
| | **Avg.** | **1.85** | **1.52** | **1.11** | **67.60** | **63.96** | **60.03** | **1.49** | **64.14** |
| $B2T_{TLP}$ | B | 1.45 | 2.29 | 2.97 | 65.89 | 65.59 | 64.96 | 2.24 | 65.48 |
| | C | 1.04 | 2.08 | 2.93 | 61.91 | 62.30 | 60.71 | 2.02 | 61.64 |
| | D | 1.09 | 1.86 | 2.45 | 61.43 | 59.53 | 58.11 | 1.80 | 59.69 |
| | E | 1.11 | 1.57 | 1.87 | 71.73 | 73.90 | 73.36 | 1.52 | 73.00 |
| | **Avg.** | **1.20** | **1.99** | **2.60** | **66.27** | **66.47** | **65.81** | **1.93** | **66.18** |

**Table 2.12:** Performance of different described techniques compared to a decoder-encoder cascade with the VBR scheme under the LP configuration and the PU optimization level.

| Method | Class | BDBR increase[%] | | | Time Saving (TS)[%] | | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | BDBR | TS |
| $\text{Trivial}_{\text{PU}}$ | B | 6.39 | 12.75 | 19.98 | 90.78 | 91.08 | 91.25 | 13.04 | 91.04 |
| | C | 6.14 | 14.78 | 24.94 | 90.00 | 90.36 | 90.79 | 15.29 | 90.39 |
| | D | 5.86 | 13.30 | 22.20 | 89.85 | 89.56 | 89.99 | 13.79 | 89.80 |
| | E | 10.72 | 19.77 | 29.27 | 91.56 | 92.38 | 92.33 | 19.92 | 92.09 |
| | **Avg.** | **7.18** | **14.88** | **23.64** | **90.80** | **91.19** | **91.36** | **15.23** | **91.12** |
| $\text{T2B}_{\text{ML\_PU}}$ | B | 1.49 | 2.30 | 2.88 | 76.74 | 75.28 | 74.39 | 2.23 | 75.47 |
| | C | 1.78 | 2.60 | 2.23 | 77.73 | 72.67 | 65.76 | 2.20 | 72.02 |
| | D | 1.45 | 2.25 | 2.04 | 74.97 | 69.77 | 62.92 | 1.91 | 69.10 |
| | E | 3.76 | 2.46 | 2.02 | 87.13 | 84.91 | 83.51 | 2.75 | 85.19 |
| | **Avg.** | **1.97** | **2.40** | **2.32** | **78.13** | **76.17** | **74.36** | **2.23** | **76.22** |
| $\text{B2T}_{\text{TLP\_PU}}$ | B | 1.68 | 2.84 | 3.60 | 79.53 | 79.50 | 78.42 | 2.71 | 79.15 |
| | C | 1.63 | 3.16 | 3.93 | 76.20 | 76.22 | 70.76 | 2.91 | 74.40 |
| | D | 1.58 | 2.78 | 3.52 | 75.54 | 73.79 | 72.19 | 2.63 | 73.84 |
| | E | 1.71 | 2.43 | 2.81 | 85.29 | 86.24 | 85.68 | 2.31 | 85.74 |
| | **Avg.** | **1.65** | **2.80** | **3.48** | **79.96** | **80.15** | **78.84** | **2.65** | **79.65** |

**Table 2.13:** Performance of all described techniques compared to a decoder-encoder cascade under the VBR condition

| CFG | Method | BDBR increase[%] | | | Time Saving (TS)[%] | | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | ΔQP = 2 | ΔQP = 4 | ΔQP = 4 | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | BDBR | TS |
| LP | Trivial | 3.78 | 7.32 | 11.37 | 74.52 | 75.48 | 76.94 | 7.49 | 75.65 |
| | T2B | 0.45 | 0.65 | 0.62 | 47.99 | 50.54 | 51.34 | 0.57 | 49.96 |
| | $T2B_{ML}$ | 1.85 | 1.52 | 1.11 | 68.00 | 64.20 | 60.23 | 1.49 | 64.14 |
| | B2T | 0.93 | 1.41 | 1.81 | 63.24 | 61.50 | 61.79 | 1.38 | 62.18 |
| | $Trivial_{PU}$ | 7.18 | 14.88 | 23.64 | 90.82 | 91.19 | 91.36 | 15.23 | 91.12 |
| | $T2B_{PU}$ | 0.99 | 1.45 | 1.51 | 62.82 | 64.04 | 64.96 | 1.32 | 63.94 |
| | $T2B_{ML\_PU}$ | 1.97 | 2.40 | 2.32 | 78.13 | 76.17 | 74.36 | 2.23 | 76.22 |
| | $B2T_{PU}$ | 1.42 | 2.27 | 2.73 | 77.23 | 76.31 | 75.37 | 2.14 | 76.30 |
| | $B2T_{TLP\_PU}$ | 1.65 | 2.80 | 3.48 | 79.96 | 80.15 | 78.84 | 2.65 | 79.65 |
| RA | Trivial | 2.13 | 4.57 | 7.29 | 74.18 | 74.42 | 74.63 | 4.66 | 74.41 |
| | T2B | 0.27 | 0.32 | 0.29 | 50.99 | 51.66 | 52.38 | 0.29 | 51.67 |
| | $T2B_{ML}$ | 1.05 | 0.95 | 0.64 | 68.46 | 63.47 | 60.04 | 0.88 | 63.99 |
| | B2T | 0.57 | 0.95 | 1.21 | 64.68 | 63.51 | 62.79 | 0.91 | 63.66 |
| | $Trivial_{PU}$ | 3.70 | 8.62 | 14.23 | 90.15 | 90.24 | 90.31 | 8.85 | 90.24 |
| | $T2B_{PU}$ | 0.44 | 0.68 | 0.66 | 65.75 | 66.14 | 66.76 | 0.59 | 66.22 |
| | $T2B_{ML\_PU}$ | 2.51 | 2.21 | 1.48 | 80.39 | 80.71 | 79.22 | 2.07 | 80.11 |
| | $B2T_{PU}$ | 0.72 | 1.43 | 1.63 | 79.15 | 77.89 | 76.85 | 1.26 | 77.96 |
| | $B2T_{TLP\_PU}$ | 1.04 | 2.03 | 2.77 | 83.07 | 82.40 | 81.71 | 1.95 | 82.39 |

**Table 2.14:** Input bit rate setup for CBR

| Class | Resolution | Bit rate of the input video $R_i$[kbps] | | | |
|---|---|---|---|---|---|
| | | $R_{i0}$ | $R_{i1}$ | $R_{i2}$ | $R_{i3}$ |
| B | 1920x1080 | 12000 | 5000 | 2500 | 1500 |
| C | 832x480 | 2300 | 1000 | 500 | 300 |
| D | 416x240 | 600 | 250 | 100 | 75 |
| E | 1280x720 | 5000 | 2000 | 1000 | 600 |

**Coding performance at different input quantization parameters**

Because the proposed approaches mostly use the CTU structure in the input video to evaluate the CTU structures in the output video, the performance of proposed method depends on the behaviour of the input CTU structures, which depends on the quantization parameter of the input video or the bit rate of the input video.

As can be seen from Figure 2.12, when the input quantization parameter increases, the performance of the proposed approaches grows up accordingly.

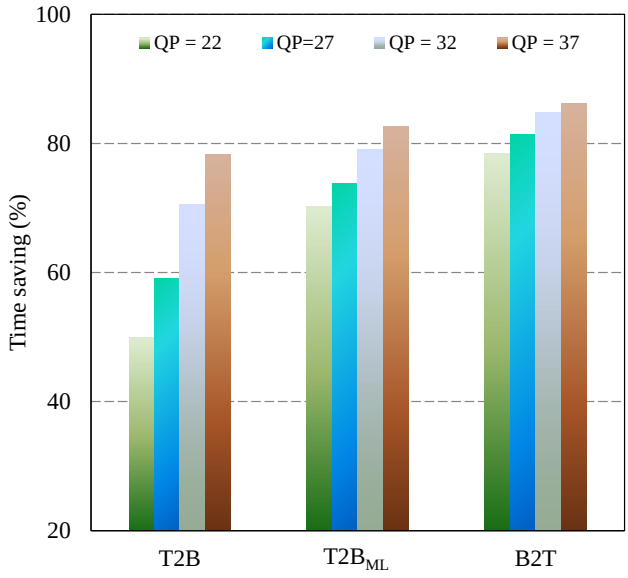
When the input quantization is low, the CTU structures of the input video are complex with many small CUs at depth 3. Therefore, the re-encoder may have to evaluate a high amount of CUs in a large range of the depth (from 3 to 0). Consequently, the complexity reduction is low.

On the other hand, when input QP increase, the complexity of the input CTU structures is reduced with many CUs at low depths. Therefore, the re-encoder, in many cases, does not need to go to higher depths. In this case, the number of RD evaluation is reduced, thus the complexity reduction is increased.

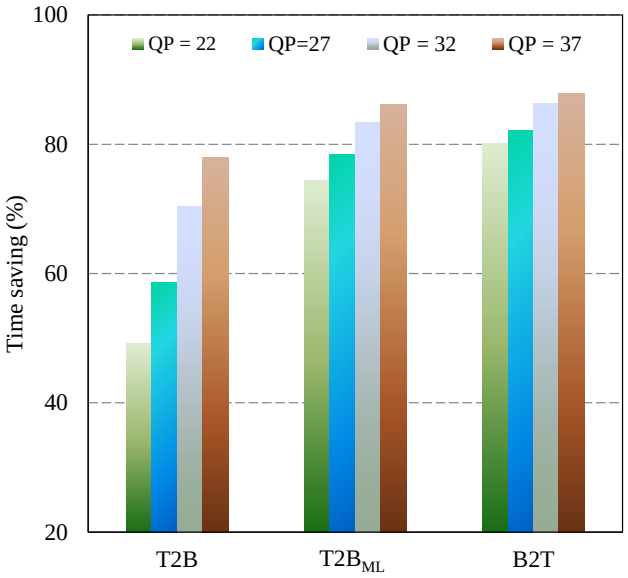
#### 2.3.6.3 Coding performance under the CBR scenario

In practical scenarios, when streamed over the internet, the video may be switched between networks with different bandwidth limitations. In such scenarios, the video is transcoded in use of a constant bit rate encoder. In this section, the proposed algorithms are evaluated in the following scheme. The input video is encoded at a constant bit rate of $R_i$(kbps). Afterwards, the video is reconstructed and transcoded to a lower bit rate $R_o$(kbps) given by Equation 2.2. Three bit rate reduction factors ($\alpha$) of 0.15, 0.30 and 0.45 have been examined. For higher bit rate reductions, other transcoding approaches (spatial or temporal transcoding) are suggested. Both LP and RA configurations are tested. For BDBR evaluation, the experiment is carried out for four input bit rate values ($R_{i0}$, $R_{i1}$, $R_{i2}$, $R_{i3}$) as defined in Table 2.14.

Table 2.15 presents the performance of the proposed algorithms under CBR test condition. In addition, these results are visualized in Figure 2.13.

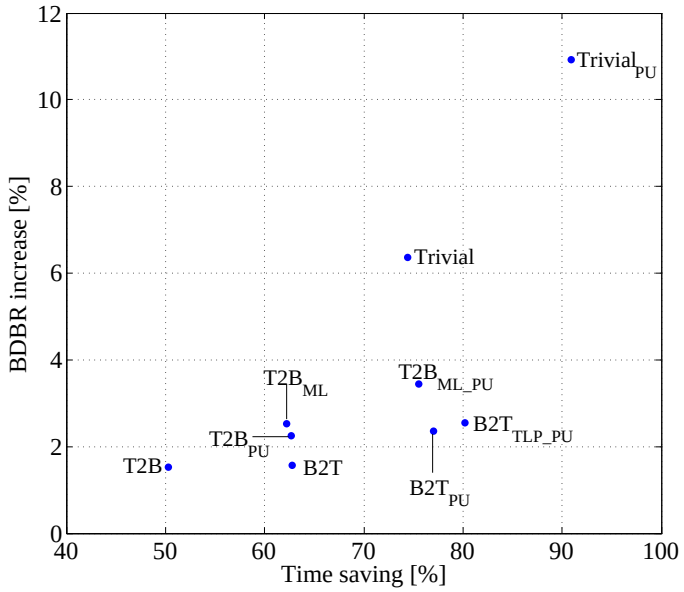



(a) $\Delta$QP = 6.

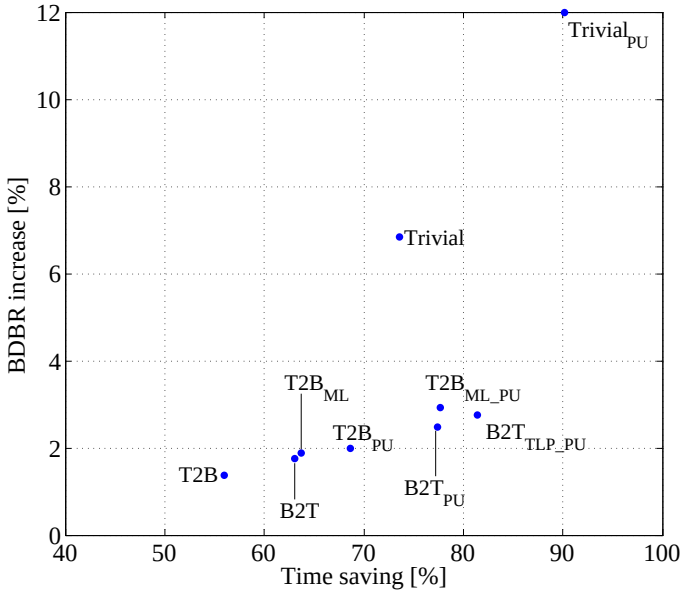



(b) Average performance of different $\Delta$QP values.

**Figure 2.12:** Visualization of the performance results of the proposed techniques with different input QPs - the LD and VBR configuration. The proposed solutions demonstrate high performance at high quantization parameters of the input video.




(a) Low delay configuration.




(b) Random access configuration.

**Figure 2.13:** Visualization of the performance results of the proposed techniques with the use of the constant bit rate profile.

In comparison with the use of VBR, the proposed algorithms yield the same complexity reduction. In contrast, the performance is slightly worse in terms of bit rate loss. This loss is generated due to a large range of QP differences between the input and output video. This large QP difference may appear since QPs of the input and output video are independently derived to achieve the input and output rates. Note that the difference in bit rate loss between CBR and VBR is below 1%. For instance, a $\Delta$QP of 2 in the VBR configuration results in a bit rate reduction of 30%. The performance of $B2T_{PU}$ is a time saving of 77% with a bit rate penalty of 1.42%. With a similar bit rate reduction of 30% in VBR setting, this approach is able to achieve a similar time saving of 77%. However, this results in a higher bit rate penalty of 2.68%.

**Table 2.15:** Performance of all described techniques compared to a decoder-encoder cascade under the CBR condition

| CFG | Method | BDBR increase[%] | | | Time Saving (TS)[%] | | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\alpha = 0.15$ | $\alpha = 0.30$ | $\alpha = 0.45$ | $\alpha = 0.15$ | $\alpha = 0.30$ | $\alpha = 0.45$ | BDBR | TS |
| LP | Trivial | 3.87 | 5.49 | 9.75 | 74.31 | 74.59 | 74.49 | 6.37 | 74.46 |
| | T2B | 1.40 | 1.27 | 1.89 | 49.89 | 50.32 | 50.64 | 1.52 | 50.28 |
| | T2B_ML | 2.40 | 2.12 | 3.04 | 68.12 | 61.84 | 56.58 | 2.52 | 62.18 |
| | B2T | 1.81 | 1.95 | 0.92 | 63.80 | 62.94 | 61.47 | 1.56 | 62.74 |
| | Trivial_PU | 5.67 | 10.58 | 16.55 | 90.77 | 90.97 | 90.96 | 10.93 | 90.90 |
| | T2B_PU | 1.70 | 2.06 | 2.94 | 62.64 | 62.80 | 62.76 | 2.24 | 62.73 |
| | T2BML_PU | 2.57 | 3.82 | 3.92 | 81.97 | 75.34 | 69.40 | 3.44 | 75.57 |
| | B2T_PU | 2.22 | 2.68 | 2.18 | 78.12 | 77.20 | 75.74 | 2.36 | 77.02 |
| | B2TTLP_PU | 2.16 | 3.14 | 2.33 | 81.07 | 80.39 | 79.23 | 2.54 | 80.23 |
| RA | Trivial | 5.75 | 7.01 | 7.80 | 73.57 | 73.59 | 73.69 | 6.85 | 73.61 |
| | T2B | 3.00 | 0.20 | 0.94 | 55.45 | 55.80 | 56.57 | 1.38 | 55.94 |
| | T2B_ML | 1.70 | 2.15 | 1.82 | 68.61 | 62.58 | 59.84 | 1.89 | 63.68 |
| | B2T | 2.03 | 2.64 | 0.58 | 64.01 | 63.02 | 62.03 | 1.75 | 63.02 |
| | Trivial_PU | 8.79 | 12.04 | 15.48 | 90.17 | 90.19 | 90.23 | 12.10 | 90.20 |
| | T2B_PU | 1.78 | 2.70 | 1.46 | 68.72 | 68.70 | 68.49 | 1.98 | 68.64 |
| | T2BML_PU | 2.73 | 3.61 | 2.45 | 82.68 | 76.99 | 73.34 | 2.93 | 77.67 |
| | B2T_PU | 1.95 | 3.30 | 2.21 | 78.05 | 77.87 | 76.36 | 2.48 | 77.43 |
| | B2TTLP_PU | 2.80 | 3.01 | 2.47 | 81.81 | 81.47 | 80.90 | 2.76 | 81.40 |

Again, the proposed methods demonstrate a superior performance over trivial approaches. Among these proposed algorithms, the $B2T_{TLP\_PU}$ shows the best performance with a remarkable complexity reduction of about 80% with bit rate losses of 2.54% and 2.76% for LP and RA, respectively.

#### 2.3.6.4 Performance comparison with the state-of-the-art

Since transrating for HEVC is a novel topic, to evaluate the performance of the proposed methods, these methods are compared with various fast encoding algorithms [10, 11, 13, 14, 30–32], and an HEVC composition transcoder [19] in terms of transrating complexity and bit rate increases. The bit rates are set using a VBR scheme.

These fast encoding algorithms are used to encode the reconstructed video of the input stream. It should be noted that the input coding information is not utilized in these fast encoding references for HEVC except by De Praeter [19]. Therefore, the comparison is not entirely fair. However, the significant performance improvement of our proposed methods implies that the proposed methods are notably efficient in reducing the complexity of an HEVC transcoder. Since each reference work yields different values of BDBR and time saving, we obtain the ratio between BDBR and time saving (B/T) [14]. This parameter shows the amount of BDBR loss per time saving. Lower B/T infers better performance. The coding performance comparison is shown in Table 2.16.

In general, our proposed methods are notably more time-saving than other methods at both CU and PU optimization levels. In terms of B/T evaluation, bit rate loss of our proposed methods is usually lower than or equal to other methods for the same complexity reduction. There are a few exceptions such as joint CU and PU optimization under the LP configuration where Liquan [32] yields a lowest B/T. However, this method achieves a complexity reduction of 52% which is much less than the complexity of $T2B_{ML}$ and B2T (76%).

## 2.4 Conclusions

In this chapter, we proposed several optimized transrating techniques for HEVC. The correlation of coding information of collocated CUs in the input and output video streams was exploited to reduce the complexity of CU and PU evaluations. At the CU level, two options for recursing through the split tree are considered, namely top (lower depths) to bottom (higher depths) or bottom to top. The top to bottom approach (T2B) accelerates the RD cost calculation of a CU by immediate splitting to smaller sizes or by early termination of the recursion, depending on the input CU structure. A more advanced method ($T2B_{ML}$) predicts the partition of a CU by using decision trees generated with machine learning techniques.

**Table 2.16:** Performance comparison with related works under VBR in terms of time saving (TS) and BDBR/TS (B/T)

| CFG | Optimization | Method | BDBR[%] | TS[%] | B/T |
|---|---|---|---|---|---|
| LP | CU | Shen [13] | 1.66 | 42 | 3.95 |
| | | Xiong [10] | 1.90 | 42 | 4.52 |
| | | Lee [30] | 1.22 | 61 | 2.00 |
| | | Ahn [31] | 1.00 | 43 | 2.33 |
| | | De Praeter [19] | 2.01 | 65 | 3.09 |
| | | **T2B** | **0.57** | **50** | **1.14** |
| | | **T2B$_{\mathbf{ML}}$** | **1.49** | **64** | **2.33** |
| | | **B2T** | **1.38** | **62** | **2.23** |
| | CU + PU | Liquan [11] | 1.15 | 41 | 2.80 |
| | | Liquan [32] | 0.88 | 52 | 1.69 |
| | | **T2B** | **1.32** | **64** | **2.06** |
| | | **T2B$_{\mathbf{ML}}$** | **2.23** | **76** | **2.93** |
| | | **B2T** | **2.14** | **76** | **2.82** |
| RA | CU | Correa [14] | 0.28 | 37 | 0.77 |
| | | Shen [13] | 1.40 | 45 | 3.11 |
| | | Xiong [10] | 2.21 | 40 | 5.53 |
| | | Lee [30] | 1.43 | 62 | 2.31 |
| | | Ahn [31] | 1.40 | 49 | 2.86 |
| | | **T2B** | **0.29** | **52** | **0.59** |
| | | **T2B$_{\mathbf{ML}}$** | **0.88** | **64** | **1.38** |
| | | **B2T** | **0.91** | **64** | **1.42** |
| | CU + PU | Correa [14] | 1.33 | 63 | 2.11 |
| | | Liquan [11] | 1.50 | 42 | 3.57 |
| | | Liquan [32] | 0.68 | 49 | 1.39 |
| | | **T2B** | **0.59** | **66** | **0.89** |
| | | **T2B$_{\mathbf{ML}}$** | **2.07** | **80** | **2.59** |
| | | **B2T** | **1.26** | **78** | **1.62** |

On the other hand, in the bottom to top approach (B2T), the CU structure of input CUs is re-used and recursively evaluated by merging sub-CUs into larger CUs. Additionally, the splitting behavior of neighboring CUs is also considered to reduce the number of RD evaluations in the B2T$_{\mathrm{TLP}}$ method.

Furthermore, during the PU evaluation process, the number of PU candidates is reduced by exploiting information from the input video stream. Experimental results show that the proposed transrating methods maintain coding efficiency of an unmodified cascaded decoder-encoder, while significantly reducing the transcoder complexity. On the PU evaluation level, B2T$_{\mathrm{TLP}}$ can reduce the complexity of

transrating by 82% with a bit rate increase by 1.95%. In addition, by considering all the proposed techniques, trade-offs between transrating complexity and coding performance can be made.

**The work described in this chapter led to the following publications:**

- **Luong Pham Van**, Johan De Praeter, Glenn Van Wallendael, Sebastiaan Van Leuven, Jan De Cock, Rik Van de Walle. Efficient bit rate transcoding for high efficiency video coding. *IEEE Transactions on Multimedia*. 2016. March 2016.
- **Luong Pham Van**, Jan De Cock, Glenn Van Wallendael, Sebastiaan Van Leuven, Rafael Rodriguez-Sanchez, Jose L. Martinez, Peter Lambert, Rik Van de Walle. Fast transrating for high efficiency video coding based on machine learning. In *Proceeding of the IEEE International Conference on Image Processing (ICIP)*. September 2013.

# References

[1] K. Seo, S. Lee, J. Kim, and J. Koh. *Rate Control Algorithm for Fast Bit-rate Conversion Transcoding*. IEEE Transactions on Consumer Electronics, 46(4):1128–1136, November 2000.

[2] P.A.A. Assuncao and M. Ghanbari. *A Frequency-domain Video Transcoder for Dynamic Bit-rate Reduction of MPEG-2 Bit Streams*. IEEE Transactions on Circuits and Systems for Video Technology, 8(8):953–967, December 1998.

[3] A. Eleftheriadis and P. Batra. *Dynamic Rate Shaping of Compressed Digital Video*. IEEE Transactions on Multimedia, 8(2):297–314, April 2006.

[4] D. Lefol, D. Bull, and N. Canagarajah. *Performance Evaluation of Transcoding Algorithms for H.264*. IEEE Transactions on Consumer Electronics, 52(1):215–222, February 2006.

[5] J. De Cock, S. Notebaert, P. Lambert, and R. Van de Walle. *Requantization Transcoding for H.264/AVC Video Coding*. Signal Processing: Image Communication, 25:235–254, April 2010.

[6] N. Hait and D. Malah. *Model-Based Transrating of H.264 Coded Video*. IEEE Transactions on Circuits and Systems for Video Technology, 19(8):1129–1142, August 2009.

[7] C. Deknudt, P. Corlay, A. S Bacquet, M. Zwingelstein-Colin, and F. Coudoux. *Reduced Complexity H.264/AVC Transrating Based on Frequency Selectivity for High-Definition Streams*. IEEE Transactions on Consumer Electronics, 56(4):2430–2437, November 2010.

[8] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) Standard*. IEEE Transactions on Circuits and Systems for Video Technology, 22(12):1649–1668, December 2012.

[9] L. Shen, Z. Zhang, and Z. Liu. *Effective CU Size Decision for HEVC Intracoding*. IEEE Transactions on Image Processing, 23(10):4232–4241, October 2014.

[10] J. Xiong, H. Li, Q. Wu, and F. Meng. *A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence*. IEEE Transactions on Multimedia, 16(2):559–564, February 2014.

[11] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang. *An Effective CU Size Decision Method for HEVC Encoders*. IEEE Transactions on Multimedia, 15(2):465–470, February 2013.

[12] Z. Pan, S. Kwong, M.-T. Sun, and J. Lei. *Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC*. IEEE Transactions on Broadcasting, 60(2):405–412, June 2014.

[13] X. Shen and L. Yu. *CU Splitting Early Termination based on Weighted SVM*. EURASIP Journal on Image and Video Processing, 2013:4, January 2013.

[14] G. Correa, P.A. Assuncao, L. Volcan Agostini, and L.A. da Silva Cruz. *Fast HEVC Encoding Decisions Using Data Mining*. IEEE Transactions on Circuits and Systems for Video Technology, 25(4):660–673, April 2015.

[15] D. Lefol, D. Bull, and N. Canagarajah. *Mode Refinement Algorithm for H.264 Intra Frame Requantization*. In IEEE International Symposium on Circuits and Systems (ISCAS), October 2006.

[16] D. Lefol and D. Bull. *Mode Refinement Algorithm for H.264 Inter Frame Requantization*. In IEEE International Conference on Image Processing (ICIP), pages 845–848, October 2006.

[17] T. Shanableh, E. Peixoto, and E. Izquierdo. *MPEG-2 to HEVC Video Transcoding with Content-based Modeling*. IEEE Transactions on Circuits and Systems for Video Technology, 23(7):1191–1196, July 2013.

[18] E. Peixoto, T. Shanableh, and E. Izquierdo. *H.264/AVC to HEVC Video Transcoder Based on Dynamic Thresholding and Content Modeling*. IEEE Transactions on Circuits and Systems for Video Technology, 24(1):99–112, January 2014.

[19] J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle. *Efficient Transcoding for Spatially Misaligned Compositions for HEVC*. In IEEE International Conference on Image Processing (ICIP), pages 2487–2491, October 2014.

[20] G. Van Wallendael, J. De Cock, and R. Van de Walle. *Fast Transcoding for Video Delivery by Means of a Control Stream*. In IEEE International Conference on Image Processing (ICIP), pages 733–736, September 2012.

[21] ISO/IEC JTC1/SC29/WG11. *Requirements of the Scalable Enhancement of HEVC*. w12956, Geneva, CH, July 2012.

[22] I. Kim, K. McCann, K. Sugimoto, B. Bross, and W. Han. *High Efficiency Video Coding (HEVC) Test Model 7 Encoder Description*. JCTVC-1002, Geneva, CH, May 2012.

[23] F. Bossen. *Common Test Conditions and Software Reference Configurations*. JCTVC-I1100, Geneva, CH, May 2012.

[24] G. J. Sullivan and T. Wiegand. *Rate-distortion optimization for video compression*. IEEE Signal Processing Magazine, 15(6):74–90, November 1998.

[25] I. H. Witten and F. Eibe. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[26] J. R. Quinlan. *Programs for Machine Learning*. Morgan Kaufmann, 1993.

[27] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. document VCEG-M33 of ITU-T Video Coding Experts Group (VCEG), April 2001.

[28] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, New York, 1992.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. *Image Quality Assessment: From Error Visibility to Structural Similarity*. IEEE Transactions on Image Processing, 13(4):600–612, April 2004.

[30] J. Lee, S. Kim, K. Lim, and S. Lee. *A Fast CU Size Decision Algorithm for HEVC*. IEEE Transactions on Circuits and Systems for Video Technology, 25(3):411–421, March 2015.

[31] S. Ahn, B. Lee, and M. Kim. *A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding*. IEEE Transactions on Circuits and Systems for Video Technology, 25(3):422–435, March 2015.

[32] L. Shen, Z. Zhang, and Z. Liu. *Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations*. IEEE Transactions on Circuits and Systems for Video Technology, 24(10):1709–1722, October 2014.

# 3

# Efficient motion estimation for HEVC transrating

## 3.1 Rationale and related work

In the previous chapter, the complexity of an HEVC bit rate transcoder is reduced by optimizing the block partitioning process of the encoder. Various techniques have been proposed to utilize the correlation between the coding information of the input and output video; thus, the number of evaluated coding units and prediction units is reduced. However, the motion estimation (ME) process has not been optimized during the coding mode evaluation. It has been well-known that motion estimation is the most time-consumed part of an encoder. According to our extensive observation in Section 3.2, this process accounts for a large portion of 62 percent of the HEVC encoding time. Logically, motion estimation is highly considered when optimizing an encoder. This chapter further upgrades the bit rate transcoding process by improving the motion estimation process in the encoding part.

Many research activities have been focusing on optimizing the motion estimation process of a transcoder. Most of these efforts derive a base motion vector for the output block from the motion vectors of collocated blocks in the input video. Then, a refinement is applied to the base motion vector to enhance it's quality. In [1], a search window of $[\pm 3, \pm 3]$ pixels was used to refine the base motion vector; and in most cases, a smaller search window of $[\pm 1, \pm 1]$ pixels was mentioned to generate satisfying results. An adaptive motion refinement scheme was pro-

posed for frame-rate conversions in [2]. Firstly, a base motion vector was obtained by using a so-called forward dominant vector selection method. Then, the motion vector might be refined with a delta motion vector in use of the sum of absolute differences (SAD) of pixels in the reconstructed block and the current block. This refinement uses a search window of $[\pm 2, \pm 2]$ pixels.

More recently, Peixoto et al. suggested a reusing motion vector technique for a transcoder from H.264/AVC to HEVC [3]. First, the integer motion vector of prediction units (PUs) in the HEVC stream is derived by either evaluating the collocated H.264/AVC motion vector that covers the largest area within the CU or evaluating all collocated H.264/AVC motion vectors. The output motion vector is not refined further at the integer pixel level. Finally, the default HEVC sub-pixel search is performed.

These aforesaid motion refinement approaches owe its drawback to the fact that the base motion vector was refined at the integer pixel level with the use of a small fixed-size window. Using this small window may ignore the optimal motion vector, especially when the video features a high motion activity, or upon significantly different characteristics (bit rate and/or resolution) between the input and output videos.

The HEVC reference software (HM) [4] adopts Test Zone Search (TZSearch) as the default fast integer ME algorithm. TZSearch applies either a diamond-shaped pattern or a square-shaped pattern to search for the optimal integer motion vector. This search process launches at the initial search point. Then a raster search might also be carried out. At the end of TZSearch, the motion vector is conditionally improved by a simple raster or star refinement. Our extensive observation shows that TZSearch accounts for about 22% of the total encoding time. Although the complexity of TZSearch is significantly lower compared to a full search that evaluates any motion vector within a pre-defined window, this can be further optimized by adapting the search area and the search pattern.

In this chapter, we further improve our transrating scheme presented previously by employing the correlation of the input and output motion vectors. This ensures a reduction of the complexity for TZSearch in the HEVC encoder. In general, the initial search point is adaptively selected first. Then, based on the rate-distortion (RD) cost yielded by encoding the corresponding PU using this starting point, the search area is decided using an online-trained Bayes decision rule. Finally, the integer motion vector is searched on either of two fast search schemes for each type of search area. It is believed that the proposed technique can solve the mentioned problems of existing motion vector refinement approaches for a cascaded pixel domain transcoder.

The remainder of this chapter is organized as follows: The time consumption of motion estimation in the HEVC encoder is analysed in Section 3.2. An overview of TZSearch is given in Section 3.3; the correlation of input and output motion vec-

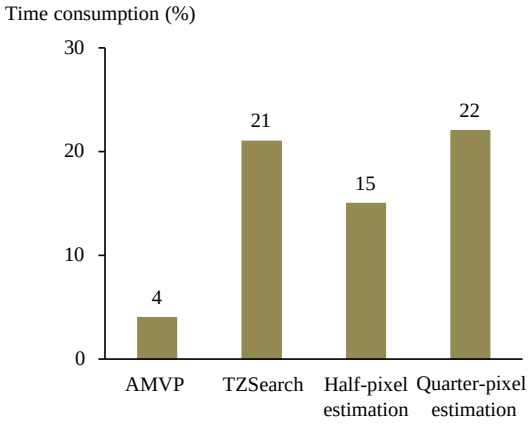



**Figure 3.1:** Time consumption distribution of the motion estimation in the HEVC encoder.

tors is analyzed in Section 3.4; thereafter, the proposed TZSearch improvement is presented in Section 3.5; experimental results of the proposed technique on motion estimation are shown in Section 3.6.

## 3.2 Motion estimation complexity analysis

In this experiment, the low-delay configuration is used. The prediction structure is set IPPP with the number of reference frames of 4. The quantization parameter is set $\{22, 27, 32, 37\}$. Four sequences with different motion activities ranging from low to high (FourPeople, ParkScene, BQMall, and BasketballDrill) are encoded using the reference software HM 7 [5]. The experiment indicates that motion estimation is the most complex part of the encoder with a consumption of 62% encoding time.

The motion estimation process in HEVC includes four steps. The first step is advanced motion vector prediction (AVMP). The output of the AVMP process is the predictor motion vector of a PU. Then, the optimal integer motion vector is obtained. Currently, TZSearch is used in the reference software for the integer motion estimation. Finally, the optimal integer motion vector is then refined at the half-pixel accuracy followed by a refinement at quater-pixel accuracy. The complexity of these four steps in terms of time consumption is depicted in Figure 3.1.

As we can see from Figure 3.1, the search for the predicted motion vector is notably less time consuming than the other parts with about 4% total time encoding. The TZSearch and quarter-pixel refinement processes are the most time consuming components that cost about 22% total encoding time. Since the number of inpthe half-pixel motion estimation

## 3.3 Test Zone Search algorithm

Test Zone Search (TZSearch) algorithm has been implemented in the HEVC reference software (HM) [4] as a fast tool to find the integer motion vector of a prediction unit. Generally, this technique consists of three processes, including first search, raster search, and refinement.

### 3.3.1 First search

The goal of this step is to estimate the global minimum point. This makes use of either a diamond search or square search with different stride lengths from 1 through the search window size in multiples of 2. Currently, the diamond search (Figure 3.2) is adopted as default in the reference software. A diamond search for a search window size of 64 results in 6 grids with 8 inspection points and 1 grid with 4 inspection points. In total, the RD cost of 52 (6x8+4) search points are obtained using Equation 3.1. The motion vector of the minimal RD cost point is selected. The distance between (best distance) this motion vector and the predictor is used to decide whether this motion vector is further refined. If this distance is zero, the predictor motion vector is assumed to be the global optimum. Therefore, a further refinement is unnecessary and the search is stopped. Otherwise, this output is then refined further in the next steps.

$$\begin{cases} RD(MV) = SAD(O,P) + \lambda.R(MV - pMV) \\ SAD(O,P) = \sum_{x=1}^{X} \sum_{y=1}^{Y} |O(x,y) - P(x,y)| \end{cases} \tag{3.1}$$

where, SAD is the sum of absolute difference between the original PU ($O$), and its predicted PU ($P$), at the position located by the motion vector in the reference frame; $\lambda$ indicates the Lagrange multiplier; $pMV$ is the prediction motion vector that is obtained by the advanced motion vector prediction (AMVP) process; *R(.)* represents the number of bits for encoding the motion vector difference; $X$ and $Y$ are the width and height of the PU, respectively.

### 3.3.2 Raster search

The raster search is only carried out if the best distance obtained from the first search step is larger than a predefined value. This predefined raster value can be customized in order to provide a trade-off between the complexity of TZSeach and coding performance. Currently, this value is set 5 in the default configuration file.

The raster seach scans thorough whole search window by a stride of the predefined raster value. Figure 3.3 illustrates the raster search with a stride of 5.

It is important to note that TZSearch is conditionally performed. Choosing the initial search point being near the global optimum results in a reduction the

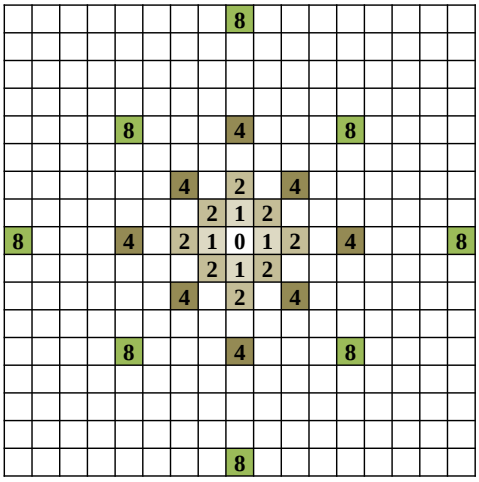



**Figure 3.2:** Diamond search pattern.

best distance. Therefore, improving the performance of selecting the initial search point would avoid the raster search, thus a reduction of TZSearch complexity is made. This is a motivation of re-defining the initial search point.

### 3.3.3 Raster/star refinements

The refinement is performed when the distance between the predictor and the output of first search (or of raster search, if raster search is invoked) is non zero. Either a square search or diamond search is carried out. In the default configuration, diamond search is enabled. The number of evaluated points in the refinement is limited to 8.

## 3.4 Analysis of input motion information

Due to the fact that the motion vector of a block in the output video and those of the collocated blocks in the input video reflect the motion of the same object, they are highly correlated. This section evaluates this correlation to figure out which representation of input motion vectors mostly correlates with the motion vector of an output block. To do this, the *RD* costs of points in a $[5, 5]$ window around an input motion vector are measured. Then, the probability that the motion vector with minimal *RD* cost (the global optimization point) located in the observed window is calculated.

In the experiment, four sequences with an activity ranging from low to high have been selected. These can be categorized into low motion and low complexity

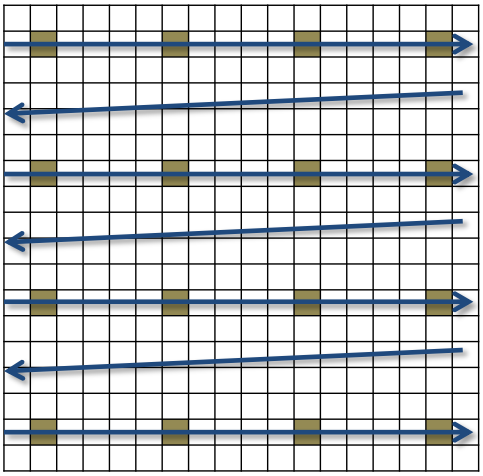


**Figure 3.3:** Raster search pattern with a stride of 5.

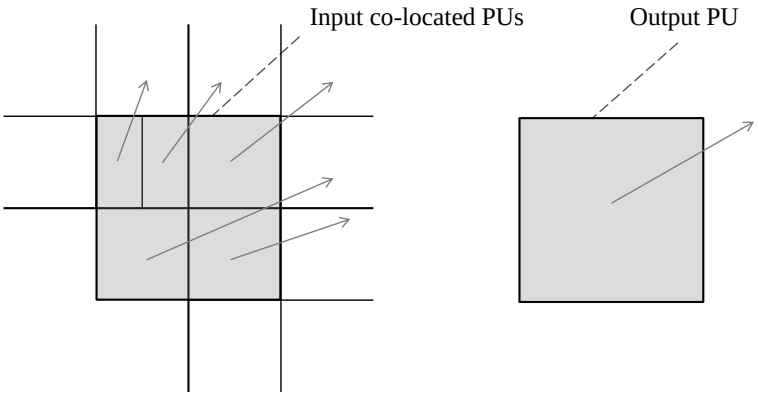



**Figure 3.4:** An output PU associates with multiple input PUs.

(*FourPeople*), medium activity (*ParkScene*, and *BQSquare*) and high motion (*BasketballDrill*). These sequences have been encoded using a quantization parameter (QP) of 32. Then, the bitstreams were decoded and re-encoded using a higher QP of 36.

Since an output prediction unit (PU) may overlap several input PUs as illustrated in Figure 3.4, the motion of this output PU may correspond to multiple input motion vectors. Therefore, we present these multiple input motion vectors in a single representation. This presentation can be either the weighted mean motion vector (*wmMV*) or the median motion vector (*meMV*) of the input motion vectors.

The *wmMV* and *meMV* vectors are defined by Equation 3.2 and Equation 3.3.

$$wmMV = \frac{\sum_{i=1}^{N} w_i MV_i}{\sum_{i=1}^{N} w_i} \tag{3.2}$$

$$meMV = Median\{MV_1, ..., MV_N\} \tag{3.3}$$

where N is the number of the input collocated PUs. $w_i$ and $MV_i$ represent the number of pixels and the motion vector of $ith$ PU in the input collocated PU set.

In addition, this single motion vector can be the input motion vector with minimum *RD* cost (the best input motion vector - *biMV*) defined by Equation 3.4 (*S* is the set of input motion vectors).

$$biMV = \arg\min_{x \in S} RD(x) \tag{3.4}$$

Observing the probability distributions shows that *biMV* demonstrates a better coding performance than other instances of the input motion vectors. In terms of *RD* cost per pixel, the mean of *RD(biMV)* is smaller than that of *RD(wmMV)* and *RD(meMV)*. For instance, the mean of *RD(biMV)* cost is 13.89 while these of *wmMV* and *meMV* are 17.73 and 17.46, respectively. There is a higher probability where the global optimization point is located in the $[5, 5]$ window that centered on *biMV* compared to windows centered on *wmMV* or *meMV* (Figure 3.5). In particular, there is a 29% probability that the output motion vector equals biMV. High probability of *biMV* being the output motion vector demonstrates that *biMV* makes the best initial motion vector for motion estimation in transrating.

The inferior performance of using *wmMV* or *meMV* became clear when the output PU corresponds with a lot of smaller PUs in the collocated input area. Certainly when this number of input motion vectors got high and divergent, *wmMV* and *meMV* were no longer accurate in describing the motion of output PUs.

## 3.5 Proposed fast TZSearch algorithm

The flow chart of the proposed integer motion estimation algorithm is illustrated in Figure 3.6. The proposed algorithm includes four steps.

**Base motion vector selection:** The base motion vector (*bMV*) selection is performed to improve the quality of the initial point. Base on the characteristics of the base motion vector, a search pattern is selected for the first search step. The selection is described as follow.

**Search pattern selection:** Three search patterns include an evaluation of 4 points around (*bMV*), fast three-step search, and triangle search are proposed.

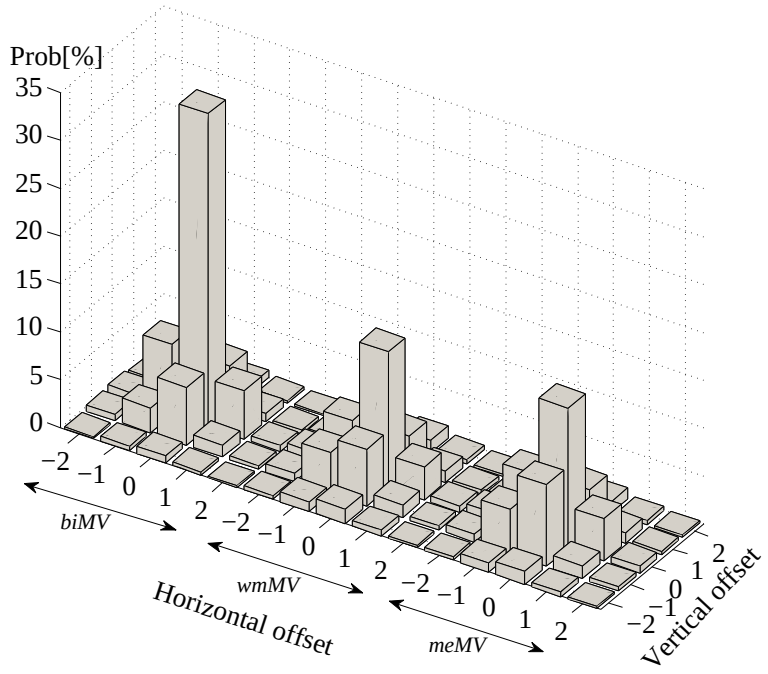



**Figure 3.5:** Distribution of output motion vector in 5x5 windows centered around $biMV$, $wmMV$, and $meMV$. The output motion vector is mostly located around $biMV$.

When the best input motion vector and the predictor are identical, only four point around the predictor are evaluated. Otherwise, the search range is classified into a small area ($[5, 5]$ window) or a larger window. The classification is a Baye's decision rule which is obtained by an online training process. For the small window, F3SS is deployed while triangle search is used for a large window search.

**First search:** The first search estimates the global optimum using of three proposed search patterns. After the search pattern is decided, the first search is performed. The output of the first search is considered for the raster search and refinement.

**Raster search:** When the 4-point search pattern or F3SS is used in the first search, the raster search is skipped. Otherwise, the distance between the output and the predictor is checked to decide whether the raster search is performed. If this distance is larger than the predefined raster value, raster search is invoked.

**Refinement:** The refinement step is left unchanged compared to TZSearch.

These five steps are discussed in more details in the following sections. In the flowchart (Figure 3.6), an online training is performed during encoding the first *N* frames of each sequence to obtain the threshold which is used to select search pattern. It should be noted that during the training phase, the un-optimized

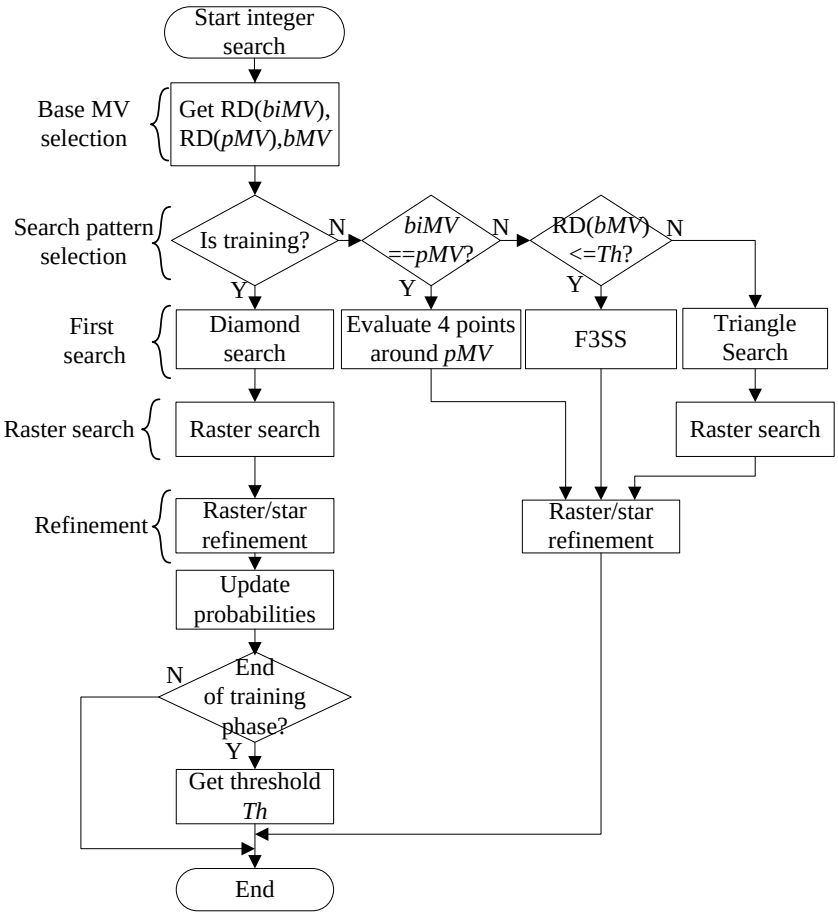


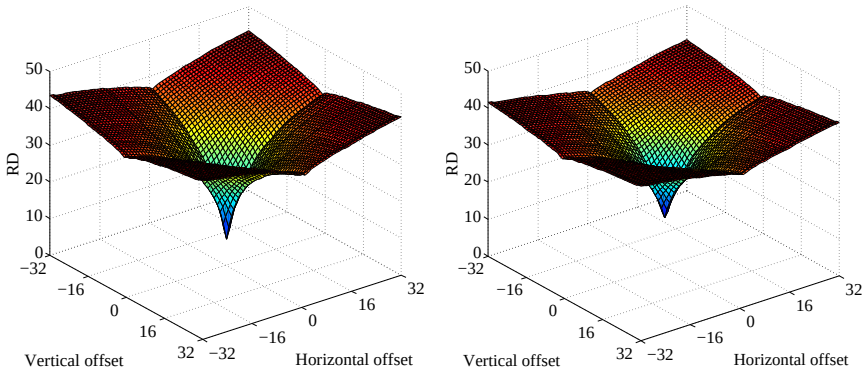

**Figure 3.6:** Flowchart of the proposed fast TZSearch algorithm for transrating. The raster search is conditionally invoked when the between the output of first search and $bMV$ is larger than the predefined raster value. The refinement is performed when this distance is non zero.

TZSearch algorithm is performed.

### 3.5.1 Base motion vector selection

**Selection of the base motion vector**

In this step, the *bMV* motion vector is derived. The best motion vector (*biMV*) of collocated input motion vector is obtained using the criteria defined by Equation 3.4. From the candidate set of *pMV* and *biMV*, the base motion vector *bMV* is selected as in Equation 3.5. The unselected motion vector is stored as a reference




(a) *biMV* and *pMV* share the same value. (b) *biMV* and *pMV* are different.

**Figure 3.7:** RD costs of the base motion vector (0, 0) and motion vectors that differ from this vector by various offsets.

motion vector (*rMV*) to support the triangle search in the first search step.

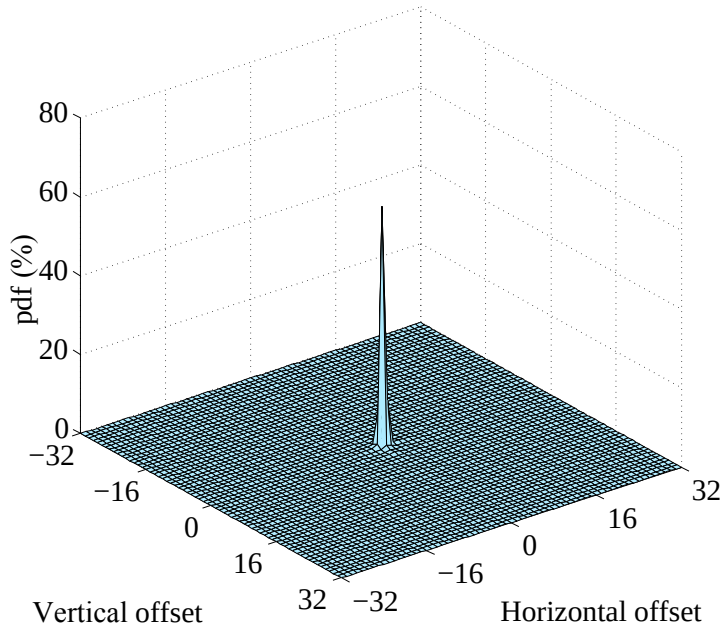
$$bMV = \underset{x \in \{biMV, pMV\}}{\arg\min} RD(x) \tag{3.5}$$

**Evaluation of the base motion vector**

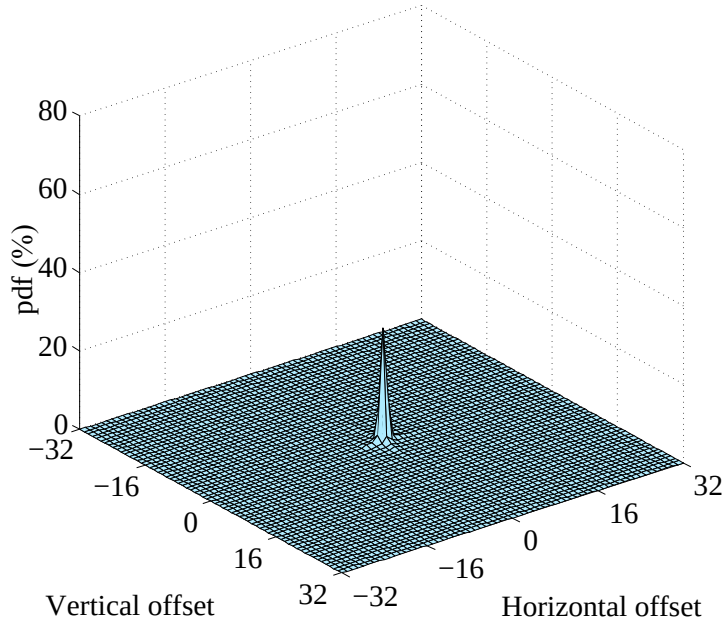
In order to evaluate the quality of the base motion vector, the RD cost and the probability that this motion vector is the optimal integer motion vector of an output block have been measured. These values are compared to that of other motion vectors which differ from the base motion vector by various offsets. Four sequences with the transrating scheme presented in Section 3.4 have been evaluated. Experimental results are shown in Figure 3.10 and Figure 3.8.

The RD cost surface in Figure 3.10 demonstrates that the base motion vector provides the best rate-distortion performance relative to other points. This means that the base motion vector has a high correlation with the global optimization point. Moreover, the *RD* cost of the base motion vector of the case *biMV* and *pMV* are identical is smaller than that when *biMV* and *pMV* are different. On the other hand, the base motion vector and the global optimization point have a higher correlation in case *biMV* and *pMV* share the same value. This correlation is depicted by the distribution surface in Figure 3.8.

As can be seen from Figure 3.8, there is a notable probability that the base motion vector is the output integer motion vector. When *biMV* matches *pMV*, this probability is measured at a value of 65% on average. In addition, there is an 85% probability that the best integer motion vector is the base motion vector or 4 points around the base motion vector towards the horizontal or vertical directions. On the




(a) *biMV* and *pMV* share the same value.




(b) *biMV* and *pMV* are different.

**Figure 3.8:** Probability that a motion vector is the output integer motion vector of a block. (0, 0) is the base motion vector.

other hand, when *biMV* and *pMV* are different, these probabilities become lower with 30% and 50%, respectively.

This analysis implies that when *biMV* and *pMV* are the same, only a slight refinement for the base motion vector is required for a satisfying performance. Otherwise, a larger search area should be used to refine the base motion vector. This conclusion is exploited in the search pattern selection of the proposed approach.

### 3.5.2 Search pattern selection

Search area selection has been motivated by the fact that when the RD cost of encoding the prediction unit using *bMV* (*R(bMV)*) is small enough, it may be considered as the global optimization point. Therefore, the refinement window can be small. On the other hand, a high RD cost indicates this point is a local optimum. Consequently, the global optimum should be re-evaluated in a larger window. The appropriate window size is selected by evaluating two criteria as visualized in the 'search pattern selection' part of Figure 3.6.

- When *biMV* and *pMV* share a value, both the input video stream and the spatial information concur on the best MV. As measured in Section 3.5.1, in this case, there is a significant probability of 85% that the base motion vector or 4 points around this motion vector are the optimal integer motion vector. Therefore, only four points around this initial motion vector are evaluated.

- When *biMV* and *pMV* do not concur, further search in a larger window is performed. However, the size of the window should be adaptively determined such that the complexity of searching is lowered while retaining coding performance. The window size is derived upon a threshold (*Th*) of *RD* cost of *bMV*. The threshold is derived under a Bayes decision rule relying on the RD cost of *bMV* during an online training phase after which the search window of a *bMV* is decided upon this threshold. If the *RD* of *bMV* is smaller than *Th*, the search area is limited to a $[5, 5]$ window; otherwise, a larger window is used. In the following section, the derivation of *Th* is interpreted.

In order to early determine the search area for a given *bMV*, a two-class classification problem is defined with a $[5, 5]$ window and a larger window, $C_1$ and $C_2$. In other words, $C_1$ and $C_2$ are the events that output whether a motion vector is inside the $[5, 5]$ window or outside this window, respectively. The *RD(bMV)* is obtained to determine which of the two classes should be assigned to the *bMV*. Using Bayes' theorem, the posterior probabilities of classification are given as

$$p(C_k|R) = \frac{p(R|C_k)p(C_k)}{p(R)}. \tag{3.6}$$

**Table 3.1:** Probabilities for determining *Th*

| Name | Meaning |
|---|---|
| $p(C_k)$ | Probability of event $C_k$ |
| $p(R)$ | Probability of quantized $RD(bMV)$ |
| $p(R\|C_k)$ | Probability of quantized $RD(bMV)$given $C_k$ |

Herein $p(R|C_k)$, $p(R)$, and $p(C_k)$ are defined in Table 3.1 and $R = RD(bMV)$.

To minimize misclassification rate, each value of ***RD(bMV)*** is assigned to the class for which the posterior probability $p(C_k|R)$ is largest [6]. As shown in Figure 3.9(b), *Th* is the point for which $p(C_1|R) = p(C_2|R)$. Since $RD(bMV)$ is real and positive, it is quantized by a step of 1. Therefore, the condition $p(C_1|R) = p(C_2|R)$ may no longer be valid. To solve this problem, the final formula for *Th* is expressed as

$$Th = \arg\min_{r} |P(C_1|R = r) - P(C_2|R = r)| \tag{3.7}$$

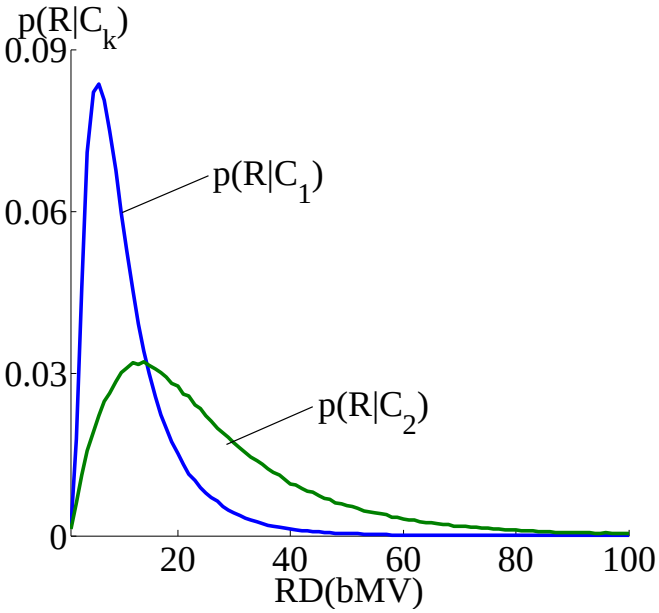
The distribution of $RD(bMV)$ significantly depends on the quantization parameter, the CU size and video content. To ensure their successful adaptation to different block sizes, a threshold is calculated for every CU size independently, i.e., four thresholds in total. The probabilities to determining the thresholds are obtained through an online training phase. The training phase happens during the encoding of the first *N* frames of video sequence. During the training phase, a diamond search is applied as in TZSearch. At the end of the training phase, the necessary probabilities for calculating *Th* are generated. *Th* is then calculated using Equation 3.7. From then on, the search range of PUs in the remaining frames is decided using this *Th*.

### 3.5.3 Proposed fast search patterns

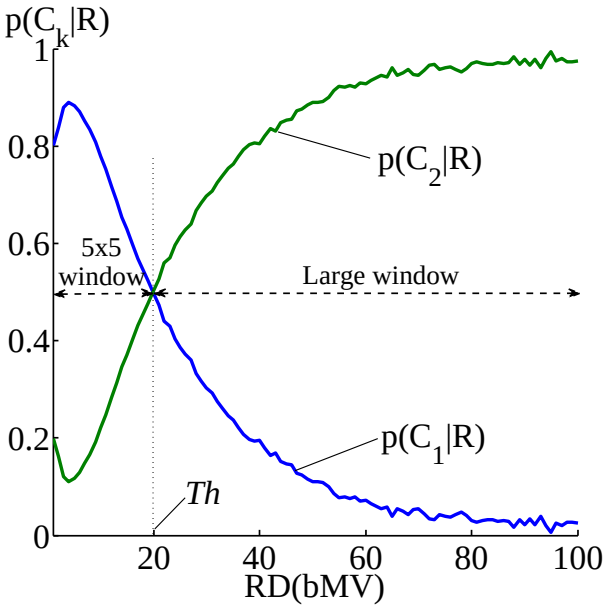
Two search patterns are proposed for each type of search area class. For the window of $[5, 5]$, a fast three steps search (F3SS) with early termination of refinement is applied. A larger window encounters the adoption of a triangle search derived from the diamond pattern.

**Fast three steps search**

The proposed F3SS works on the assumption that the motion estimation matching error decreases monotonically as the search moves towards the position of the global minimum error [7]. F3SS carries out three search iterations starting from

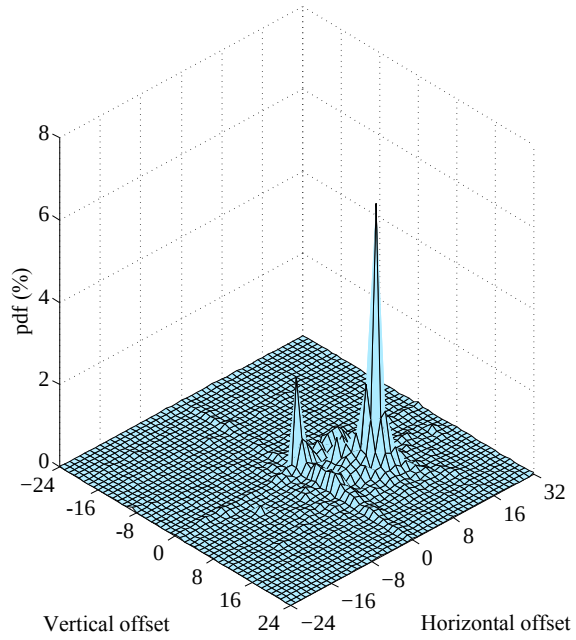



(a) Probability of $RD(bMV)$ given $C_k$.

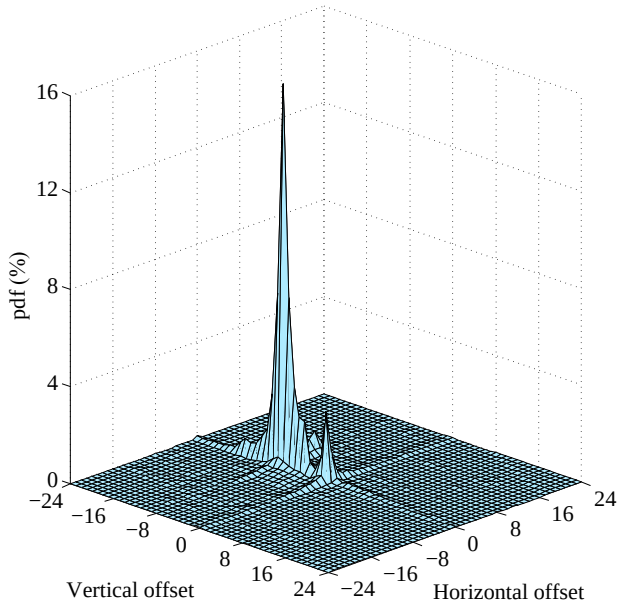



(b) Posterior probability of $C_k$.

**Figure 3.9:** Probabilities used for determining *Th*, QP = 27, ΔQP = 4, CU size 16x16.

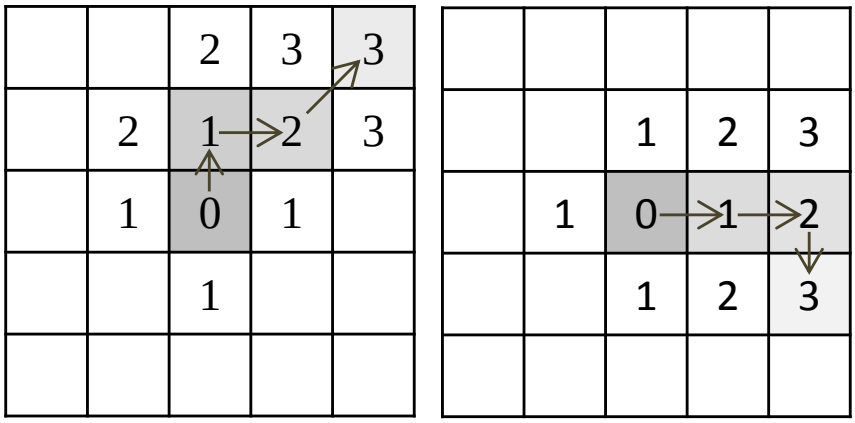



(a) *biMV* and *pMV* are different by (8,4).



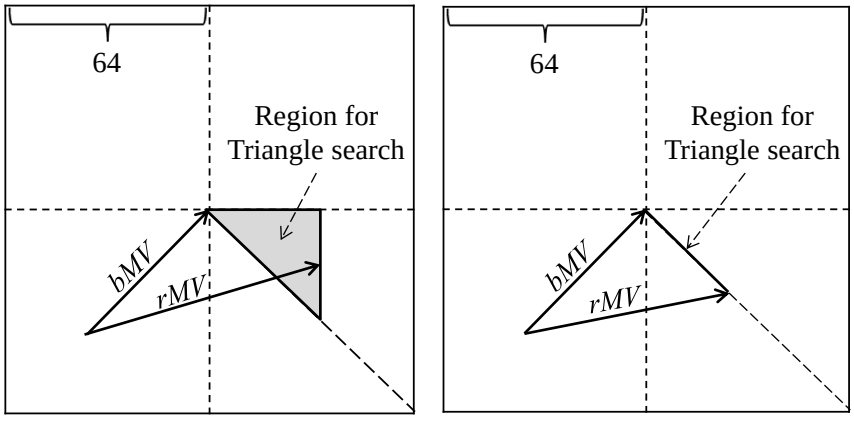

(b) *biMV* and *pMV* are different by (-16, 0).

**Figure 3.10:** Distribution of the output motion vector when *biMV* and *pMV* are different.




(a) 10 points are tested. (b) 9 points are tested.

**Figure 3.11:** Examples of F3SS. The number indicates the search round. Colored points are roots.




(a) $rMV$ is not on a line of diamond pattern. (b) $rMV$ is on a line of diamond pattern.

**Figure 3.12:** Triangle search algorithm.

*bMV* (see for example Figure 3.11). The root of the current iteration is the best point of the previous iteration. During an iteration, the pending evaluated neighboring points (top, left, bottom, right) are considered. Additionally, the third iteration will experience the evaluation of the corner of the $[5, 5]$ window if applicable. An iteration is terminated if the best *RD* cost of this iteration is lower than the *RD* cost of its root.

**Triangle search**

Our analysis has shown that the distribution of the output motion vector is directed towards the *rMV* (which was obtained during base motion vector selection). More specifically, the output motion vector is mostly located between *bMV* and *rMV*. Therefore, the diamond search is adapted to a search range ($SR_T$) limited to the triangle as illustrated in Figure 3.12. However, when *rMV* lies on a line of the diamond pattern, only this line is evaluated. The search range is limited as in Equation 3.8 where the constant of 3 ensures that the search size is larger than 2.

$$SR_T = \max\{3, |bMV_x - rMV_x|, |bMV_y - rMV_y|\} \tag{3.8}$$

## 3.6 Experimental results

The proposed fast motion estimation (PFME) has been built on the top of the previously presented $\text{B2T}_{\text{TLP\_PU}}$ work [8] which started from HEVC reference software v7.0 [4]. $\text{B2T}_{\text{TLP\_PU}}$ reduces the computational complexity of CU and PU partitioning while the motion estimation is optimized by the proposed TZSearch algorithm. In order to evaluate the performance of PFME, a performance comparison between the proposed technique and four other motion estimation strategies have been made: RWM, RMD, $\text{WM}_{\text{Refine}}$, and $\text{MD}_{\text{Refine}}$. The trivial RWM and RMD methods directly use the weighted mean motion vector (*wmMV*) and the median motion vector (*meMV*) of the input motion vectors as the optimal integer motion vector without a refinement in the integer level. On the other hand, $\text{WM}_{\text{Refine}}$ and $\text{MD}_{\text{Refine}}$ respectively refine wmMV and meMV vectors by using the proposed adaptive search method to obtain the optimal integer motion vector. PFME, $\text{WM}_{\text{Refine}}$, and $\text{MD}_{\text{Refine}}$ differ for their base motion vector: PFME uses the best input motion vector (*biMV*) as the base motion vector while the other methods use *wmMV* and *meMV*.

Performance results are based on the low-delay coding conditions as defined in [5]. Input $QP_1$ was chosen from $\{22, 27, 32, 37\}$. Output $QP_2$ is higher than $QP_1$ by a $\Delta$QP with $\Delta$QP = 2, 4, 6. All sequences of class B, C, D, and E (16 sequences in total) have been tested. The performance of the aforementioned algorithms is evaluated by comparing with $\text{B2T}_{\text{TLP\_PU}}$ and an unmodified cascaded decoder encoder (*Ref*) in terms of transrating time saving and Bjøntegaard Delta

Bit rate (BDBR) [9]. The time saving term is defined as in Equation 3.9. In the BDBR measurement, Peak Signal to Noise Ratio (PSNR) calculations between the re-encoded and the original sequence are used. Furthermore, the first 10 frames are used for the Bayesian training phase of the proposed PFME strategy. Detailed results of the RWM and PFME are shown in Table 3.2, while the comparison results are presented in Table 3.3.

$$TS(\%) = \frac{T_{Original}\ (ms) - T_{Proposed}\ (ms)}{T_{Original}\ (ms)} \tag{3.9}$$

**Table 3.2:** Performance results of the proposed motion estimation scheme

| Method | Class | BDBR increase[%] | | | Time Saving(TS)[%] | | | Average | |
|---|---|---|---|---|---|---|---|---|---|
| | | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | ΔQP = 2 | ΔQP = 4 | ΔQP = 6 | BDBR | TS |
| RwM | B | 1.86 | 3.35 | 4.62 | 84.59 | 83.92 | 83.42 | 3.28 | 83.98 |
| | C | 1.62 | 3.57 | 5.04 | 83.06 | 82.79 | 81.86 | 3.41 | 82.57 |
| | D | 1.72 | 3.42 | 4.65 | 80.66 | 79.46 | 77.88 | 3.26 | 79.33 |
| | E | 1.87 | 2.99 | 3.83 | 87.50 | 88.07 | 87.69 | 2.90 | 87.76 |
| | Average | **1.76** | **3.36** | **4.58** | **84.73** | **84.36** | **83.82** | **3.23** | **84.30** |
| Proposed | B | 1.65 | 2.85 | 3.76 | 83.25 | 82.56 | 81.92 | 2.75 | 82.58 |
| | C | 1.30 | 2.77 | 3.81 | 81.65 | 81.17 | 80.13 | 2.63 | 80.98 |
| | D | 1.25 | 2.51 | 3.34 | 79.41 | 78.17 | 75.99 | 2.37 | 77.85 |
| | E | 1.61 | 2.49 | 3.02 | 87.15 | 87.78 | 87.43 | 2.37 | 87.45 |
| | Average | **1.44** | **2.67** | **3.51** | **83.54** | **83.16** | **82.50** | **2.54** | **83.06** |

**Table 3.3:** Average performance results of different motion estimation strategies

| Method | Compare with Anchor | | Compare with $B2T_{TLP\_PU}$ | |
|---|---|---|---|---|
| | BDBR | TS | BDBR | TS |
| RWM | 3.23 | 84.30 | 0.58 | 22.00 |
| RMD | 3.13 | 84.08 | 0.48 | 21.42 |
| $WM_{Refine}$ | 2.78 | 83.11 | 0.13 | 16.71 |
| $MD_{Refine}$ | 2.73 | 82.91 | 0.08 | 15.76 |
| PFME | 2.54 | 83.06 | -0.11 | 16.45 |
| $B2T_{TLP\_PU}$ | 2.65 | 79.74 | 0.00 | 0.00 |

As can be seen from Table 3.2, compared to the state-of-the-art $B2T_{TLP\_PU}$ technique, the proposed method outperforms in terms of both coding performance and computational complexity reduction. In terms of bit rate, a 0.11% reduction was reported. This advance results from the adaptive selection of the starting point in TZSearch. In terms of transrating complexity, the proposed method achieves a 16.40% reduction by early determining the search area combined with two proposed fast search strategies for each search area. Overall, the proposed transrating scheme significantly reduces the complexity by 83.06% with a slight increase of bit rate (2.54%) compared to the unmodified cascade of decoder and encoder.

Experimental results in Table 3.3 show that RWM and RMD achieve similar performances at about 22% complexity along with bit rate increases by about 0.58% and 0.48% compared to the $B2T_{TLP\_PU}$ technique. In use of the proposed refinement, these bit rate penalties are reduced to 0.13% and 0.08% for WMRefine and MDRefine, respectively. Among these fast motion estimation strategies, the proposed PFME technique demonstrates best coding performance. PFME is able to achieve a similar complexity reduction of WMRefine and MDRefine (about 16%). In terms of bit rate, PFME can reduce the bit rate of the output video by 0.11% compared to the $B2T_{TLP\_PU}$ approach.

The proposed motion estimation approach outperforms two fast motion estimation algorithms presented in [10, 11]. The Pan's algorithm is able to reduce the complexity of the encoder by 15% with a bit rate penalty of 0.55%. On the other hand, the Yang approach achieves a complexity reduction of 11% with a negligible bit rate reduction of 0.01%. Obviously, the proposed technique is superior in terms of both the complexity reduction (16%) and the bit rate performance (-0.11%).

## 3.7 Conclusions and future works

In this chapter, the transrating scheme of HEVC video has been further improved by optimizing the motion estimation step. A fast TZSearch algorithm is proposed by utilizing the correlation between the motion information of the input and output

video stream. The initial point for the proposed search algorithm is adaptively selected from the input motion vectors and the median predictor while the search area is determined using self-learning Bayes decision rules. Two fast search schemes are proposed for each search area. Experimental results show that the proposed algorithm can achieve a good coding performance in terms of RD performance and complexity reduction for transrating. Together with the proposed fast coding block partitioning techniques, the proposed fast motion estimation scheme can reduce the computational complexity of the HEVC transrater by 83% with only 2.54% bit rate penalty.

The proposed fast TZSearch has been applied in the transcoding context. However, it can be deployed for optimizing the motion estimation process of the HEVC encoder. For instance, based on the rate-distortion cost, the search window can be adaptively selected by using the proposed search pattern selection. Furthermore, two fast search patterns would be used to derive the integer motion vector in a small window. In a large window, a diamond search would be applied.

**The work described in this chapter led to the following publication:**

- **Luong Pham Van**, Jan De Cock, Antonio Jesus Diaz-Honrubia, Glenn Van Wallendael, Sebastiaan Van Leuven, Rik Van de Walle. Fast motion estimation for closed-loop HEVC transrating. In *Proceeding of the IEEE International Conference on Image Processing (ICIP)*. October 2014.

# References

[1] N. Bjork and C. Christopoulos. *Transcoder architectures for video coding*. IEEE Transactions on Consumer Electronics, 44(1):88–98, February 1998.

[2] J. Youn, M.-T. Sun, and C.-W. Lin. *Motion vector refinement for high-performance transcoding*. IEEE Transactions on Multimedia, 1(1):30–40, March 1999.

[3] E. Peixoto and E. Izquierdo. *A Complexity-Scalable Transcoder From H.264/AVC to the New HEVC Codec*. In IEEE International Conference on Image Processing (ICIP), pages 737–740, September 2012.

[4] I. Kim, K. McCann, K. Sugimoto, B. Bross, and W. Han. *High Efficiency Video Coding (HEVC) Test Model 7 Encoder Description*. JCTVC-1002, Geneva, CH, May 2012.

[5] F. Bossen. *Common Test Conditions and Software Reference Configurations*. JCTVC-I1100, Geneva, CH, May 2012.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[7] J. Jain and A. Jain. *Displacement Measurement and Its Application in Interframe Image Coding*. IEEE Transactions on Communications, 29(12):1799–1808, Dec. 1981.

[8] L. Pham Van, J. De Praeter, G. Van Wallendael, S. Van Leuven, J. De Cock, and R. Van de Walle. *Efficient Bit Rate Transcoding for High Efficiency Video Coding*. IEEE Transactions on Multimedia, 18(3):364–378, March 2016.

[9] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. document VCEG-M33 of ITU-T Video Coding Experts Group (VCEG), April 2001.

[10] 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, 62(3):675–684, September 2016.

[11] S. H. Yang, J. Z. Jiang, and H. J. Yang. *Fast motion estimation for HEVC with directional search*. Electronics Letters, 50(9):673–675, April 2014.

# 4

# Spatial transcoding using machine learning

## 4.1 Rationale and related work

Ultra-high definition displays have become commonplace in the current consumer electronics market, leading to an increasing demand for ultra-high definition content. In heterogeneous multimedia environments, adapting the resolution of such a high definition video to the display capabilities of multimedia terminals is especially necessary. This is due to the fact that while a video is a single presentation with high resolution, it is often displayed on a large number of devices that are differently characterized for their display resolution, processing capacity, battery life, *etc*. Hence, downsizing the spatial resolution of the video stream to the screen size of the end-user device would imply the device limitations. Firstly, since only lowered resolution video is decoded, less at-device computation is required; thus, the battery life is improved. Secondly, a reduction of spatial resolution leads to a bit rate reduction of the video stream, which can save network bandwidth and storage cost.

In the previous chapters, video adaptation with a small bit rate reduction(less than 50%) has been investigated with the use of transrating. For a larger bit rate reduction, spatial resolution down-scaling is highly recommended. Therefore, in this chapter, we optimize the adaptation process where a high bit rate reduction or resolution adaptation is needed by changing the spatial resolution of video.

Reducing the spatial resolution of video streams, which is also referred to

as spatial transcoding, can be performed in the compressed domain or pixel domain. The resolution conversion in the compressed domain provides a very low complexity solution with limited memory size requirements. On the other hand, pixel-based resolution reduction is a high complexity approach, in which decoding, pixel domain downscaling, and subsequently re-encoding are implemented. Given the high importance of reducing the resolution of video streams, many spatial transcoding techniques have been proposed for previous video coding standards such as MPEG-1, MPEG-2, H.264/AVC (e.g., [1–4]). In this chapter, we focus on efficient spatial resolution transcoding for the newly finalized HEVC standard.

Most of the spatial resolution transcoding techniques for the former coding standards (e.g., MPEG-1, MPEG-2) performed the reduction in the compressed domain, typically the frequency domain. In these techniques, reduced-resolution blocks are constructed from the frequency components of the original DCT blocks of the input video. Several arbitrary downsizing approaches in the transform domain have been described in [5, 6]. Although these techniques provided very low complexity solutions, the low-resolution frames suffered from drift errors (as discussed in the open-loop transcoder in Section 2.1) due to which the quality of the output video is gradually degraded.

Yin *et al*. proposed a so-called *Partial_Encode* architecture to minimize the drift error [7]. In this architecture, the error of residual signal due to mismatched referencing is estimated. Then, a compensation process is performed to compensate this error. Finally, the compensated residual is transformed and entropy encoded. Together with the insertion of intra-refresh frames, the proposed architecture in [7] was able to eliminate the drift errors with a trade-off between quality and transcoding complexity.

It is important to note that the aforementioned spatial transcoding techniques would not work well for HEVC video streams since there is a significant difference in the transforming structure between HEVC and the formers. MPEG-1, MPEG-2 and H.264/AVC use the same transform block size for every macroblock (e.g., $4\times4$ or $8\times8$), except the high profile of H.264/AVC in which the $4\times4$ and $8\times8$ transform block sizes are mixed in a frame. In contrast, HEVC applies a quad-tree transform structure, in which a coding unit can be encoded by different transform units of different sizes (ranging from $4\times4$ to $32\times32$). An arbitrary scaling factor would result in a misalignment between the downsized block and its collocated blocks in the original video as depicted in Figure 4.1. This misalignment raises a huge challenge to derive the optimal transform unit structure of an output block from the input blocks. Moreover, other mode mapping issues can arise when performing the resolution downsizing process in the transform domain for a pre-encoded HEVC stream. These issues include sub-block partitioning, multiple reference pictures, and variable prediction direction in B pictures [3].

A closed-loop transcoder, which performs conversion in the pixel domain, pro-

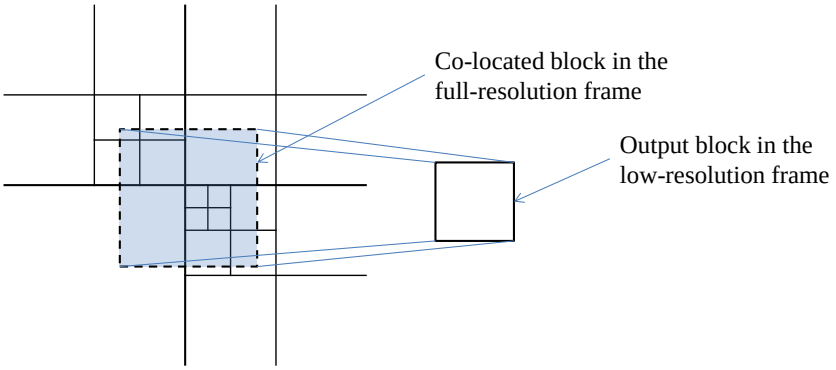



**Figure 4.1:** Misalignment in the transform domain between a block in the downscaled frame and it's collocated blocks in the original frame.

vides high coding performance to reduce the spatial resolution of a video stream. This straightforward transcoding architecture, however, requires a huge amount of computation associated with the re-encoder part. In order to reduce the computational complexity of a pixel-based resolution reduction approach, many techniques have been proposed. These techniques can be grouped into two groups: motion mapping and fast mode decision.

A motion mapping approach has been proposed for downsizing an H.264/AVC video [8]. The motion of the output video is derived from the input motion vectors using a weighted median filter. The obtained motion vector may be further improved by using a quarter pixel refinement. By predicting the motion vectors, motion estimation can be accelerated, which results in a reduced transcoding complexity. This method can achieve a notable complexity reduction because searching integer motion vectors in H.264/AVC has a high portion of the encoding complexity. When this motion mapping approach is apply to a transcoder of HEVC streams without an optimization of block partitioning, the complexity reduction would be limited. The extensive observation in Chapter 3, the integer motion estimation takes a portion of 21% encoding time. Therefore, the motion mapping approach is recommended after the deployment of a block partitioning optimization.

Several fast mode decision techniques have been proposed to reduce the complexity of a closed-loop transcoder [9–11]. These methods obtain the partitioning mode of an output macroblock by down-scaling the partition of the input collocated marcroblocks. Motion estimation is then performed with the derived partition. These methods, however, can work with a simple scaling factor of 2 while in real applications; the scaling factor might be arbitrary.

Alternative techniques predict the encoding information by using machine learning to exploit the correlation between the coding information of the input and out-

put video [12–18]. Different machine learning algorithms have been used in these transcoding techniques. In [12] and [13], support vector machines (SVM [19]) are applied for transcoding H.264/AVC video. Decision trees (DT [20]) are also widely used for transcoding [14–17]. More recently, the linear discriminant function algorithm has been used for transcoding a video from H.264/AVC to HEVC [18].

In the existing literature, most machine learning techniques for transcoding are based on offline-training [12–17]. This means that the prediction model is trained on a set of videos (the training set) and evaluated on a separate test set. In this case, the prediction model only has to be trained once. However, such a model is not content-adaptive. In [18], this problem is solved by using an online-training approach. The prediction model is made content-adaptive by training on the first $L$ frames of the sequence to predict the decisions for the following frames. A disadvantage of this approach is that retraining might be necessary after changes in the content of the video.

The existing transcoding techniques demonstrate that the re-encoding complexity can be significantly reduced by using machine learning. However, the following problems should be considered to achieve the best performance of transcoding. Firstly, only a single machine learning algorithm is applied in existing techniques. However, the effects of using different algorithms are unpredictable. Secondly, although attempts have been spent on using online training, an adaptive feature selection mechanism has not been taken into account. Finally, the optimization of machine learning parameters has not received much attention.

This chapter discuses several techniques to address these issues by applying machine learning to fast downsizing of videos pre-encoded with high efficiency video coding. As the main contribution to the state-of-the-art, the performance of different machine learning strategies is investigated. To determine the optimal strategy, optimized and unoptimized versions of different algorithms are compared. Additionally, the potential benefits of content-adaptive feature selection are examined and both online and offline training strategies are tested.

The rest of this chapter is organized as follows. First, the proposed arbitrary downsizing architecture is described in Section 4.2. Then, the machine learning model for predicting coding unit splits is proposed in Section 4.3. In this section, the optimization of machine learning is elaborated on. Thereafter, a transcoding complexity control mechanism, which can be used with some machine learning algorithms, is presented in Section 4.4. The performance of the proposed techniques is evaluated in Section 4.5. Finally, Section 4.6 conveys the conclusions drawn.

## 4.2 Arbitrary downsizing architecture

The proposed transcoding system is depicted in Figure 4.2. First, the input high resolution HEVC video is decoded followed by the extraction of the coding infor-

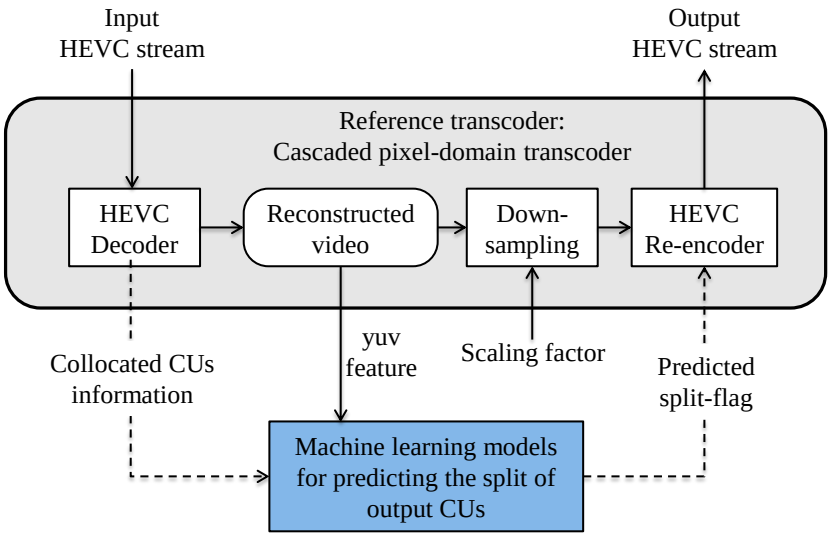



**Figure 4.2:** Proposed arbitrary downsizing architecture for HEVC video. Splitting of output CUs is predicted by using coding information of the input video and features extracted from the reconstructed video.

mation and raw video features of the reconstructed video. Based on the network bandwidth constraints and/or the screen resolution of playback devices, the downsizing factor can be chosen. Using a non-normative downsampling filter [20], the decoded video is then resized by dividing its width and height by this scaling factor. Meanwhile, the splitting behaviour of the CUs in the output video stream is predicted using machine learning models. Finally, the resized video is re-encoded using this predicted information. Details of building and optimizing the prediction model are elaborated on in Section 4.3.

It should be noted that the arbitrary scaling factor may lead to misalignment between the collocated area of the output CU and the collocated input CUs. In this case, the existing mode mapping algorithms [3, 21] cannot be used. However, this problem can be solved by the proposed machine learning method. The correlation between the block size of the output CU and the coding information of partly collocated and fully collocated CUs is exploited by machine learning.

## 4.3 CU splitting prediction models

The CU splitting prediction models are constructed using machine learning algorithms to predict the CU structure of the output video. There are several challenges to be considered when building the prediction models in order to improve

the transcoding performance. First, the accuracy of the prediction should be improved as much as possible; if the prediction is more accurate, the quality loss will be smaller. Furthermore, the prediction step needs to be computationally effective. Secondly, the model should be adaptive, such that it is possible to construct a model and optimize it regardless of transcoding parameters such as the scaling factor of the video resolution.

This chapter proposes several strategies to handle the above challenges to downsize the spatial resolution of an HEVC stream based on machine learning. Since either an online or an offline training mechanism can be applied, these mechanisms are described in the following subsection. To achieve optimal prediction accuracy, several machine learning algorithms are then investigated. For these machine learning algorithms, the optimal parameters are adaptively chosen depending on the training data as described thereafter. The end of the sections presents the justifications on how best features are selected during training by using an adaptive feature selection mechanism. This mechanism aims to lower overall complexity of the machine learning system.

### 4.3.1 Training mechanism

Three prediction models, which respectively predict the splitting behavior of CUs at depth 0, 1, and 2, are constructed with machine learning algorithms. The purpose of the training process is to build these prediction models based on a data set. The data set includes multiple samples. Each sample represents values of features (the coding information of the input CUs) and the split flag of the corresponding CU in the output video. Figure 4.3 visualizes a data set while Figure 4.4 shows an example of a sample.

The performance of several machine learning algorithms has been evaluated in the proposed transcoding architecture. For these algorithms, either an online or offline training strategy can be applied.

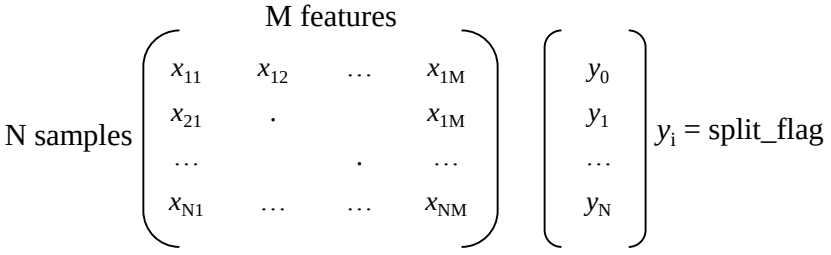



**Figure 4.3:** A data set with N samples. Each sample has M features and the output.

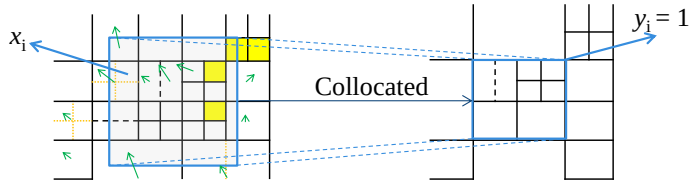



**Figure 4.4:** A sample of data set. The coding information of the collocated blocks (e.g., the splitting behavior at the CU, PU, TU levels, coding mode, and motion vectors) is utilized to predict the split flag an output CU.

### Online training

For online training, a set of $N$ frames is first transcoded without acceleration. Then, the coding information from the input bit stream, the features of the decoded video and the *split-flags* of the CUs in the resized sequence are extracted from these $L$ frames. Based on this training data, the machine learning parameters and the important features are selected. The three prediction models are then built using these optimal settings. Using these models, the CU structure of the following frames can be predicted to reduce the complexity of the re-encoding step.

The advantage of this online training mechanism is that the parameters of the machine learning models adapt to the content of the video. Additionally, this method does not depend on the coding configuration, e.g., it is independent of the scaling factor used during transcoding. However, if the properties of the video content greatly change (e. g., a scene change or zooming), the prediction accuracy of the model might be reduced and retraining might be necessary.

### Offline training

Alternatively offline training can be used, where the first $E$ frames of selected sequences are used as a training set to build the three prediction models. The other sequences are then encoded using these models. One of the challenges of offline training is that it needs a set of training sequences that sufficiently represents the complete test set. To achieve this, the training sequences are selected by using a cross-validation technique. The prediction models are first constructed using only a single training sequence with input resolutions varying from 832x480 up to $1920\times1080$ pixels as listed in [22]. The number of training frames is based on the resolution of the sequence. The $1920\times1080$, $1280\times720$, and $832\times480$ sequences respectively use 50, 75, 100 frames for training. After a prediction model is built using a single sequence, all other sequences are transcoded using this model. The three sequences that result in the highest compression efficiency for the test set are then combined as the final training set on which final offline prediction models are then trained.

The advantage of offline training is that the model only needs to be trained

once for a single scaling factor. However, this also means that for each possible scaling factor, a different model is needed. This might make offline training less fitting for arbitrary downscaling than for, e.g., dyadic downscaling or transcoding between different video standards. Moreover, the performance of the models might be negatively affected if video sequences in the test set differ greatly from the training set.

### 4.3.2 Evaluated machine learning algorithms

Several supervised machine learning algorithms, which label training data and predict the correct label for an input sample, can be used for classifying a set of input data [19, 23–28]. One of the goals of this chapter is to investigate if there is a significant difference between machine learning techniques. Therefore, this chapter elaborates on the performance of four commonly used algorithms: decision trees (DT) [24], RandomForest (RF) [23], adaptive boosting (AdaBoost) [25], and support vector machines (SVM) [19]. The implementation of these algorithms is based on the Python programming language and can be found in [29]. An overview of these algorithms will be provided next.

#### 4.3.2.1 Decision tree classification

The DT algorithm is a technique for classification based on simple decision rules. The model consists of a root, internal nodes, leaf nodes, and branches. At the root and each internal node, the input sample is evaluated using the decision rule of that node. Depending on the outcome of that rule, the input sample follows one of the branches originating from the node. When the input sample reaches a leaf node, the DT model returns the prediction given by that leaf node. Implementation-wise, a tree consists of many if-else statements, which results in a low complexity for generating predictions. Decision trees are also often used since the model can be easily visualized. However, this algorithm bears a disadvantage in its possibility of an overcomplicated tree resulting in overfitting, which negatively influences the performance of the DT model on test sets. A decision tree is also highly sensitive to small variations in the data set, denoting its potential generation of different results when some samples are removed or added. The splitting rule at each node of the decision tree is derived as following.

These are several algorithms that can be used to obtain the splitting rule of a node. Notable ones include ID3 (Iterative Dichotomiser 3 [30], C4.5 [31], CART (Classification and Regression Tree [32]), and CHAID (CHi-squared Automatic Interaction Detector [33]). In this research, the decision tree has been built in use of CART.

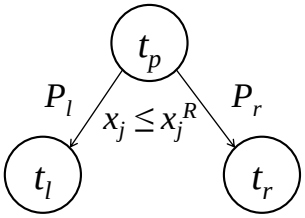



**Figure 4.5:** Splitting rule of CART.

**Building decision tree using CART algorithm based on Gini splitting rule [34]**

Let $t_p$ be a parent node whose two child nodes, i.e., $t_r$ (right node) and $t_l$ (left node). Consider the training sample with variable matrix X with M number of variables $x_j$ and N the number of training samples. Let class vector Y consist of N samples with total amount of K classes. In the proposed transcoder, K is 2 and Y represents the split flag value of the output CUs.

The splitting rule performs the splitting of learning sample into smaller parts. At each node, data have to be divided into two parts with maximum homogeneity as shown in Figure 4.5 where $x_j$ is the $j^{th}$ variable and $x_j^R$ is the best splitting value of variable $x_j$.

The maximum homogeneity requirement of child nodes is characterized by the so-called impurity function $i(t)$. It should be noted that the impurity of parent node $t_p$ is constant for any of the possible splits $x_j \leq x_j^R$, $j = 1, ..., M$. Hence, the maximum homogeneity $i(t_c)$ of child nodes will be equivalent to the maximization of change of impurity function $\delta i(t)$:

$$\delta i(t) = i(t_p) - E[i(t_c)]. \tag{4.1}$$

Let $P_l$, $P_r$ be probabilities of right and left nodes, respectively. The change of impurity can be rewritten as:

$$\delta i(t) = i(t_p) - P_l i(t_l) - P_r i(t_r). \tag{4.2}$$

Consequently, in order to obtain the maximum homogeneity of child nodes, the CART algorithm will solve the following maximization problem:

$$\underset{x_j \leq x_j^R, j=1,...,M}{\arg\max} [i(t_p) - P_l i(t_l) - P_r i(t_r)]. \tag{4.3}$$

By solving Equation 4.3, CART will search through all possible values of all variables in matrix X for the best classification question $x_j \leq x_j^R$ which maximizes the change of impurity function $i(t)$.

The remaining question is how the impurity function $i(t)$ can be defined. In theory, there are several definitions of the impurity function, e.g., Gini splitting

rule, Twoing splitting rule. Among these rules, Gini splitting rule is the most widely used in practice. This rule defines the impurity function as:

$$i(t) = \sum_{k \neq l} p(k|t)p(l|t). \tag{4.4}$$

where $k$, $l$ is the class index; $p(k|t)$ represents the conditional probability of class $k$ provided we are in node $t$.

Applying the impurity function ( 4.4) to ( 4.2), we have:

$$\delta i(t) = -\sum_{k=1}^{K} p^2(k|t_p) + P_l \sum_{k=1}^{K} p^2(k|t_l) + P_r \sum_{k=1}^{K} p^2(k|t_r) \tag{4.5}$$

Hence, the Gini splitting algorithm solves the following maximization problem:

$$\underset{x_j \leq x_j^R, j=1,\ldots,M}{\arg\max} [-\sum_{k=1}^{K} p^2(k|t_p) + P_l \sum_{k=1}^{K} p^2(k|t_l) + P_r \sum_{k=1}^{K} p^2(k|t_r)]. \tag{4.6}$$

The Gini splitting algorithm will search in training set for the largest class and isolate it from the rest of the data.

Expending of a node will be stopped when a certain condition is matched. For instances, splitting of a node is terminated when the depth of this node reaches the allowed maximum depth, or this node contains less than minimum samples split samples.

#### 4.3.2.2 Adaptive boosting

The advantage of decision classification is that decision trees are not stable. A small change in the data can make a large difference, which results in a performance degradation of the classification.

AdaBoost improves the performance of a weak classifier such as a decision tree by using an iterative approach. The main purpose of adaptive boosting is to construct a classification rule C(x) with the lowest misclassification error rate (MER) for a give learning data set. With the assumption that the training data are independently and identically distributed samples from an unknown probability distribution $Prob(X, C)$, then MER for $C(x)$ is defined in [35] as:

$$\begin{aligned} E_{X,C}\mathbb{I}_{C(X)\neq C} &= E_X Prob(C(X) \neq C|X) \\ &= 1 - E_X Prob(C(X) = C|X) \\ &= 1 - \sum_{k=1}^{K} E_X[\mathbb{I}_{C(X)=k} Prob(C = k|X)]. \end{aligned} \tag{4.7}$$

**Algorithm 4** AdaBoost algorithm [25]

1: Initialize the weights for learning samples $w_h = 1/n, h = 1, 2, ..., N$.
2: **for** $i$ = 1 to I **do**
Fit a classifier $T^{(i)}(x)$ to the training data using $w_h$.
Compute

$$err^{(i)} = \sum_{h=1}^{N} w_h \mathbb{I}\big(c_h \neq T^{(i)}(x_h)\big) / \sum_{h=1}^{N} w_h.$$

Compute

$$\alpha^{(i)} = log\frac{1-err^{(i)}}{err^{(i)}}.$$

Update new weights for learning samples

$$w_h \leftarrow w_h.\exp\Big(\alpha^{(i)}.\mathbb{I}\big(c_h \neq T^{(i)}(x_h)\big)\Big), h = 1, 2, ..., N.$$

Re-normalize $w_h$

3: **end for**
4: Output

$$C(x) = \arg\max_k \sum_{i=1}^{\mathrm{I}} \alpha^{(i)}.\mathbb{I}\big(T^{(i)}(x) = k\big).$$

It is clear that, solving the maximization problem 4.8 will minimize the misclassification error quantity.

$$C^*(x) = \arg\max_{k \in K} Prob(C = k|X = x). \tag{4.8}$$

The Adaboost algorithm performs an iterative procedure that combines many weak classifiers to construct the classifier $C^*(x)$. The procedure is initiated by building a classifier (e.g., decision tree) with the use of an unweighted training sample. At each iteration, training samples are assigned a weight, and a new classifier is built using the new weight. At the end of an iteration, the classifier is assigned a score. The final classifier is defined as the linear combination of the classifiers from each iteration. Let $T(x)$ represent a classifier that maps a class label to $x$, then the Adaboost algorithm handles I iterations as in Algorithm 4, where I is the number of boosting iterations.

#### 4.3.2.3 Random forest

Random forest (RF) classifier is also an improvement of decision tree. The decision of the random forest classifier is based on the vote from multiple decision trees. To construct a tree in the RF, a subset of features is randomly selected from all available features. The decision tree is then built with these features based on a sub set of learning samples. The final decision of RF is derived by taking the majority vote of these decision trees as follows:

$$C^*(x) = \arg\max_{k \in K} \sum_{i=1}^{t} f_i(x, k). \tag{4.9}$$

where $f_i(x, k)$ is the vote of tree $i^{th}$. If tree $i^{th}$ assigns $x$ a label $k$, $f_i(x, k)$ is 1. Otherwise, this constitutes 0.

By randomly building these trees and combining these decisions of multiple trees to make the final decision, the random forest algorithm can address the disadvantage of the decision tree algorithm that is highly sensitive to learning data. However, this algorithm is more computationally complex since multiple trees need to be constructed.

#### 4.3.2.4 Supported vector machine learning

While the previous three algorithms use rule-based decision tree classification, supported vector machine (SVM) is memory-based. In SVM, the dataset is mapped to a high-dimensional space with the goal of constructing a hyper-plane that maximizes the distance between samples belonging to different classes. As a result, SVM has higher storage and computing requirements than DT-based algorithms. Moreover, the complexity of the algorithm greatly increases with the number of features and samples.

To construct a classification based on SVM, we consider a binary classification problem of which a sample is assigned a label of either 0 or 1. The main idea of SVM is to obtain a unique separating hyperplane that maximizes margin between two classes. Given $N$ training data samples:

$$\{x_i, y_i\}_{i=1}^{N}, x_i \in R^M, y_i \in \{0, 1\}. \tag{4.10}$$

where $\{x_i, y_i\}$ is the $i^{th}$ training sample, $x_i$ is the input feature vector and $y_i$ is the class label which is the output label of $x_i$.

The decision rule of the SVM algorithm is based on a discriminant function associated with the hyperplane. The function is defined in Equation 4.11, where the kernel function $\phi(.)$ is a nonlinear function that maps the input $x_i$ into a higherdimensional space.

$$f(x) = w^T \phi(x) + b \tag{4.11}$$

Mathematically, the hyperplane is derived by minimizing the cost function defined by Equation 4.12 with the constraints in Equation 4.13.

$$J(w) = \frac{1}{2}w^T x = \frac{1}{2}w^2 = \sum_{i=1}^{N} w_i^2. \tag{4.12}$$

$$y_i\big(w^T\phi(x_i)\big) \geq 1. \tag{4.13}$$

To make the classification problem more general, slack variable $\xi$ and user-defined regularization $C$ are introduced. Hence, the classification problem is derived by minimizing the following quantity:

$$J(w) = \frac{1}{2}w^T x + C\sum_{i=1}^{N} \xi_i \tag{4.14}$$

subject to

$$\begin{aligned} y_i\big(w^T\phi(x_i) + b\big) &\geq 1 - \xi_i \\ \xi_i &\geq 0. \end{aligned} \tag{4.15}$$

With the introduction of $\xi$ and $C$, the optimization in Equation 4.14 becomes a trade-off between the empirical risk (i.e., the training errors represented by the second term) and model complexity (the first term) [36]. It has been proven that, the optimization of this cost function can be solved by the saddle point of Lagrange function:

$$\begin{aligned} \Gamma(w, b, \alpha, \xi, \beta) = \frac{1}{2}w^2 &+ C\sum_{i=1}^{N} \xi_i \\ &- \sum_{i=1}^{N} \alpha_i \Big( y_i\big(w^T\phi(x_i) + b\big) - 1 + \xi_i \Big) \\ &- \sum_{i=1}^{N} \beta_i \xi_i. \end{aligned} \tag{4.16}$$

where $\alpha_i$ and $\beta_i$ are Lagrange multipliers associated with the constraints in Equation 4.15. The Lagrange multipliers are derived by solving the following maximization problem:

$$\alpha^* = \arg\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(x_i x_j). \tag{4.17}$$

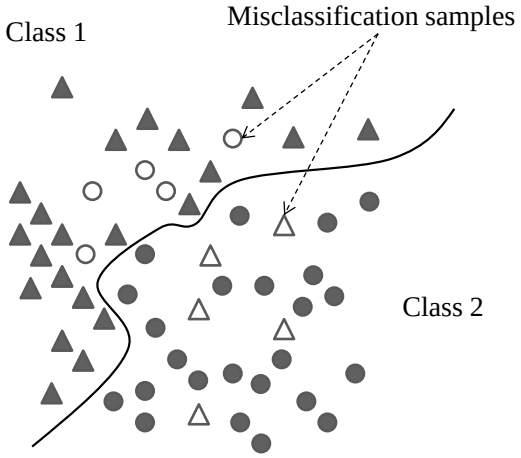



**Figure 4.6:** Examples of Classification using SVM.

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, C \geq \alpha_i \geq 0, i = 1, 2, ..., N \tag{4.18}$$

Herein, $K(x_i x_j) = \phi^T(x_i)\phi(x_j))$ is the kernel operator. The point $x_i$ with a strength of $\alpha_i$ is called support vector.

The problem in Equation 4.17 can be solved by using the *Karush-Kuhn-Tucker* theorem [37]. In the scope of this research, the Gaussian Radial Basis Function (RBF) is investigated as the kernel function since it handles both the non-linear and linear case, and has fewer numerical difficulties. An example of SVM classification using a non-linear RBF kernel is visualized in Figure 4.6.

### 4.3.3 Content-adaptive parameter selection

The classification performance depends not only on the learning algorithm, but also on the parameters of this algorithm. For a given algorithm, the classification accuracy varies widely when the parameter settings change. Therefore, a parameter selection method has to be deployed to select proper parameters for a given data set. The meaning of these parameters is summarized in Table 4.1. In the tree-based algorithms, $max\_depth$ and $min\_samples\_leaf$ of a tree are the most important parameters. In SVM, two parameters of the kernel, $C$ and $\gamma$ significantly affect the prediction performance. Additionally, the number of trees (*ntree*) in a random forest affects not only the coding performance but also the prediction time. A high *ntree* increases the accuracy of the prediction. However, it leads to a higher prediction complexity since the number of trees that need to be evaluated increases

**Table 4.1:** The non-optimized machine learning parameters.

| Algorithm | Parameters | Meaning | Default |
|---|---|---|---|
| SVM | $C$ | Parameter C of the error term | 1 |
| | $\gamma$ | Kernel coefficient for the kernel | 0 |
| Adaboost, DT, RF | $max\_depth$ | The maximum depth of the tree | None |
| | $min\_samples\_leaf$ | The minimum number of samples required to be at a leaf node | 1 |
| RF | *ntree* | Number of trees in the forest | 10 |

linearly with *ntree* [29]. The parameter selection for tree-based algorithms and SVM is presented first. Then, the proposed method for *ntree* selection is given.

#### 4.3.3.1 Optimization of general parameters

General parameters of machine learning algorithms can be derived by using a grid-search with cross-validation [36, 37]. This approach tests all possible combinations for a set of parameters. The combination with the best cross-validation accuracy is chosen. In this chapter, $max\_depth$ is selected from $\{3, 6, 9, 12\}$ while $min\_samples\_leaf$ is $\{1\%, 2\%, 3\%, 4\%, 5\%, 6\%\}$ of the total number of samples in training data. $C$ lies in the range of $\{1, 10, 50, 100, 500, 1000\}$ while is $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

#### 4.3.3.2 Proposed *ntree* selection mechanism

Selecting *ntree* in RF should achieve a trade-off between prediction accuracy and prediction time. If a grid-search is applied, a high ntree is often selected from the given set, resulting in a high prediction complexity. Hence, this approach is not ultimately optimal for selecting *ntree*.

We propose an efficient mechanism for *ntree* selection based on the Out-of-Bag error (*oobE*) - a parameter to estimate the prediction error of a random forest model [29]. First, the estimation of this oob error is presented, and the proposed ntree for the random forest algorithm is given thereafter.

**Out-of-Bag (oob) error calculation:**

We consider building a random forest model with *t* trees (classifiers) based on training set X containing N samples $(x_i, y_i)$.

In order to construct *t* trees of the model, X is randomly re-sampled to create *t* bootstrap datasets ($S = \{S1, S2, ..., S_t\}$). Each classifier $C_i$ is constructed by learning on a bootstrap $S_i$. When all classifiers are generated, *oobE* is derived as follow.

The prediction error for every sample $(x_i, y_i)$ in the original training set X is evaluated. For each $(x_i, y_i)$, a set of classifiers *C* is selected from *t* trees such that all selected classifiers eject $(x_i, y_i)$. Let *K* be the number of selected ones. The misclassification rate $m_i$ of $(x_i, y_i)$ is given as:

$$m_i = \frac{1}{K} \sum_{k=1}^{K} f\big(C_k(x_i)|y_i\big) \tag{4.19}$$

Herein, $f(.)$ is 1 if the classifier $C_k$ returns $y_i$ for the input $x_i$. Otherwise, this quantity is 0.

The *oobE* quantity of the model is the mean of misclassification rates of all samples in the training sets:

$$oobE = \frac{1}{N} \sum_{i=1}^{N} m_i \tag{4.20}$$

**Selecting *ntree* based on oob estimation:**

The prediction performance based on *oobE* of a classification model is defined as follow:

$$oobP = 1 - oobE \tag{4.21}$$

When *ntree* increases, the *oobP* score increases accordingly. However, after a certain number of trees, the prediction performance stabilizes around a threshold *Thr*. This stabilization implies that with a certain number of trees, overfitting is mostly eliminated in building the classification model based on the given training set. The threshold *Thr* is derived by fitting a curve using a classification and regression trees (CART) [32] model as follows. First, *oobP* quantities are obtained for values of ntree from 5 to 50. A CART model is then generated to fit the *oobP* scores (Figure 4.7). *Thr* is set as the maximum *oobP* of the CART model, since this is the value to which *oobP* converges. Then, the optimal *ntree* is selected as:

$$ntree = \underset{n \in (5:50)}{\arg\min} \{oobP_n \geq Thr\} \tag{4.22}$$

where *oobPn* is the *oobP* with *n* trees. Since this *ntree* is the minimal value for which the *oobP* score is equal to or greater than the chosen threshold, this number of trees is assumed to be the optimal trade-off between prediction accuracy and the complexity of the model.

### 4.3.4 Content-adaptive feature selection

The splitting behavior of a CU in the output video is highly dependent on not only the features of the decoded video, but also on the coding information of the

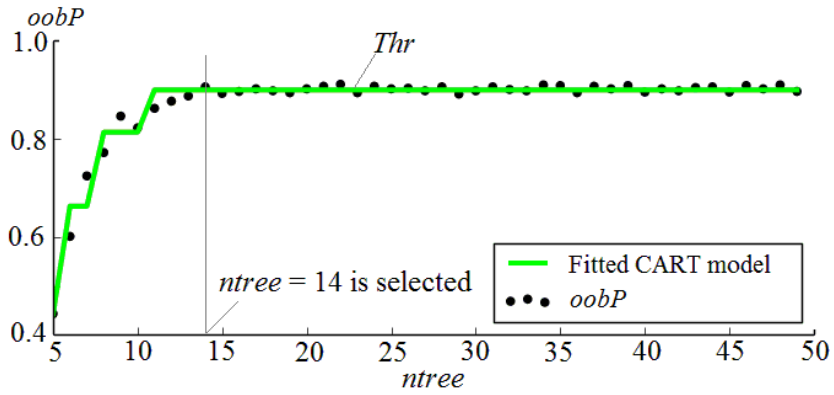



**Figure 4.7:** *ntree* selection for RandomForest, with block size = 32. The input bit stream was encoded using QP = 32 and is downsized by a factor of 2. If grid-search is used, *ntree* is chosen as 49. However, the proposed method selects a smaller *ntree* of 14.

colocated CUs in the input video. Hence, a number of metrics describing the relevant characteristics of the content of the decoded video and the coding information of the input video have been acquired to get an accurate prediction model of the splitting behavior. However, using all of the features might not be optimal since irrelevant features may introduce noise. On the other hand, using only a small set of features brings about a larger generalization error. Therefore, it is necessary to select the most important features [38]. First, an overview of the features is given. Then, a feature elimination mechanism is provided.

#### 4.3.4.1 Overview of all features

A total of 52 features were extracted from the decoded video and the input bit stream. These features were used as the initial feature set for training.

**Features from the decoded video:** A set of 36 features from the decoded video were considered upon various studies [39, 40]. They are based on Sobel filtering on frame pixel values, consecutive frame comparison, pixel value variations, and various combinations and variations of the aforementioned. The calculations were performed on the luminance component of the region of picture that is collocated with the block for which the splitting behavior needs to be determined. These features mostly describe spatial and temporal activity in the picture. Examples include temporal and spatial indexes.

**Features from the input bit stream:** 16 features are based on coding information of the collocated input blocks. These features are the following.

- At the CU level, the mean, variance, maximum, and minimum of the input CU depths are included.
- At the TU level, the mean, variance, maximum, and minimum of input TU depths are also used.
- At the PU level, the depth of a PU is defined as the CU depth if the PU is not split. Otherwise; the PU depth equals the CU depth plus 1. The mean, variance, maximum, and minimum of the input PU depths are included.
- Additionally, the variance of the input motion vectors is taken into account.
- The last two features are the variance and the mean of the transform coefficient variance.

#### 4.3.4.2 Feature elimination using RandomForest

Selection of effective and relevant features is crucial for classification. A good feature selector results in reducing training time, prediction time, as well as reduce memory requirements. In addition, this eliminates irrelevant or noisy features - which can result in an increase of the prediction accuracy. The classical feature selection approaches can be classified into three categories including filter, embedded, and wrapper [41]:

- *The filter approach* consists of a pre-processing procedure prior to the learning step. It uses a specialized feature selection algorithm to remove features from the data set. Based on the statistical model, the importance of every feature is derived. The features are sorted by importance. Only features of greater importance than a threshold are retained. This approach is robust against overfitting since the selection is independent of the machine learning algorithm. This may, however, fail to select the most "useful" features.

  The filter approach is much less time consuming compared to other approaches. Thanks to the computational efficiency, filter method is usually chosen upon a great number of features. However, the filter method requires a threshold to decide what features should be eliminated. A fixed threshold may lead to removing important features or keeping noisy ones.

- *The embedded approach* performs feature selection automatically as part of the machine learning algorithm. During the learning process, the importances of feature subsets are evaluated by a cross-validation mechanism. The most useful subset is searched on their importance.

- *The wrapper approach* performs a pre-processing step like the filter approach. However, it uses a machine learning algorithm as a part of the feature selection process. For example, a random forest algorithm inherently

allows the calculation of feature importance. Evaluations of the importance of feature subsets are through a cross-validation. The feature set with the highest importance can then be retained. The difference between the wrapper and embedded approaches is that the wrapper approach evaluates feature subsets by using a pre-processing procedure with a different machine learning algorithm while embedded approach carries out the evaluation during learning process.

Compared to the filter approach, the embedded and wrapper techniques introduce a higher performance since they always keep important features. In contrast, filter approach is more robust against overfitting. In addition, the complexity of the embedded and wrapper techniques is much higher than the filter method because they perform a search for the best feature subset.

Recently, an enhanced strategy of the filter and wrapper methods have been proposed in [42] which can help the model to keep the useful features and to avoid overfitting. This chapter proposes a content-adaptive feature selection algorithm based on the method proposed in [42]. The selection is conducted before the learning process. The training set for feature selection is obtained from the information of $L$ first frames of the video sequence. The machine learning algorithm used in the selection process is RandomForest, which provides high prediction performance and low training complexity. The selection process is as follows:

1. *Feature ranking*: The importance of each feature is determined for 50 runs of RandomForest training (Figure 4.8). This importance is calculated during each run as the expected fraction of samples that the feature will contribute to. The features are then sorted in descending order upon the average value over the 50 runs (Figure 4.9).

2. *Determining the threshold*: The standard deviation (STD) of the feature importance of each feature is calculated. These results are plotted by the same order as in the feature ranking step. A CART model is then fitted to this data (Figure 4.10). The threshold for feature selection is set as the minimum prediction of the CART model. This threshold is a measure for the minimal noise of the feature importance. If a feature has an importance below this threshold, the importance can therefore be considered as 0.

3. *Feature elimination*: Only features with an average feature score greater than the threshold are retained.

Since in each tree of RandomForest, features and samples are randomly selected, and 50 RamdomForest models are built, features are fairly evaluated. By doing so, the selected features help avoid overfitting. Additionally, the threshold is dynamically derived; hence, the important features are retained while less important ones are removed for every model.

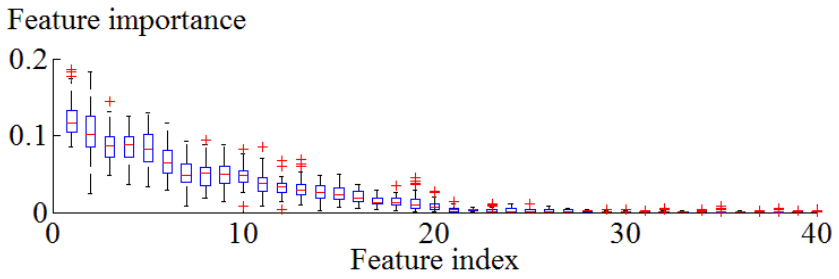



**Figure 4.8:** Feature importance for 50 runs. The variance of the feature importance becomes smaller as the average importance reaches zero.

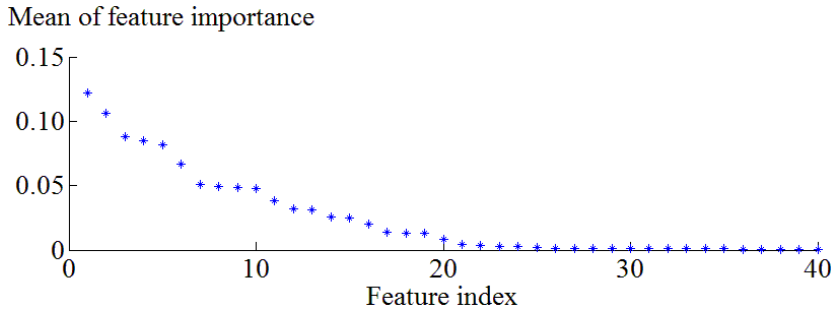



**Figure 4.9:** The average importance of the top 40 most important features. The other features have an importance very close to zero.

## 4.4 Transcoding complexity control

Reducing the transcoding complexity usually results in a decrease in coding performance. However, in some cases, a high coding performance might be necessary. Therefore, an optimal trade-off should be made between transcoding complexity and coding performance.

In order to achieve a trade-off between complexity and coding performance, some machine learning algorithms such as decision trees and random forests can be modified to output probabilistic values for each prediction. This means that the splitting behavior of a CU is predicted with a confidence $c$. To achieve higher coding performance, only decisions with a higher $c$ should be allowed. To control this trade-off between coding performance and transcoding complexity, a threshold $T_c$ is defined. When $c$ is larger than $T_c$, the predicted split-flag directly controls the splitting behavior of CUs; otherwise, the CU is fully evaluated for both split and not split. With a high $T_c$, the number of full evaluations increases, resulting in a higher transcoding complexity, while improving coding performance. Consequently, the transcoding complexity can be controlled by adjusting $T_c$ to achieve

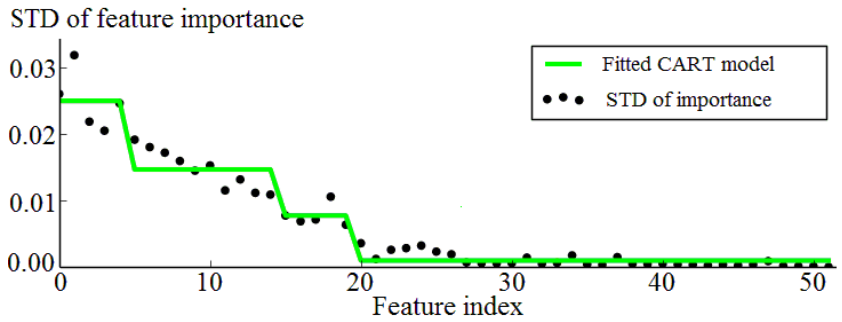



**Figure 4.10:** Selecting the threshold for feature elimination using CART. The standard deviation of important features is larger than for the noisy features, which have a standard deviation close to zero. The threshold, which is the minimum of the CART model, results in selecting the 29 most important features.

a trade-off between complexity and coding performance. The idea of controlling the transcoding complexity using the the probabilistic values for each prediction has been proposed in [43, 44].

## 4.5 Experimental results

The proposed downsizing transcoder was tested with various machine learning algorithms. First, the methodology of experiments is presented. Then, the machine learning algorithms are compared to each other and to a trivial method. The performances of online and offline training strategies are compared as well. The effects of optimizations are also considered. Thereafter, the influence of the complexity control mechanism is shown. Finally, the proposed method is compared to state-of-the art HEVC fast encoding algorithms.

### 4.5.1 Methodology of the experiments

In the following experiments, the original video is encoded using an HEVC encoder following a low delay prediction structure. This structure is characterized by an IPPP prediction order using four reference frames. The QP is set as $\{22, 27, 32, 37\}$. Version 12 of the HEVC reference software [45] is used. Fast motion estimation and fast mode decision are enabled. All sequences (16 in total) bear input resolutions varying from 832x480 up to 1920x1080 pixels except two sequences Traffic ($3840\times2048$) and PeopleOnStreet ($3840\times2160$) [22] which are bigger.

After the initial encoding step, the HEVC bit stream is decoded and downsized by a scaling factor of $\sigma \in \{1.33, 2.00, 4.00, 1.50\}$. Although the algorithm

can handle any possible scaling factor, these factors have been selected as a representative set. They respectively reduce the dimensions of the picture to 3/4, 2/4, 1/4,and 2/3 of the original dimensions. A maximum reduction of 1/4 is used since this decreases the total resolution to 1/16. Additionally, the value of 2/3 was tested since this occurs in a scenario when 1080p video is downsized to 720p.

Following the downsizing step, the resulting video is re-encoded using the proposed machine learning prediction models with the number of training frames $L$ at 10. Since only the impact of $\sigma$ on the transcoding performance is evaluated, the other coding conditions of the output video (profile, QP) are the same as in the input video.

The performance of the proposed transcoding technique is evaluated by comparing it to a non-optimized cascaded decoder-encoder in terms of Bjøntegaard Delta Bit Rate (BDBR) [46] and time saving (TS). BDBR is a measure for the relative bit rate increase to achieve the same quality (expressed in peak signal-to-noise ratio, PSNR) as the cascaded decoder-encoder. The original video is used as a reference for calculating PSNR. Time saving is given by 4.23) where $T_{Proposed}$ is the total encoding time using the proposed method and $T_{Original}$ is the total encoding time using an unmodified cascaded decoder-encoder setup. Since training time for machine learning depends on the implementation efficiency of the algorithm, these time measurements are considered out of scope and therefore, are not taken into account.

$$TS(\%) = \frac{T_{Original(ms)} - T_{Proposed(ms)}}{T_{Original}(ms)} * 100 \tag{4.23}$$

Since using machine learning algorithms results in additional training and prediction phases, simple transcoding (trivial) techniques should also be evaluated. Therefore, this chapter proposes a trivial method to compare with the machine learning based transcoding techniques. This trivial method predicts the splitting behavior of an output CU using the mean depth of the collocated input CUs. In case where the mean depth is higher than the current depth of the output CU, an immediate split occurs in the output CU. Else, the output CU abstains from further splitting and is encoded upon the current depth. The motivation of using the mean depth of the input collocated CUs is that there is a high correlation between this feature and the splitting behavior of output CUs, as the importance of this feature has been observed to consistently rank high compared to other features.

### 4.5.2 Evaluation of the performance of machine learning algorithms for downsizing

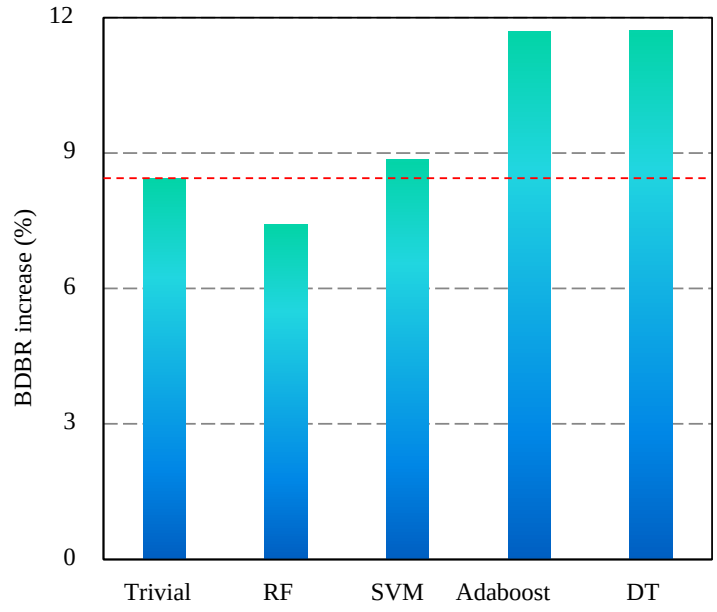
The performance of different machine learning algorithms without optimizing the parameters is presented first. In these tests, the online training mechanism was used. Then, the results with the parameter optimization are discussed. The offline

training mechanism, which has been often used in related work, is also evaluated. Finally, the results of feature selection are presented.

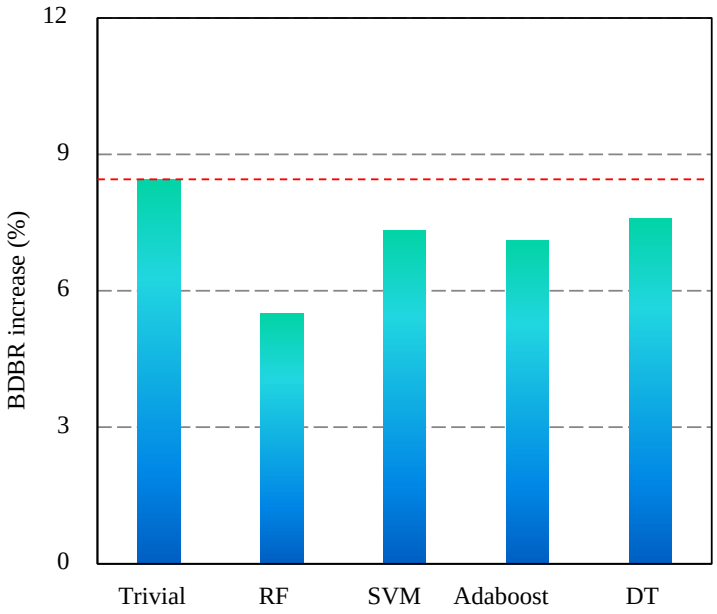
In a first experiment, non-optimized parameters are used. The chosen values for the parameters with a high impact on the prediction accuracy are given in Table 4.1 [29]. Note that setting $max\_depth$ and $min\_samples\_leaf$ to 1 means that the complete tree will be generated, i.e., the model will likely be overfitted to the training data.

Table 4.2 shows the experimental results in using non-optimized parameters for the machine learning algorithm. It is seen that all machine learning methods achieve the same complexity reduction as the trivial method (about 71% on average). These results are similar since the CU structure is always predictable, which means that the encoder can skip the complete CU partitioning process. Any slight difference in complexity reduction owes to the fact that less blocks need to be evaluated if the predicted structure contains more large blocks. These can also be justifiable for the complexity reduction in other tables (Table 4.3, Table 4.4, and Table 4.5).

The experimental results of the nom-optimized model and the proposed optimized model are visualized in Figure 4.11. As is also seen from this figure, contrary to the TS, the performance of the methods differs in terms of BDBR. Using AdaBoost and DT leads to high bit rate increases (around 11.7% on average for each) without the optimization of the machine learning parameters. These two methods perform worse than the trivial method, with a BDBR increase of 8.44%. The performance of RF and SVM is better with an increase by 7.42% and 8.84%, respectively. When the parameters are optimized by using the proposed approach, the performance of the proposed is notably enhanced and better than the trivial method. As a conclusion, unoptimized machine learning models may produce a bad fit for the given data set. Therefore, an optimization of these parameters should be performed if machine learning is used for transcoding.




(a) Non-optimized models.




(b) Optimized models with the proposed online-optimization strategy.

**Figure 4.11:** The performance of non-optimized model and the proposed online-optimization strategy.

**Table 4.2:** Results for an online machine learning strategy, using unoptimized parameters.
The models do not fit the data well. Hence, the bit rate penalty is high for all machine learning algorithm.

| $\sigma$ | BD Rate (%) | | | | | TS (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Trivial | RF | SVM | Adaboost | DT | Trivial | RF | SVM | Adaboost | DT |
| 1.33 | 9.20 | 7.37 | 9.27 | 12.44 | 12.69 | 71.65 | 72.73 | 72.01 | 72.40 | 72.52 |
| 1.50 | 8.26 | 7.14 | 8.87 | 11.60 | 11.79 | 72.23 | 72.49 | 71.93 | 72.16 | 72.43 |
| 2.00 | 7.17 | 7.13 | 7.98 | 10.87 | 10.96 | 71.16 | 71.07 | 70.74 | 70.74 | 70.90 |
| 4.00 | 9.12 | 8.03 | 9.27 | 11.88 | 11.40 | 69.49 | 69.35 | 69.15 | 68.96 | 69.18 |
| Average | 8.44 | 7.42 | 8.85 | 11.70 | 11.71 | 71.13 | 71.41 | 70.96 | 71.06 | 71.26 |

**Table 4.3:** Results for an online machine learning strategy, using optimized parameters, without feature selection.
All algorithms achieve the same time saving. RF is the best method with only 5.41% bit rate penalty.

| $\sigma$ | BD Rate (%) | | | | | | TS (%) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Trivial | RF | $RF_{200}$ | SVM | Adaboost | DT | Trivial | RF | $RF_{200}$ | SVM | Adaboost | DT |
| 1.33 | 9.20 | 5.29 | 5.27 | 6.67 | 7.25 | 7.29 | 71.65 | 71.71 | 71.64 | 72.24 | 71.69 | 72.30 |
| 1.50 | 8.26 | 5.27 | 5.25 | 6.98 | 7.03 | 6.89 | 72.23 | 71.65 | 71.37 | 72.26 | 71.65 | 71.90 |
| 2.00 | 7.17 | 5.15 | 5.11 | 7.04 | 6.75 | 6.87 | 71.16 | 70.32 | 69.96 | 70.27 | 70.33 | 70.88 |
| 4.00 | 9.12 | 5.95 | 5.91 | 7.94 | 7.60 | 8.49 | 69.49 | 68.69 | 68.36 | 67.94 | 68.57 | 69.18 |
| Average | 8.44 | 5.41 | 5.38 | 7.16 | 7.15 | 7.38 | 71.13 | 70.59 | 70.33 | 70.68 | 70.56 | 71.07 |

**Table 4.4:** Results for an offline machine learning strategy, using optimized parameters, without feature selection. The performance of online training is better than offline training.

| $\sigma$ | BD Rate (%) | | | | | TS (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Trivial | RF | SVM | Adaboost | DT | Trivial | RF | SVM | Adaboost | DT |
| 1.33 | 9.20 | 6.75 | 8.56 | 7.31 | 7.42 | 71.65 | 73.71 | 74.47 | 73.93 | 74.25 |
| 1.50 | 8.26 | 6.86 | 8.47 | 7.25 | 8.24 | 72.23 | 73.61 | 73.65 | 73.94 | 74.08 |
| 2.00 | 7.17 | 6.82 | 11.12 | 7.73 | 7.51 | 71.16 | 72.23 | 73.00 | 72.38 | 72.77 |
| 4.00 | 9.12 | 7.85 | 9.82 | 8.43 | 9.48 | 69.49 | 70.44 | 70.95 | 70.82 | 71.05 |
| Average | 8.44 | 7.07 | 9.49 | 7.68 | 8.16 | 71.13 | 72.5 | 73.02 | 72.77 | 73.04 |

**Table 4.5:** Results for an online machine learning strategy, using optimized parameters, with feature selection. The results are similar to not using feature selection.

| $\sigma$ | BD Rate (%) | | | | | TS (%) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Trivial | RF | SVM | Adaboost | DT | Trivial | RF | SVM | Adaboost | DT |
| 1.33 | 9.20 | 5.33 | 6.89 | 7.21 | 7.36 | 71.65 | 71.76 | 72.15 | 71.78 | 72.21 |
| 1.50 | 8.26 | 5.32 | 7.07 | 7.01 | 7.00 | 72.23 | 71.80 | 71.82 | 72.24 | 72.20 |
| 2.00 | 7.17 | 5.22 | 6.75 | 6.78 | 6.92 | 71.16 | 70.24 | 69.76 | 70.28 | 70.47 |
| 4.00 | 9.12 | 6.15 | 8.61 | 7.41 | 8.83 | 69.49 | 68.58 | 67.68 | 67.70 | 68.51 |
| Average | 8.44 | 5.50 | 7.33 | 7.10 | 7.59 | 71.13 | 70.59 | 70.35 | 70.50 | 70.85 |

The transcoding performance with optimized machine learning algorithms as described in Section 4.3 is presented in Table 4.3, where $RF_{200}$ is the result of the RF algorithm with a fixed *ntree* of 200. These experiments use the complete set of 52 features. As can be seen from these results, the coding performance significantly improves compared to using non-optimized parameters. With a slightly different complexity reduction, the bit rate penalty of using optimized parameters is clearly reduced. The SVM, AdaBoost, and DT algorithms demonstrate a similar performance with bit rate increases by 7.16%, 7.15%, and 7.38%, respectively. The RF algorithm is better than the others with a 5.41% bit rate penalty. By using the proposed *ntree* selection, the number of trees varies from 10 to 30. Although the number of trees is notably lower than 200, the performance of RF remains the same while the prediction time is significantly reduced, indicating that the proposed *ntree* selection indeed selects an optimal number of trees.

The results of the offline training strategy are shown in Table 4.4. Online training performs better than offline training even when cross-validation is used to select the training sequences for offline training. For example, the online model has an average BDBR of 5.41% for RF, whereas the offline model has a BDBR of 7.07%. The offline model for SVM has a higher BDBR than the trivial model since optimal machine learning parameters of an offline model may not be optimal for every sequence. Moreover, to apply the offline strategy to other downsizing scaling factors, the model would have to be retrained. Consequently, to achieve the best performance, online training should be used whenever possible.

Finally, the performance of the feature elimination algorithm has also been analyzed. The result of transcoding with optimized parameters and feature selection is shown in Table 4.5. In general, the feature elimination algorithm succeeds in reducing noisy features while retaining a similar coding performance. A comparison of the results in Table 4.5 and Table 4.3, where the complete feature set was used, shows that the BDBR remains similar after feature selection. The influence of the feature selection is higher with a BDBR increase of 5.50%, 7.33%, and 7.59% for RF, SVM, and DT, respectively. On the other hand, the feature selection mechanism decreases the BDBR of the AdaBoost algorithm to 7.10%.

### 4.5.3 Transcoding complexity control scheme

The transcoding complexity can be controlled by adjusting the threshold of the confidence of prediction $T_c$. To investigate the effect of the threshold on the transcoding complexity, this threshold is varied from 0.5 to 0.9 with a step of 0.1. The machine learning algorithm RF offers the best transcoding performance among the investigated algorithms. The experimental results with different thresholds are depicted in Figure 4.12. When the threshold increases, bit rate penalty and complexity reduction increase accordingly.

In particularly, when $T_c$ is set to 0.5, the complexity reduction is around 70% with a bit rate increase of around 6% when $\sigma = 4.00$ and 5% for the other cases. In contrast, when $T_c$ is 0.9, the complexity reduction drops to about 33% with a negligible bit rate penalty.

### 4.5.4 Comparison of the proposed methods with the state-of-the-art fast HEVC encoder algorithms

In order to compare the performance of the proposed method with several state-of-the-art fast encoding algorithms for HEVC, the fast CU decision algorithms including Shen's algorithm [47], Xiong's algorithm [48], and Lee's algorithm [49] have been evaluated. The goal of these methods is to predict the depth of a given CU and allow early termination of the evaluation of this CU. While Shen's and Lee's algorithms utilize observations and statistical experiments, Xiong's algorithm is designed based on the pyramid motion divergence feature which is derived through a theoretical analysis. The coding information from the input video stream is not considered in these methods, since they are not specifically aimed at a transcoding scenario. The average performance of these methods is shown in Figure 4.12.

The proposed method outperforms the state-of-the-art fast encoding algorithms when changing the threshold $T_c$ of the prediction confidence as can be seen in Figure 4.12. When $T_c$ is 0.9, the proposed method results in the same bit rate increase (0.15%). However, the proposed method achieves 35% time saving as much as that of Lees algorithm at 19%. When $T_c$ drops to 0.7, the proposed method and Shen's algorithm have a similar bit rate increase. However, the proposed method outperforms Shen's algorithm in terms of transcoding complexity reduction. The proposed method can achieve 55% complexity reduction while Shen's algorithm achieves 35%. With a slightly lower $T_c$, the proposed method demonstrates a better performance compared to Xiong's algorithm as well.

## 4.6 Conclusion

Different machine learning strategies for downsizing HEVC video are investigated in this chapter. The pre-encoded HEVC video stream is decoded. The reconstructed video is subsequently downsized using an arbitrary factor, which allows adapting the video to the network and/or device constraints. Afterwards, the resized video is re-encoded. The machine learning models utilize the correlation between coding information of the input and output coding units to accelerate the re-encoding process. Different optimized and unoptimized machine learning algorithms have been tested in both online and offline training strategies. Alongside, the effects of an adaptive feature selection algorithm have been investigated.

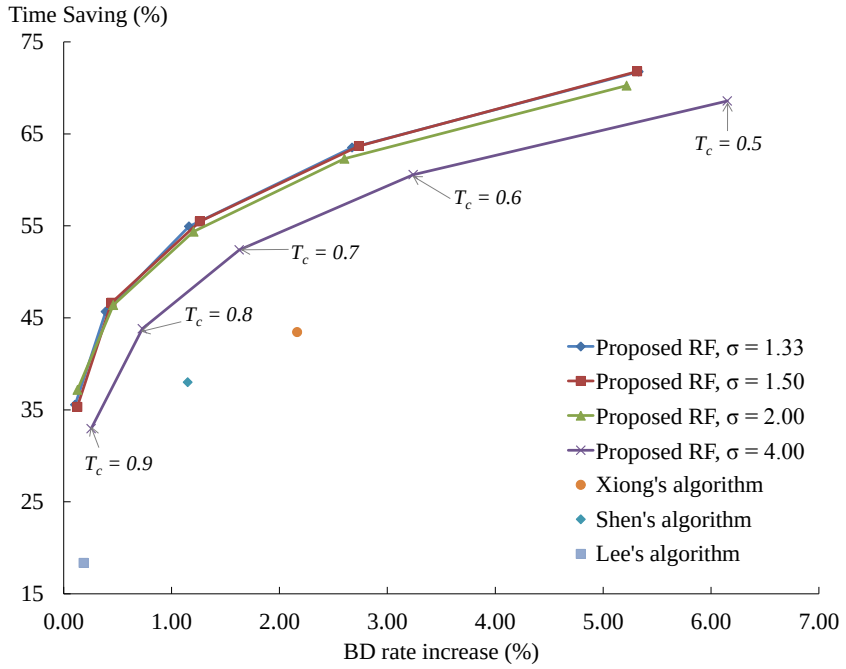



**Figure 4.12:** The transcoding complexity using RF can be controlled by adjusting the threshold of the prediction confidence. Moreover, with a proper $T_c$, the proposed method outperforms the state-of-the-art fast encoding algorithms.

Experimental results have shown that machine learning algorithms should only be used when optimized, since otherwise a trivial method might perform better. If optimizing the machine learning algorithm is seemingly infeasible for the desired application, a trivial method should therefore be recommended instead.

If machine learning methods are used with the proposed optimizations, an online training strategy is preferred over an offline training strategy. This makes the models more adaptive to the content and results in a higher coding efficiency. Additionally, for some algorithms, the transcoding complexity can be controlled to achieve a trade-off between transcoding complexity and coding performance. Among the investigated machine learning algorithms, Random Forest resulted in the best transcoding performance and supports complexity control. This method can reduce the complexity by 71% on average with a bit rate increase by 5.4%. With a negligible bit rate increase (0.1%), this method can reduce the transcoding complexity by 35%.

Based on the proposed work in this chapter, there can be two implications for further improvements of transcoding performance. Firstly, when the content of the video changes (e.g., after a scene change or when the camera pans or zooms),

the online trained models might need to be retrained; thus, retraining strategies for adapting the model to both gradual and sudden changes can be examined. Secondly, the machine learning approaches in this chapter can also be applied to improve other types of transcoding such as transrating, temporal transcoding, and transcoding from other video coding standards to HEVC.

**The work described in this chapter led to the following publications:**

- **Luong Pham Van**, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, Rik Van de Walle. Performance analysis of machine learning for arbitrary downsizing of pre-encoded HEVC video. *IEEE Transactions on Consumer Electronics*. 2015. November 2015.
- **Luong Pham Van**, Johan De Praeter, Glenn Van Wallendael, Jan De Cock, Rik Van de Walle. Machine learning for arbitrary downsizing of pre-encoded video in HEVC. In *IEEE International Conference on Consumer Electronics (ICCE)*. January 2015.

# References

[1] P.A.A. Assuncao and M. Ghanbari. *A Frequency-domain Video Transcoder for Dynamic Bit-rate Reduction of MPEG-2 Bit Streams*. IEEE Transactions on Circuits and Systems for Video Technology, 8(8):953–967, December 1998.

[2] B. Shen, I. K. Sethi, and B. Vasudev. *Adaptive motion-vector resampling for compressed video downscaling*. IEEE Transactions on Circuits and Systems for Video Technology, 9(6):929–936, September 1999.

[3] J. De Cock, S. Notebaert, K. Vermeirsch, P. Lambert, and R. Van de Walle. *Dyadic Spatial Resolution Reduction Transcoding for H.264/AVC*. Multimedia Systems, 16(2):139–149, January 2010.

[4] H. Shu and L.-P. Chau. *The Realization of Arbitrary Downsizing Video Transcoding*. IEEE Transactions on Circuits and Systems for Video Technology, 16(4):540–546, April 2006.

[5] H. Shu and L.-P. Chau. *An efficient arbitrary downsizing algorithm for video transcoding*. IEEE Transactions on Circuits and Systems for Video Technology, 14(6):887–891, June 2004.

[6] V. Patil, R. Kumar, and J. Mukherjee. *A Fast Arbitrary Factor Video Resizing Algorithm*. IEEE Transactions on Circuits and Systems for Video Technology, 16(9):1164–1171, September 2006.

[7] P. Yin, A. Vetro, B. Liu, and H. Sun. *Drift compensation for reduced spatial resolution transcoding*. IEEE Transactions on Circuits and Systems for Video Technology, 12(11):1009–1020, November 2002.

[8] Y.-P. Tam and H. Sun. *Fast Motion Re-Estimation for Arbitrary Downsizing Video Transcoding using H.264/AVC Standard*. IEEE Transactions on Consumer Electronics, 50(3):887–894, August 2004.

[9] P. Zhang, Y. Lu, Q. Huang, and W. Gao. *Mode mapping method for H.264/AVC spatial downscaling transcoding*. In IEEE International Conference on Image Processing (ICIP), pages 2781–2784, October 2004.

[10] H. Shen, X. Sun, F. Wu, H. Li, and S. Li. *A Fast Downsizing Video Transcoder for H.264/AVC with Rate-Distortion Optimal Mode Decision*. In IEEE International Conference on Multimedia and Expo (ICME), pages 2017–2020, July 2006.

[11] J. De Cock, S. Notebaert, K. Vermeirsch, P. Lambert, and R. Van de Walle. *Efficient spatial resolution reduction transcoding for H.264/AVC*. In IEEE International Conference on Image Processing (ICIP), pages 1208–1211, October 2008.

[12] X. Jing, W.-C Siu, L.-P. Chau, and A. Constantinides. *Efficient Inter Mode Decision for H.263 to H.264 Video Transcoding using Support Vector Machines*. In IEEE International Symposium on Circuits and Systems (ISCAS), pages 2349–2352, 2009.

[13] Z.-G. Liu, Y. Yang, and X.-H. Ji. *Fast Macroblock Mode Decision for H.264/AVC Baseline Profile Video Transcoder based on Support Vector Machines*. Multimedia Systems, 18(5):359–372, October 2012.

[14] G. Fernández-Escribano, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Garrido. *A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding*. IEEE Transactions on Circuits and Systems for Video Technology, 18(2):172–185, February 2008.

[15] G. Fernández-Escribano, J. Bialkowski, J. A. Gámez, P. Cuenca, and L. Orozco-Barbosa. *Low-complexity Heterogeneous Video Transcoding using Data Mining*. Transactions on Multimedia, 10(2):286–299, February 2008.

[16] J. Martinez, G. Fernandez-Escribano, H. Kalva, W. Fernando, and P. Cuenca. *Wyner-Ziv to H.264 Video Transcoder for Low Cost Video Encoding*. IEEE Transactions on Consumer Electronics, 55(3):1453–1461, August 2009.

[17] J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle. *Efficient Transcoding for Spatial Misaligned Compositions for HEVC*. In IEEE International Conference on Image Processing (ICIP), pages 2494–2498, 2014.

[18] E. Peixoto, T. Shanableh, and E. Izquierdo. *H.264/AVC to HEVC Video Transcoder Based on Dynamic Thresholding and Content Modeling*. IEEE Transactions on Circuits and Systems for Video Technology, 24(1):99–112, January 2014.

[19] C. Cortes and V. Vapnik. *Support-Vector Networks*. Machine Learning, 20(3):237–297, September 1995.

[20] Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6). *AHG Report on Spatial Scalability Resampling*. JVT-R006, Bangkok, January 2006.

[21] P. Zhang, Y. Lu, Q. Huang, and W. Gao. *Mode Mapping Method for H.264/AVC Spatial Downscaling Transcoding*. In IEEE International Conference on Image Processing (ICIP), pages 2781–2784, October 2004.

[22] F. Bossen. *Common Test Conditions and Software Reference Configurations*. JCTVC-I1100, Geneva, CH, May 2012.

[23] L. Breiman. *Random Forests*. Machine Learning, 45(1):5–32, October 2001.

[24] L. Rokach and O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific, 2008.

[25] Y. Freund and R. E. Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*. Journal of Computer and System Sciences, 55(1):119–139, August 1997.

[26] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge, MA, USA: Cambridge University Press, 2012.

[27] O. Maimon. *Knowledge Discovery and Data Mining-The Info Fuzzy Network (IFN) Methodology*. Kluwer Academic Publishers, Massive Computing, 2001.

[28] T. Shanableh and K. Assaleh. *Feature Modeling using Polynomial Classifiers and Stepwise Regression*. Neurocomputing, 73(10-12):1752–1759, June 2010.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. *Scikit-learn: Machine Learning in Python*. J. Machine Learning Res., 12:2825–2830, October 2011.

[30] J. R. Quinlan. *Induction of decision trees*. Machine Learning, 1(1):81–106, March 1986.

[31] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[32] L. Barber, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, New York, 1984.

[33] G. V. Kass. *An Exploratory Technique for Investigating Large Quantities of Categorical Data*. Journal of the Royal Statistical Society, Series C (applied Statistics), 29(2):119127, December 1979.

[34] R. Timofeev. *Classification and Regression Trees Theory and Applications*. Master Thesis, Humboldt University, Berlin, 2004.

[35] J. Zhu, S. Rosset, H. Zou, and T. Hastie. *Multi-class Adaboost*. Technical report, Department of Statistics,University of Michigan, January 2006.

[36] B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1999.

[37] H. W. Kuhn and A. W. Tucker. *Nonlinear Programming*. In Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pages 481–492, Berkeley, California, 1951. University of California Press.

[38] P. Domingos. *A few useful Things to Know about Machine Learning*. Communications of the ACM, 55(10):78–87, October 2012.

[39] S. Jeannin and A. Divakaran. *MPEG-7 Visual Motion Descriptors*. IEEE Transactions on Circuits and Systems for Video Technology, 11(6):720–724, June 2001.

[40] Z. Ma, M. Xu, and Y.-F. Ou. *Modeling of Rate and Perceptual Quality of Compressed Video as Functions of Frame Rate and Quantization Stepsize and Its Applications*. IEEE Transactions on Circuits and Systems for Video Technology, 22(5):671–682, May 2012.

[41] I. Guyon and A. Elisseeff. *An introduction to variable and feature selection*. Journal of Machine Learning Research, 3:1157–1182, March 2003.

[42] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. *Variable Selection using Random Forests*. Pattern Recognition Letters, 31(14):2225–2236, October 2010.

[43] L. P. Van, J. De Cock, A. J. Diaz-Honrubia, G. Van Wallendael, S. Van Leuven, and R. Van de Walle. *Fast Motion Estimation for Closed-Loop HEVC Transrating*. In IEEE International Conference on Image Processing (ICIP), pages 2492–2496, October 2014.

[44] J. De Praeter, G. Van Wallendael, J. Slowack, and P. Lambert. *Spatially Misaligned HEVC Transcoding with Computational-Complexity Scalability*. Journal of Visual Communication and Image Representation, 40(A):149–158, October 2016.

[45] K. McCann, B. Bross, W.-J. Han, I.-K. Kim, K. Sugimoto, and G. J. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 12 (HM 12) Encoder Description*. JCT-VC, Doc. JCTVC-N1002, July 2013.

[46] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. document VCEG-M33 of ITU-T Video Coding Experts Group (VCEG), April 2001.

[47] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang. *An Effective CU Size Decision Method for HEVC Encoders*. Transactions on Multimedia, 15(2):465–470, February 2013.

[48] J. Xiong, H. Li, Q. Wu, and F. Meng. *A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence*. Transactions on Multimedia, 16(2):559–564, February 2014.

[49] H. S. Lee, K. Y. Kim, T. R. Kim, and G. H. Park. *Fast Encoding Algorithm Based on Depth of Coding-Unit for High Efficiency Video Coding*. Optical Engineering, 51(6):1–11, June 2012.

# 5
# Overall Conclusion

Video is typically distributed in a ubiquitous landscape that is featured by a huge diversity of constraints including limited bandwidth on the network or in resources of end-user devices. To deal with these constraints, there is a need for adaptation of video streams. Video adaptation can be achieved by video transcoding techniques that are able to reduce the bit rate or resolution of the video to satisfy these constraints. Bit rate transcoding (transrating) is often adopted in network bandwidth adaptation while resolution downscaling (spatial transcoding) is used to deal with the limitations of both the network and end-user devices. Thanks to its high flexibility, transcoding has been deployed in many applications. Consequently, research on transcoding is an active topic.

The main downside of a transcoder is its high computational complexity in video conversion, which leads to an increase in the energy cost within the network. Although an open-loop transcoder is considered as a computationally effective solution, it may significantly degrade the visual quality of video. In contrast, a closed-loop transcoder can achieve a better rate-distortion performance. This advantage benefits from re-evaluating the coding information during the re-encoding process. The main downside of a closed loop transcoder is a huge computational complexity associated with the re-encoding part. This work has focused on improving the performance of a closed-loop transcoder. Particularly, the efforts in this dissertation aimed at answering the following two questions:

- **How to effectively deploy the correlation between coding information of the input and output video to reduce the computational complexity of an**

**HEVC video transcoder**. The deployment is able to reduce the transcoding complexity while maintaining coding performance of the video in terms of rate-distortion; and

- **How to achieve a scalable-complexity scheme**. With the use of this scheme, transcoding complexity can be adjusted to get a trade-off between the complexity and the coding performance.

Recently, the new high efficiency video coding standard (HEVC) has been introduced with a superior compression performance related to H.264/AVC. This advantage has made a belief that HEVC would be widely used in the next couple of years. Therefore, this standard has been selected in evaluating the performance of the proposed transcoding techniques.

In Chapter 1, an overview of the HEVC specification has been given. Compared to H.264/AVC, a larger block size is used in HEVC. This large block size, together with a complex qua-tree block partitioning mechanism, has made the encoder very computationally complex. These features result in a huge computational complexity when transcoding a video stream that is originally encoded using this standard.

In Chapter 2, we focused on reducing the computational complexity of an HEVC bit rate transcoder. To reduce complexity, first, an analysis of the block partitioning correlation of the input and output video was carried out. Then, this analysis has been utilized by four proposed techniques to simplify the transcoder.

The first approach in Chapter 2 re-uses the coding structure of the input video to evaluate the coding units in the output video. The complexity reduction of this approach is not very high since the coding unit at higher depth is always evaluated. Experimental results have shown that this approach can achieve a 45% - 50% complexity reduction. A more intelligent approach has made use of machine-learning techniques to early terminate the block partitioning process. In the machine-learning-based approach, the prediction quality is evaluated. Based on the output of prediction and the prediction quality, a decision on splitting of a block is given. By proposing thresholds for the prediction quality, the computational complexity of the transcoder has been controlled, thus resulting in a scalable transcoding-complexity scheme. A good trade-off between the transcoding complexity and coding performance has been found with a 64% complexity reduction at a 1.4% bit rate penalty. It is important to note that these approaches use the traditional evaluation flow of HEVC, in which a coding unit is evaluated from lower depths to higher depths. As a drawback, the splitting behaviour of the higher depth blocks cannot be utilized to terminate the evaluation of lower depth blocks. To address this issue, a novel approach bottom-to-top (B2T) has been proposed. This approach evaluates the coding unit from higher depth first. Then, the coding units at higher depth are conditionally tested given the splitting behavior of CUs at lower

depth. Compared to the machine-learning-based method, this approach achieves the same performance. However, as is simpler for deployment, this approach has been further improved by exploiting the spatial and temporal correlation of splitting behaviour in the output video.

Also in Chapter 2, following the optimization of block partitioning at the coding unit level, an adaptive prediction unit selection has been proposed. This technique is capable of reducing the number prediction candidates for a coding unit from 8 to 1 or 2. Together with four proposed coding unit partition mechanisms, the proposed prediction unit selection has achieved a significant reduction of transcoding complexity. Among these solutions, the novel B2T approaches achieved the best performance with an 80% complexity reduction with a bit rate penalty of 2.5% on average. Additionally, it has been shown that the proposed approaches clearly outperform the state-of-the-art fast transcoding and fast encoding algorithms.

In the evaluation of coding mode for a coding unit, motion estimation accounts for a large portion of complexity. Chapter 3 optimized the motion estimation process of a HEVC bit rate transcoder. Since the motion vectors of a block in the output video and its co-located blocks in the input video reflect the motion of the same object, they are closely related. This correlation has been utilized to simplify the motion estimation process of the transcoder. One challenge is that an output block may associate with multiple input motion vectors. Analysis in Chapter 3 has shown that the input motion vector, that provides the best rate-distortion performance for evaluating the output block, is most correlative with the motion vector of the output block. An adaptive window search using this motion vector has been proposed. Moreover, unnecessary evaluated motion vectors have been discarded by proposed search patterns. Experimental results have shown that the proposed can reduce the transcoding complexity by 16%. When this approach is integrated with the proposed method in Chapter 2, the computational complexity can be reduced by 83.06% on average with a bit rate penalty of 2.4%.

HEVC targets on compressing very high-resolution videos such as full HD, *4k* or even *8k*. Delivering such high-resolution videos to the mobile network is not efficient since mobile devices are incompetent to display full frame of these videos. Therefore, spatial transcoding is needed and this technique has been considered in Chapter 4. The biggest challenge of a spatial transcoder comes from an arbitrary resolution-scaling factor, which results in a miss-alignment between the output block and the co-located blocks in the input video. Moreover, multiple reference directions, the availability of intra-coded blocks in inter frames also forms other challenges. These issues make the use of existing spatial transcoding solutions such as mode mapping or open-loop transcoding impossible. To alleviate these problems, Chapter 4 has made use of machine learning techniques to optimize the spatial transcoder for HEVC stream.

Various machine learning algorithms may provide different transcoding performances. To select the best algorithm, four widely used machine learning ones have been evaluated in predicting the splitting behaviour of the output blocks. Several issues have been raised when using the machine learning algorithms. First, this mission is to select the best algorithm that has the highest prediction performance. Second, parameters of machine learning and the features (variables) should be optimized to avoid over-fitting and over-training as well as to reduce the training and prediction time. It should be noted that these challenges have not been solved in the most machine-learning-based transcoding techniques. As an important contribution, Chapter 4 proposes several solutions to deal with these issues. Content-adaptive parameter estimation with the use of grid-search has been proposed. Additionally, a content-adaptive feature selection with the use of the random forest algorithm has been implemented to remove noise features. Furthermore, the number of tree in the random forest approach is also adaptively selected. Finally, a complexity-scalable scheme has been proposed by re-estimating the prediction that has low quality. Experimental results have shown that the random forest algorithm has demonstrated the best prediction performance. The optimization of parameters and features has provided a significant improvement in the transcoding performance. A comparison to the state-of-the-art fast encoding algorithms has shown that the proposed approaches are clearly superior.

In summary, the proposed approaches have answered the two research questions. The computational complexity of HEVC video transcoding is significantly reduced with a scalable scheme. Moreover, these proposed approaches have clearly shown superior performance relative to the state-of-the-art transcoding algorithms.

**Applications of the proposed techniques to non-transcoding user cases:**

The main purpose of transcoding is to satisfy the network and device constraints when transmitting a video. However, an efficiency transcoding technique would possibly result in benefits for post-processing applications, e.g., watermarking, and data insertion. In a watermarking or data hiding application, the security information may be hidden into a video in pixel domain. In such a scenario, a pre-encoded video need to be decoded and the information is subsequently inserted into the reconstructed video. Thereafter, the video is re-encoded. The re-encoding operation can be accelerated by using the proposed transcoding techniques.

When the bandwidth of the backbone network is rich, simulcast streaming may be selected. In this scenario, the proposed techniques can be utilized to speed up encoding of multiple versions of the same video.

Currently, efforts are ongoing to improve the extensions of the HEVC standard including multi-view coding and scalable coding. In multi-view coding, different views of a same scene are encoded. Therefore, there would be a high correlation between coding information among these views. Hence, encoding of various views could be optimized by using the proposed techniques that have been used to

accelerate bit rate transcoding of HEVC video.

It can be implied that scalable video coding is the middle solution between simulcast and transcoding for video adaptation. It contains the fundamental disadvantages of higher bandwidth and decoding complexities for end users and higher encoding complexities compared to single layer coding. It is unclear whether the scalable extension will dominate single-layer coding in applications that requires video adaptation. In any way, transcoding will not lose its place in such adaptation applications. Even in the case where scalable video coding would be preferred; this technique would get benefits from the proposed techniques to optimize encoding of different layers.

# A Evaluation conditions

## A.1 Encoder configurations

The performance of HEVC is often evaluated in use of various configurations that target different applications [1]. The main difference among these configurations is the prediction structure. The prediction structure and the values of other coding parameters (e.g., quantization parameters, syntax declaration of CU and TU, rate control, and motion estimation strategy) are defined in a configuration file that is read by the encoder of reference software, e.g., HM test model version 12 [2]. In this work, Random-Access (RA) and Low-Delay P (LDP) configurations have been tested. Section A.1.1 and Section A.1.2 listed the setting of these configurations. It should be noted that, in the default setting, the quantization parameter of intra-coded pictures is set 32. However, in the experiments of this work, it has been varied from 22 to 37 by a step of 5.

RA configuration makes use of a hierarchical B structure, in which a picture can use both past pictures and future pictures as reference frames. Therefore, the coding efficiency achieved by RA is higher than the other configurations. As a main drawback, this configuration suffers from a larger delay due to the reordering of the pictures. To allow random accessing and to control error propagation, intra pictures are added periodically.

In LDP configuration, only the first picture of a sequence is intra-coded while the rest are P pictures. The coding delay, in this configuration, may be made small since the reordering of pictures is not allowed and only past pictures are used for

prediction.

### A.1.1 Random-Access configuration:

#======== Profile ================
Profile : main

#======== Unit definition ================ MaxCUWidth : 64 # Maximum coding unit width in pixel.
MaxCUHeight : 64 # Maximum coding unit height in pixel.
MaxPartitionDepth : 4 # Maximum coding unit depth.
QuadtreeTULog2MaxSize : 5 # Log2 of maximum transform size for quadtree-based TU coding (2...6).
QuadtreeTULog2MinSize : 2 # Log2 of minimum transform size for quadtree-based TU coding (2...6).
QuadtreeTUMaxDepthInter : 3
QuadtreeTUMaxDepthIntra : 3

#======== Coding Structure =============
IntraPeriod : 32 # Period of I-Frame ( -1 = only first).
DecodingRefreshType : 1 # Random Accesss 0:none, 1:CRA, 2:IDR, 3:Recovery Point SEI.
GOPSize : 8 # GOP Size (number of B slice = GOPSize-1).
#Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2 temporal_id #ref_pics_active #ref_pics reference pictures predict deltaRPS #ref_idcs reference idcs
Frame1: B 8 1 0.442 0 0 0 2 3 -8 -12 -16 0
Frame2: B 4 2 0.3536 0 0 1 2 3 -4 -8 4 1 4 4 1 1 0 1
Frame3: B 2 3 0.3536 0 0 2 2 4 -2 -6 2 6 1 2 4 1 1 1 1
Frame4: B 1 4 0.68 0 0 3 2 4 -1 1 3 7 1 1 5 1 0 1 1 1
Frame5: B 3 4 0.68 0 0 3 2 4 -1 -3 1 5 1 -2 5 1 1 1 1 0
Frame6: B 6 3 0.3536 0 0 2 2 3 -2 -6 2 1 -3 5 0 1 1 1 0
Frame7: B 5 4 0.68 0 0 3 2 4 -1 -5 1 3 1 1 4 1 1 1 1
Frame8: B 7 4 0.68 0 0 3 2 4 -1 -3 -7 1 1 -2 5 1 1 1 1 0

#========== Motion Search =============
FastSearch : 1 # 0:Full search . 1:TZ search.
SearchRange : 64 # (0: Search range is a Full frame).
BipredSearchRange : 4 # Search range for bi-prediction refinement.
HadamardME : 1 # Use of hadamard measure for fractional ME.
FEN : 1 # Fast encoder decision.
FDM : 1 # Fast Decision for Merge RD cost.

#======== Quantization =============
QP : 32 # Quantization parameter(0-51).
MaxDeltaQP : 0 # CU-based multi-QP optimization.
MaxCuDQPDepth : 0 # Max depth of a minimum CuDQP for sub-LCU-level delta QP.
DeltaQpRD : 0 # Slice-based multi-QP optimization.
RDOQ : 1 # RDOQ
RDOQTS : 1 # RDOQ for transform skip.

#========== Deblock Filter ============
LoopFilterOffsetInPPS : 1 # Dbl params: 0=varying params in SliceHeader, param = base_param + GOP_offset_param; 1 (default) =constant params in PPS, param = base_param)
LoopFilterDisable : 0 # Disable deblocking filter (0=Filter, 1=No Filter).
LoopFilterBetaOffset_div2 : 0 # base_param: -6 to 6.
LoopFilterTcOffset_div2 : 0 # base_param: -6 to 6.
DeblockingFilterMetric : 0 # blockiness metric (automatically configures deblocking parameters in bitstream). Applies slice-level loop filter offsets (LoopFilterOffsetInPPS and LoopFilterDisable must be 0).

#========== Misc. ============
InternalBitDepth : 8 # codec operating bit-depth.

#========== Coding Tools ==================
SAO : 1 # Sample adaptive offset (0: OFF, 1: ON).
AMP : 1 # Asymmetric motion partitions (0: OFF, 1: ON).
TransformSkip : 1 # Transform skipping (0: OFF, 1: ON).
TransformSkipFast : 1 # Fast Transform skipping (0: OFF, 1: ON).
SAOLcuBoundary : 0 # SAOLcuBoundary using non-deblocked pixels (0: OFF, 1: ON).

#============ Slices ================
SliceMode : 0 # 0: Disable all slice options.
# 1: Enforce maximum number of LCU in an slice,
# 2: Enforce maximum number of bytes in an 'slice'
# 3: Enforce maximum number of tiles in a slice.
SliceArgument : 1500 # Argument for 'SliceMode'.
# If SliceMode==1 it represents max. SliceGranularity-sized blocks per slice.
# If SliceMode==2 it represents max. bytes per slice.
# If SliceMode==3 it represents max. tiles per slice.

LFCrossSliceBoundaryFlag : 1 # In-loop filtering, including ALF and DB, is across or not across slice boundary.
# 0:not across, 1: across.

#============ PCM ================
PCMEnabledFlag : 0 # 0: No PCM mode.
PCMLog2MaxSize : 5 # Log2 of maximum PCM block size.
PCMLog2MinSize : 3 # Log2 of minimum PCM block size.
PCMInputBitDepthFlag : 1 # 0: PCM bit-depth is internal bit-depth. 1: PCM bit-depth is input bit-depth.
PCMFilterDisableFlag : 0 # 0: Enable loop filtering on I_PCM samples. 1: Disable loop filtering on I_PCM samples.

#============ Tiles ================
TileUniformSpacing : 0 # 0: the column boundaries are indicated by TileColumnWidth array, the row boundaries are indicated by TileRowHeight array.
# 1: the column and row boundaries are distributed uniformly.
NumTileColumnsMinus1 : 0 # Number of tile columns in a picture minus 1
TileColumnWidthArray : 2 3 # Array containing tile column width values in units of CTU (from left to right in picture).
NumTileRowsMinus1 : 0 # Number of tile rows in a picture minus 1.
TileRowHeightArray : 2 # Array containing tile row height values in units of CTU (from top to bottom in picture).
LFCrossTileBoundaryFlag : 1 # In-loop filtering is across or not across tile boundary.
# 0:not across, 1: across.

#============ WaveFront ================
WaveFrontSynchro : 0 # 0: No WaveFront synchronisation (WaveFrontSubstreams must be 1 in this case).

#============ Quantization Matrix ================
ScalingList : 0 # ScalingList 0 : off, 1 : default, 2 : file read
ScalingListFile : scaling_list.txt # Scaling List file name. If file is not exist, use Default Matrix.

#============ Lossless ================
TransquantBypassEnableFlag : 0 # Value of PPS flag.
CUTransquantBypassFlagForce: 0 # Force transquant bypass mode, when transquant_bypass_enable_flag is enabled.

#============ Rate Control ======================
RateControl : 0 # Rate control: enable rate control.
TargetBitrate : 1000000 # Rate control: target bitrate, in bps.
KeepHierarchicalBit : 2 # Rate control: 0: equal bit allocation; 1: fixed ratio bit allocation; 2: adaptive ratio bit allocation.
LCULevelRateControl : 1 # Rate control: 1: LCU level RC; 0: picture level RC.
RCLCUSeparateModel : 1 # Rate control: use LCU level separate R-lambda model.
InitialQP : 0 # Rate control: initial QP.
RCForceIntraQP : 0 # Rate control: force intra QP to be equal to initial QP.
#============ End of configuration ======================

### A.1.2 Low-Delay P configuration:

#======== Profile ================
Profile : main

#======== Unit definition ================
MaxCUWidth : 64 # Maximum coding unit width in pixel
MaxCUHeight : 64 # Maximum coding unit height in pixel
MaxPartitionDepth : 4 # Maximum coding unit depth
QuadtreeTULog2MaxSize : 5 # Log2 of maximum transform size for quadtree-based TU coding (2...6)
QuadtreeTULog2MinSize : 2 # Log2 of minimum transform size for quadtree-based TU coding (2...6)
QuadtreeTUMaxDepthInter : 3
QuadtreeTUMaxDepthIntra : 3

#======== Coding Structure =============
IntraPeriod : -1 # Period of I-Frame ( -1 = only first)
DecodingRefreshType : 0 # Random Accesss 0:none, 1:CRA, 2:IDR, 3:Recovery Point SEI
GOPSize : 4 # GOP Size (number of B slice = GOPSize-1)
#Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2 temporal_id #ref_pics_active #ref_pics reference pictures predict deltaRPS #ref_idcs reference idcs
Frame1: P 1 3 0.4624 0 0 0 4 4 -1 -5 -9 -13 0
Frame2: P 2 2 0.4624 0 0 0 4 4 -1 -2 -6 -10 1 -1 5 1 1 1 0 1
Frame3: P 3 3 0.4624 0 0 0 4 4 -1 -3 -7 -11 1 -1 5 0 1 1 1 1
Frame4: P 4 1 0.578 0 0 0 4 4 -1 -4 -8 -12 1 -1 5 0 1 1 1 1

#========== Motion Search =============

FastSearch : 1 # 0:Full search 1:TZ search
SearchRange : 64 # (0: Search range is a Full frame)
BipredSearchRange : 4 # Search range for bi-prediction refinement
HadamardME : 1 # Use of hadamard measure for fractional ME
FEN : 1 # Fast encoder decision
FDM : 1 # Fast Decision for Merge RD cost

#======== Quantization =============
QP : 32 # Quantization parameter (0-51)
MaxDeltaQP : 0 # CU-based multi-QP optimization
MaxCuDQPDepth : 0 # Max depth of a minimum Cu DQP for sub-LCU-level delta QP
DeltaQpRD : 0 # Slice-based multi-QP optimization
RDOQ : 1 # RDOQ
RDOQTS : 1 # RDOQ for transform skip
TransformSkip : 1 # Transform skipping (0: OFF, 1: ON)
TransformSkipFast : 1 # Fast Transform skipping (0: OFF, 1: ON)

#========== Deblock Filter ============
LoopFilterOffsetInPPS : 1 # Dbl params: 0=varying params in SliceHeader, param = base_param + GOP_offset_param; 1 (default) =constant params in PPS, param = base_param)
LoopFilterDisable : 0 # Disable deblocking filter (0=Filter, 1=No Filter)
LoopFilterBetaOffset_div2: 0 # base_param: -6 to 6
LoopFilterTcOffset_div2 : 0 # base_param: -6 to 6
DeblockingFilterMetric : 0 # blockiness metric (automatically configures deblocking parameters in bitstream). Applies slice-level loop filter offsets (LoopFilterOffsetInPPS and LoopFilterDisable must be 0)

#========== Misc. ============
InternalBitDepth: 8 # codec operating bit-depth

#========== Coding Tools =================
SAO : 1 # Sample adaptive offset (0: OFF, 1: ON)
AMP : 1 # Asymmetric motion partitions (0: OFF, 1: ON)
SAOLcuBoundary : 0 # SAOLcuBoundary using non-deblocked pixels (0: OFF, 1: ON)

#============ Slices ================
SliceMode : 0 # 0: Disable all slice options.
# 1: Enforce maximum number of LCU in an slice,

# 2: Enforce maximum number of bytes in an 'slice'
# 3: Enforce maximum number of tiles in a slice
SliceArgument : 1500 # Argument for 'SliceMode'. # If SliceMode==1 it represents max. SliceGranularity-sized blocks per slice. # If SliceMode==2 it represents max. bytes per slice.
# If SliceMode==3 it represents max. tiles per slice.
LFCrossSliceBoundaryFlag : 1 # In-loop filtering, including ALF and DB, is across or not across slice boundary.
# 0: not across, 1: across

#============ PCM ================
PCMEnabledFlag: 0 # 0: No PCM mode.
PCMLog2MaxSize: 5 # Log2 of maximum PCM block size.
PCMLog2MinSize: 3 # Log2 of minimum PCM block size.
PCMInputBitDepthFlag: 1 # 0: PCM bit-depth is internal bit-depth. 1: PCM bit-depth is input bit-depth.
PCMFilterDisableFlag: 0 # 0: Enable loop filtering on I_PCM samples. 1: Disable loop filtering on I_PCM samples.

#============ Tiles ================
TileUniformSpacing: 0 # 0: the column boundaries are indicated by TileColumnWidth array, the row boundaries are indicated by TileRowHeight array
# 1: the column and row boundaries are distributed uniformly.
NumTileColumnsMinus1: 0 # Number of tile columns in a picture minus 1.
TileColumnWidthArray : 2 3 # Array containing tile column width values in units of CTU (from left to right in picture).
NumTileRowsMinus1: 0 # Number of tile rows in a picture minus 1.
TileRowHeightArray : 2 # Array containing tile row height values in units of CTU (from top to bottom in picture).
LFCrossTileBoundaryFlag: 1 # In-loop filtering is across or not across tile boundary.
# 0:not across, 1: across.

#============ WaveFront ================
WaveFrontSynchro : 0 # 0: No WaveFront synchronisation

#============ Quantization Matrix ================
ScalingList: 0 # ScalingList 0 : off, 1 : default, 2 : file read.
ScalingListFile : scaling_list.txt # Scaling List file name. If file is not exist, use Default Matrix.

#============ Lossless ================
TransquantBypassEnableFlag : 0 # Value of PPS flag.
CUTransquantBypassFlagForce: 0 # Force transquant bypass mode, when transquant_bypass_enable_flag is enabled

#============ Rate Control ======================
RateControl : 0 # Rate control: enable rate control
TargetBitrate : 1000000 # Rate control: target bitrate, in bps
KeepHierarchicalBit : 2 # Rate control: 0: equal bit allocation; 1: fixed ratio bit allocation; 2: adaptive ratio bit allocation
LCULevelRateControl: 1 # Rate control: 1: LCU level RC; 0: picture level RC
RCLCUSeparateModel : 1 # Rate control: use LCU level separate R-lambda model
InitialQP : 0 # Rate control: initial QP
RCForceIntraQP : 0 # Rate control: force intra QP to be equal to initial QP.
#============ End of configuration ======================

## A.2 Test sequences

In order to evaluate the performance of HEVC during its standardization, various uncompressed test sequences have been used as listed in Table A.1. These sequences are differently characterized by picture size and possible application and. These test sequences are categorized into six labels from A to F. Class A consists of sequences that were originally captured at a higher resolution than 1080p HDTV. This high resolution class is used to evaluate the coding performance of 4K or 8K video. However, these original picture sizes have been cropped to 2560x1600 pixels to reduce encoding time. Class B, which contains HDTV sequences at the picture size of 1920x1080 pixels, is for coding performance evaluation of 1080p HDTV. The sequence in classes C and D is smaller with picture sizes of 832x480 pixels and 416x240 pixels, respectively. The goal of defining these classes is to evaluate coding performance for mobile applications. Class E constitutes test sequences with a picture resolution of 1280x720 pixels. This class is used to evaluate coding performance in use of low-latency applications such as visual communications. While the sequences of the aforementioned classes are natural sequences (camera captured content), class F sequences are non-camera captured content. They are used for coding performance evaluation of video screen content, e.g., text and computer graphic.

**Table A.1:** Test sequences

| Class | Sequence name | Frame count | Frame rate (fps) | Resolution |
|---|---|---|---|---|
| A | Traffic | 150 | 30 | 2560x1600 |
| | PeopleOnStreet | 150 | 30 | |
| | Nebuta | 300 | 60 | |
| | SteamLocomotive | 300 | 60 | |
| B | Kimono | 240 | 24 | 1920x1080 |
| | ParkScene | 240 | 24 | |
| | Cactus | 500 | 50 | |
| | BQTerrace | 600 | 60 | |
| | BasketballDrive | 500 | 50 | |
| C | RaceHorses | 300 | 30 | 832x480 |
| | BQMall | 600 | 60 | |
| | PartyScene | 500 | 50 | |
| | BasketballDrill | 500 | 50 | |
| D | RaceHorses | 300 | 30 | 416x240 |
| | BQSquare | 600 | 60 | |
| | BlowingBubbles | 500 | 50 | |
| | BasketballPass | 500 | 50 | |
| E | FourPeople | 600 | 60 | 1280x720 |
| | Johnny | 600 | 60 | |
| | KristenAndSara | 600 | 60 | |
| F | BaskeballDrillText | 500 | 50 | 832x480 |
| | ChinaSpeed | 500 | 30 | 1024x768 |
| | SlideEditing | 300 | 30 | |
| | SlideShow | 500 | 20 | |

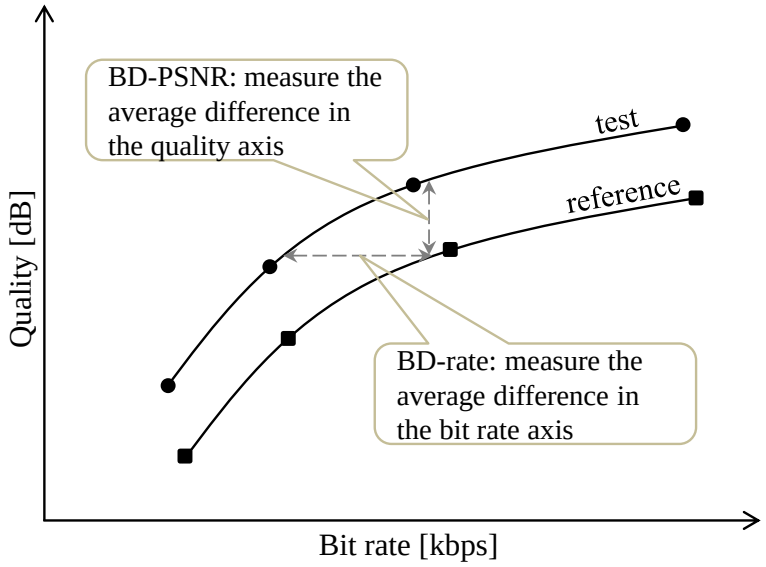



**Figure A.1:** An example of RD curves.

## A.3 Rate distortion curves and Bjøntegaard delta calculation

Rate distortion (RD) curve is often used to visualize the coding performance of a video codec. Such a RD curve represents the encoded results including bit rate (dented by the horizontal axis) and the resulting quality (denoted by the vertical axis). Comparison of the coding performance of various codecs can be obviously seen by plotting their RD cures on the same paragraph. The codec of the upper-left RD curve demonstrates a higher coding efficiency since it can achieve higher quality at lower bit rates as illustrated in Figure A.1.

Peak signal to noise ratio (PSNR) is widely used to objectively evaluate picture quality. PSNR can be calculated for each color component of pictures, i.e., luminance (Y) and chrominance components. Due to the fact that human visual system is more sensitive to luminance than to chrominance, PSNR of luminance (PSNR Y) is a more important calculation. Therefore, PSNR Y has been used for objective quality measurements in this dissertation. PSNR can be derived by the following equation.

$$PSNR = 10log_{10}\frac{(2^{bitdepth}-1)^2 * W * H}{\sum_i (O_i - D_i)^2}. \tag{A.1}$$

Herein, bitdepth represents bit depth of each pixel. *W* and *H* indicate width and

height of picture, respectively, $O_i$ and $D_i$ as pixel values of the reference picture and the decoded picture. *i* is the pixel index.

In order to make an effective comparison between coding performance of two codec (e.g., test and reference), a so-called Bjntegaard delta (BD) metric [3] has been proposed to measure the average difference of the two RD curves. This metric basically consists of two variations, i.e., BD-rate and BD-PSNR. On one hand, BD-rate indicates the average bit rate difference (difference in horizontal direction). A negative value of BD-rate implies that the test codec is better than the reference codec since it can achieve the same quality level at lower bit rates. On the other hand, BD-PSNR measures the average PSNR difference (difference in vertical direction).

In order to calculate the BD metric, the RD curves have to be obtained. It is difficult to get all the values of these curves; therefore, for each curve, four data points (PSNR and bit rate points) are obtained. Then, the curves are approximated by a third order polynomial interpolation using these corresponding four data points. Finally, BD-rate and BD-PSNR are derived by integrating the difference of two curves. The integration is implemented in horizontal direction for BD-rate and in vertical direction for BD-PSNR.

# References

[1] F. Bossen. *Common Test Conditions and Software Reference Configurations*. JCTVC-L1110, Geneva, CH, January 2014.

[2] K. McCann, B. Bross, W.-J. Han, I.-K. Kim, K. Sugimoto, and G. J. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 12 (HM 12) Encoder Description*. JCT-VC, Doc. JCTVC-N1002, July 2013.

[3] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. document VCEG-M33 of ITU-T Video Coding Experts Group (VCEG), April 2001.