Op naar één video-encoder per individu: gestuurde 'High Efficiency Video Coding'

Towards One Video Encoder per Individual: Guided High Efficiency Video Coding

Johan De Praeter

Promotoren: prof. dr. P. Lambert, dr. ir. G. Van Wallendael
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 978-94-6355-005-5
NUR 965
Wettelijk depot: D/2017/10.500/40

Examencommissie

Promotoren:
prof. dr. Peter Lambert (Universiteit Gent)
dr. ir. Glenn Van Wallendael (Universiteit Gent)

Stemgerechtigde leden:
(Voorzitter) prof. dr. ir. Gert De Cooman (Universiteit Gent)
prof. dr. ir. Filip De Turck (Universiteit Gent)
dr. ir. Jan Aelterman (Universiteit Gent)
ir. Werner Van Leekwijck (Nokia Bell Labs, Antwerpen)
prof. dr. ir. José Luis Martínez (Universidad de Castilla-La Mancha, Spanje)
prof. dr. ir. Béatrice Pesquet-Popescu (Télécom ParisTech, Frankrijk)

Interne verdediging: 2 mei 2017
Publieke verdediging: 8 juni 2017

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Elektronica en Informatiesystemen
IDLab

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

Tel.: +32-9-331.49.93

Acknowledgments

Four years ago, I was writing the acknowledgements of my master dissertation. Now, having switched to American spelling, I am writing the acknowledgments of my PhD dissertation, which will be a fair bit more elaborate than what I wrote back then. After all, compared to a master thesis, working on a PhD thesis takes much longer, and involves many more people. As such, I would like to express my gratitude to all those who contributed either directly or indirectly to the realization of this dissertation.

First of all, I would once again, as in my master dissertation, like to thank prof. Rik Van de Walle, whose courses on Multimedia Techniques and Design of Multimedia Applications led me to find IDLab (formerly Data Science Lab, formerly-formerly Multimedia Lab). Also, I would like to thank him for creating this wonderful lab.

Furthermore, I would like to thank my promotor, prof. Peter Lambert, for leading our multimedia compression group within IDLab and for keeping the river of funding flowing. I would also like to thank my former co-promotor prof. Jan De Cock, who guided my during the first years of my PhD, somewhat like a fatherly figure. In the same way, I would also like to thank my current co-promotor, dr. ir. Glenn Van Wallendael, who took over this guiding from Jan, and to whom I am very thankful for always pointing out whenever I tended to make things overcomplicated.

Additionally, I also wish to thank the members of the examination board for their feedback that improved the quality of this work: prof. Gert De Cooman, prof. Filip De Turck, dr. ir. Jan Aelterman, ir. Werner Van Leekwijck, prof. José Luis Martínez, and prof. Béatrice Pesquet.

Next, I would also like to thank both my current and former colleagues from the office C3.02 for the interesting discussions, so thanks Glenn (all possible things related to video compression), Tom (stuff about HDR), Thijs (the complexity of encoders), Ruben (machine learning and models), Niels (linking research to actually putting it into practice), and Vasilis (motion vectors and optical flows). Also, a big thank you to Luong as well for the regular discussions and collaborations related to video transcoding, and also to Sebastiaan for teaching me how to use connecting words that helped to make my papers flow better.

Besides people from my office, I am also grateful to the people with whom I worked together in projects and who provided me with many new insights. In

particular, I want to thank Jürgen from Barco, as well as Patrice and Jef from Nokia Bell Labs, and prof. Munteanu from the Vrije Universiteit Brussel. Then, going beyond Belgium, I would also like to thank Antonio from Universidad de Castilla-La Mancha in Albacete for our discussions and collaborations during and also after his three-month research stay with us in Ghent.

Now, I believe that I have listed all people who have had a direct impact on the contents of my dissertation. Of course, these four years would not have been the same without the lunchtime buddies (excluding the people from my office in order to prevent too much repetition), so thanks to Aza, Olivier, Florian, Gerald, Martin, Kristof, Laura, and (sometimes present) Steven and Baptist. Also, thanks to the gaming office (Gaétan, Jonas, and Ignace) for putting up with me whenever I came to chat about random stuff. Also, before I forget, also thanks to Ellen (and Laura and Kristof who were already mentioned) for the administrative work related to my PhD.

Finally, I would also like to thank my parents for their moral support and for always being there when I need them. And then, last but not least – on the contrary actually, I want to give a huge thanks to my wife, Steffie, (and by extension to our cat, Tijger, who faithfully keeps her company whenever I'm at work or at a conference) for putting up with me and both loving and supporting me through both the bright and dark moments of my PhD and life in general.

May 2017
Johan De Praeter

Table of Contents

List of Acronyms

A

AVC	Advanced Video Coding

B

BD	Bjøntegaard Delta

C

CIC	Coding Information Calculation
CTU	Coding Tree Unit
CU	Coding Unit

F

fps	frames per second

H

HD	High-Definition
HDR	High-Dynamic-Range
HEVC	High Efficiency Video Coding
HM	HEVC reference software

I

IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector

J

JCT-VC	Joint Collaborative Team on Video Coding
JVET	Joint Video Exploration Team

L

LDR	Low-Dynamic-Range

M

MPEG	Moving Picture Experts Group
MSE	Mean Squared Error

P

PSNR	Peak Signal-to-Noise Ratio
PU	Prediction Unit

Q

QP	Quantization Parameter

R

RD	Rate-Distortion
RDO	Rate-Distortion Optimization
RE	Residual Encoder
RoI	Region of Interest

S

SI	Spatial Index
SVM	Support Vector Machine

T

TI	Temporal Index
TS	Time Saving
TU	Transform Unit

U

UHD	Ultra-High-Definition

V

VCEG	Video Coding Experts Group
VR	Virtual Reality

W

WebRTC	Web Real-Time Communication

English summary

Imagine that you put on your virtual reality (VR) glasses to transport yourself and your living room couch to the spectator stands of a sports match. Now imagine thousands of other people having the same idea. For the people who have to film the match on location and transport this to the viewers at home, this is no dream. Instead, it is a nightmare. After all, they need to provide an appropriate video to each user sitting at home, and also to people who are on the go and want to follow the match on their smartphones. Forget for a moment about the future with VR-glasses, and come back to the present, where you have so many different devices with all kinds of different specifications, and people with all kinds of different internet connections. At this point, content providers might actually start to long for the past.

After all, nowadays, video is ubiquitous. Whereas video was only displayed on television sets and in movie theaters in the past, it is now found on smartphones, laptops, tablets, VR-glasses, and numerous other displays. In order to bring video to these devices, encoded video bitstreams are typically transported over networks with different bandwidth capacities. However, due to this heterogeneous environment containing different devices and networks, the videos have to be adapted to fit the circumstances. For example, when using VR glasses to look around in a high-resolution 360-degree video, sending the entirety of the video to the viewer consumes a large amount of bandwidth. However, since the user only sees a limited view of the 360-degree video at a time, sending all of it is a waste of bandwidth. Therefore, the video should be adapted to only transmit the region of interest that is being watched by the viewer at that time.

Adapting video in order to encode it and send it to a viewer can be done in two ways: either by using a fixed number of encoders, or by using completely personalized encoders. Using the first method, the content provider encodes the video only at certain spatial resolutions and with certain bit rates. In the above example of 360-degree video, the video is also divided into separately encoded tiles, from which only the tiles inside the region of interest are transmitted to the user. The advantage of using a fixed amount of encoders is that the computational complexity of the system remains fixed, independent of the number of viewers. However, if the network circumstances change, the viewer might need a lower-quality version of the same video that consumes less bandwidth. In the case of a fixed amount of encoders, such a change is only possible at the start of a video segment, which typically lasts between two to ten seconds depending on the application. As a result,

switching to a different version results in latency, which will negatively impact the experience of users in interactive applications.

In contrast, if the content provider opts for a system with completely personalized encoders, each user receives a video that optimally adapts to the properties of his device and network. As a result, if the available bandwidth decreases, the bit rate of the encoded video is decreased as well, meaning that no latency is introduced, since there is no need to switch to a different version of the video. However, since each individual user requires a computationally complex encoder, the overall complexity of the system will increase drastically as more users connect to the system. Consequently, this solution is currently used only when the amount of expected users is small.

In order to make such a personalized system more feasible, one could attempt to use fast encoding algorithms to reduce the computational complexity of each individual encoder. However, as is shown in Chapter 2 of this dissertation for High Efficiency Video Coding (HEVC), the methods proposed in the state-of-the-art are only able to reduce the computational complexity of the reference encoding software by less than 75%. This means that, in the best-case scenario, four fast encoders operate at the same computational cost as a single non-accelerated encoder. As such, the complexity reduction provided by these algorithms is insufficient for scenarios with a large amount of users.

However, contrary to the state-of-the-art algorithms for fast encoding, a system that encodes personalized video bitstreams has the advantage that it contains multiple encoders that encode different versions of the same video content. As such, instead of relying on statistical information generated during video encoding to make fast decisions, these personalized encoders can rely on guidance from coding decisions made by the other encoders in the system. In order to investigate the viability of a system with one encoder per individual, the use of this guided encoding has been studied in four different scenarios.

The first scenario, in Chapter 3, considers the problems that arise when creating a personalized video composition for each user. For example, in many industries, different video feeds are gathered in a control room and displayed together on a video wall. This composition also needs to be displayed on other devices in remote locations. Each of these devices should be able to rearrange the composition according to the preferences of the viewer. In order to provide each of these users with such a personalized composition, each different video composition is encoded and transmitted as a single bitstream. In order to reduce the complexity of the encoding of such a video, the coding decisions of each personalized encoding of a composition are guided by the decisions made by the encoding of the separate videos in the composition. However, if the individual videos are spatially misaligned with the grid of coded blocks of the composition, simply copying the coding information is not compression-efficient. Therefore, the coding decisions of the video in the composition are determined by predicting encoding decisions of misaligned sequences by using a trivial method or a more adaptive, computational-complexity scalable machine learning method. Using the trivial method, a complexity reduction of 82% is achieved with a bit rate overhead of

3% for a misalignment of 32 pixels. However, for other shifts, the performance of the machine learning method is better. As such, the proposed method succeeds in offering a fast-encoding solution for any amount of misalignment for a reasonable bit rate overhead.

Contrary to the first scenario in which multiple videos were combined into one, the second scenario as described in Chapter 4 considers a use case in which each viewer is presented with only a limited, personalized viewport from an ultra-high-resolution video in order to have a more immersive experience. For example, in a soccer match, one user might prefer to zoom in on his favorite players, whereas other users might instead want to follow the ball or have a general overview of the field. Since the entire ultra-high-resolution video may have a resolution of $10{,}000\times1880$ pixels or more, and since most displays only support resolutions of 1920×1080 pixels, sending the entire ultra-high-resolution video to each viewer would be a waste of bandwidth. Instead, each user should only receive a personalized crop from the ultra-high-resolution video. Since cropping such a region of interest from the full video can also result in misalignment as in the previous scenario, the cropped view is constrained to having the upper-left corner of the view match the block grid of the entire ultra-high-resolution video. Consequently, the effect of cropping can be isolated from the effect of misalignment, and the personalized view can be encoded by copying more coding information from the full video than was the case in the scenario of misalignment in compositions. By thus guiding the coding decisions of the personalized view with coding information extracted from the encoding of the entire video, complexity reductions between 96.5% and 97.5% are achieved for bit rate overheads between 8% and 20%. These overheads are still smaller than the overheads introduced by the traditional method for panoramic and 360-degree video delivery of having a fixed amount of encoders create separate encodings of different tiles. Moreover, since the personalized encoders do not suffer from the same structural latency as the tile-based method, the former could thus become a viable alternative for the latter.

The third scenario, in Chapter 5, then considers a case where the properties of the original video differ: one version of the video is filmed in high-dynamic-range (HDR), whereas the other version is optimized for display on televisions supporting only lower dynamic ranges (LDR). As such, the dynamic range of the video is personalized according to the capabilities of the receiving device. A video provider thus has to encode multiple versions of the same video, effectively multiplying the required computational complexity with the amount of different versions. However, when encoding both an HDR and LDR version of the video, the coding decisions made during the encoding of HDR can be used to guide the decisions of the LDR version, thus decreasing the encoding complexity of the latter version. Since both versions contain the same video content with only differences in pixel values, contrary to the cropped or misaligned content in the previous scenario, all coding information can be copied from one version to another. By then selecting the quantization parameter of the LDR encoding according to the model described in this dissertation, the correlation between the two versions is maximized, thus resulting in a smaller bit rate overhead. Following this model, the LDR version of

the video is encoded with a complexity reduction of 99.7% while achieving a bit rate overhead of 12.4%. In other words, using guided encoding, a video provider can simultaneously encode multiple dynamic-range versions of the same video for about the same computational complexity as when encoding a single version.

Whereas previous scenarios considered changes to the input video such as different spatial misalignments, different crops, and different dynamic ranges, the fourth and final scenario, which is described in Chapter 6, considers the use of different encoding parameters to produce bitstreams encoded at different bit rates. For example, in real-time live-streaming use cases such as virtual classrooms and video conferences, each user has a different amount of available bandwidth depending on his own network conditions. If a fixed amount of encoders are then used, each user is connected to an encoder that encodes a version of the video with a lower bit rate than the available bandwidth. Consequently, the bandwidth of the client will not be utilized to its full capacity, meaning that the provided bitstream is not necessarily encoded with the highest quality that the bandwidth capacity of the user could handle. Moreover, clients with fluctuating bandwidth will experience extra latency due to constant switching between video streams. In order to avoid such a scenario, each user should be provided with a personalized encoder that generates a bit rate version that best matches the current bandwidth of the client. Since the computational complexity of such an unoptimized system increases linearly with the amount of clients, the work in this dissertation focuses on greatly reducing the amount of complexity that each client adds to the system. In order to realize this, the numerous personalized encoders are guided by the coding decisions calculated by dedicated modules. These modules determine the optimal coding decisions when encoding the video at certain fixed bit rates. The guided encoders then only have to encode the residual picture of the video by copying all coding decisions generated by one of these modules. By creating a system with only six coding information calculation modules covering the desired bit rate range, the system can provide individual encoders with a complexity reduction of 99.2% with the average bit rate overhead of the system being 11.8%. As a result, one hundred extra users joining the proposed system results in a smaller increase of overall computational complexity than one extra user joining a similar system that does not use guided encoding.

By considering the above four scenarios, it can be concluded that guided encoding greatly accelerates the encoding of personalized video streams for individual users. Depending on the scenario, the coding information between encoders can be shared more efficiently, with cropping and misalignment introducing the greatest differences in coding information between encoders. In contrast, small changes in pixel values and bit rate allow the creation of a system in which personalized encoders can completely share all coding information with only small penalties in the form of bit rate overhead. Therefore, it would be interesting for future standardization of video compression to take these findings into account by encoding coding decisions and the residual picture as separate parts of the bitstream in order to facilitate the use of such guided encoding.

Nederlandse samenvatting –Summary in Dutch–

Stel je voor dat je jouw virtual-reality (VR) bril opzet om jezelf en de zetel in jouw woonkamer te transporteren naar de tribune van een sportwedstrijd. Stel je nu ook voor dat duizenden andere mensen hetzelfde idee krijgen. Voor diegenen die de match ter plaatse moeten filmen en deze naar de kijkers thuis moeten transporteren, is dit geen droom, maar een nachtmerrie. Zij moeten tenslotte een passende video bezorgen aan elke gebruiker die thuis zit, alsook aan de mensen op straat en op het openbaar vervoer die de wedstrijd op hun smartphones willen volgen. Vergeet even de toekomst met VR-brillen en keer terug naar het heden, waar al zovele verschillende toestellen met talloze verschillende specificaties bestaan en waar mensen achter zoveel verschillende soorten internetverbindingen zitten. Dat is het moment waarop sommige contentproviders met een verlangende blik zouden durven verlangen naar het eenvoudige verleden.

In tegenstelling tot vroeger, treft men namelijk tegenwoordig video overal aan. Waar video vroeger enkel op televisies en in bioscopen getoond werd, is het nu terug te vinden op smartphones, laptops, tablets, VR-brillen, en tal van andere beeldschermen. Om video naar deze apparaten te brengen, worden gecodeerde videostromen gewoonlijk getransporteerd over netwerken met verschillende capaciteiten op gebied van bandbreedte. Echter, wegens de heterogeniteit van deze omgeving die verschillende toestellen en netwerken omvat, moeten de video's aangepast worden naargelang de omstandigheden. Bijvoorbeeld, als iemand met een VR bril rondkijkt in een hoge-resolutie 360-gradenvideo, kost het versturen van deze volledige video naar de kijker een enorme hoeveelheid bandbreedte. Echter, aangezien de gebruiker op elk moment slechts een beperkte regio van de 360-gradenvideo in zijn gezichtsveld waarneemt, is het versturen van de volledige video een verspilling van bandbreedte. Daarom moet de video zo aangepast worden dat enkel het huidige interessegebied naar de kijker wordt verstuurd.

Het afstemmen van video om deze vervolgens te coderen en naar een kijker te versturen, kan op twee manieren: hetzij met een vast aantal encoders, hetzij door volledig gepersonaliseerde encoders. Met de eerste methode codeert de contentprovider de video slechts voor enkele spatiale resoluties en bitsnelheden. In het bovenstaande voorbeeld van 360-gradenvideo wordt de video bovendien ook onderverdeeld in afzonderlijk gecodeerde tegels, waarbij enkel de tegels die binnen het interessegebied liggen, naar de gebruiker verstuurd worden. Het voordeel van zo een beperkte hoeveelheid encoders te gebruiken, is dat de computationele

complexiteit van het systeem vast blijft, onafhankelijk van het aantal kijkers. Echter, als de omstandigheden van het netwerk veranderen, is het mogelijk dat de gebruiker een minder kwaliteitsvolle versie van dezelfde video nodig heeft omdat zijn beschikbare bandbreedte gedaald is. In het geval van een vast aantal encoders is een dergelijke sprong naar een andere versie enkel mogelijk aan het begin van een videosegment. Deze segmenten hebben – afhankelijk van de toepassing – gewoonlijk een tijdsduur van twee á tien seconden, waardoor bij het verspringen naar een andere versie een vertraging ontstaat, die een negatieve impact heeft op de gebruikerservaring bij interactieve toepassingen.

Als daarentegen de contentprovider kiest voor een systeem met volledig gepersonaliseerde encoders, zal elke gebruiker een video ontvangen die steeds optimaal afgestemd is op de eigenschappen van zijn toestel en het netwerk. Hierdoor zal dus in het geval van een daling van de beschikbare bandbreedte de bitsnelheid van de video die gecodeerd wordt, meedalen. Dit betekent bijgevolg dat er geen vertraging ontstaat, aangezien het in dit geval niet nodig is om naar een andere versie van de video te verspringen. Echter, aangezien elke individuele gebruiker een computationeel complexe encoder vereist, zal de gehele complexiteit van het systeem drastisch stijgen naarmate meer gebruikers verbinding maken met het systeem. Derhalve wordt deze oplossing momenteel enkel gebruikt wanneer het verwachte aantal gebruikers klein is.

Om het gebruik van een dergelijk gepersonaliseerd systeem haalbaar te maken, zou men algoritmes voor snelle codering kunnen gebruiken om de computationele complexiteit van elke afzonderlijke encoder te reduceren. Echter, in Hoofdstuk 2 van dit proefschrift blijkt voor High Efficiency Video Coding (HEVC) dat de methodes die voorgesteld worden in de recentste literatuur, slechts in staat zijn tot het bereiken van een complexiteitsreductie van minder dan 75% voor de referentie-encoding software. Dit betekent dat, in het beste geval, vier snelle encoders dezelfde rekentijd vereisen als een niet-versnelde encoder. Als zodanig is de complexiteitsreductie die bereikt wordt door deze algoritmes, onvoldoende voor scenario's met een groot aantal gebruikers.

In tegenstelling tot technieken voor snelle codering voor afzonderlijke encoders, heeft een systeem met gepersonaliseerde video-encoders het voordeel dat het verschillende encoders bevat die elk een verschillende versie van dezelfde beeldinhoud comprimeren. Als zodanig kunnen deze gepersonaliseerde encoders rekenen op sturing van codeerbeslissingen gemaakt door andere encoders in het systeem voor het maken van hun eigen codeerbeslissingen. Om de haalbaarheid van een systeem met één encoder per individu te onderzoeken, is het gebruik van gestuurde codering onderzocht voor vier verschillende scenario's.

Het eerste scenario, in Hoofdstuk 3, neemt de problemen onder de loep die zich voordoen bij het creëren van een gepersonaliseerde videocompositie voor elke gebruiker. Dit gebeurt bijvoorbeeld in industriële applicaties waar verschillende videobronnen verzameld worden in een controlekamer en daar op een video wall worden getoond. Deze compositie moet dan ook nog op andere toestellen op andere locaties verschijnen. Op elk van deze apparaten moet het echter mogelijk zijn voor de gebruiker om zijn eigen compositie samen te stellen naargelang zijn

voorkeur. Om iedere gebruiker met een dergelijke gepersonaliseerde compositie te voorzien, wordt elke verschillende videocompositie gecodeerd en verstuurd als een enkele bitstroom. Om de codeercomplexiteit van deze video te reduceren, worden de codeerbeslissingen van elke gepersonaliseerde codering van een compositie gestuurd door de beslissingen gemaakt tijdens het coderen van de afzonderlijke video's in de compositie. Echter, als deze individuele video's niet spatiaal gealigneerd zijn met het raster van gecodeerde blokken van de compositie, is het rechtstreeks kopiëren van de codeerinformatie niet compressie-efficiënt. Daarom worden de codeerbeslissingen van de video in de compositie bepaald door codeerbeslissingen van de niet-gealigneerde sequenties te voorspellen met behulp van een triviale methode of een meer adaptieve machine learning methode die schaalbaarheid biedt op gebied van computationele complexiteit. Bij het gebruik van de triviale methode wordt een complexiteitsreductie van 82% bereikt met een overhead van 3% in bitsnelheid indien een sequentie 32 pixels verkeerd gealigneerd is. Voor andere verschuivingen van het beeld presteert de machine learning methode beter. Als zodanig slaagt de voorgestelde werkwijze erin om voor een bescheiden overhead van bitsnelheid een oplossing te bieden voor het snel coderen van elke hoeveelheid van foutieve alignatie.

In tegenstelling tot het eerste scenario waarin meerdere video's werden samengevoegd tot één enkele compositie, beschouwt het tweede scenario, beschreven in Hoofdstuk 4, een use case waarin elke kijker voorzien wordt van een beperkte, persoonlijke weergave uit een ultra-hoge-resolutie video met de bedoeling om een meer immersieve ervaring te beleven. Bijvoorbeeld, in een voetbalwedstrijd kan het zijn dat de ene gebruiker verkiest om in te zoomen op zijn favoriete spelers, terwijl andere gebruikers de bal willen volgen of een algemeen overzicht van het speelveld verkiezen. Aangezien de volledige ultra-hoge-resolutie video een resolutie kan hebben van 10.000×1880 pixels of meer en aangezien de meeste beeldschermen slechts een resolutie van 1920×1080 kunnen tonen, is het versturen van de volledige ultra-hoge-resolutie video naar alle gebruikers een verspilling van bandbreedte. In plaats daarvan is het beter dat elke gebruiker slechts een gepersonaliseerde uitsnijding uit de ultra-hoge-resolutie video ontvangt. Aangezien het uitsnijden van een dergelijk interessegebied uit een volledige video ook kan leiden tot een foutieve alignatie zoals in het vorige scenario, wordt de beperking opgelegd dat de linkerbovenhoek van de uitsnijding moet samenvallen met het blokraster van de volledige ultra-hoge-resolutie video. Bijgevolg kan het effect van het bijsnijden geïsoleerd worden van het effect van foutieve alignatie en kan de gepersonaliseerde weergave gecodeerd worden door het kopiëren van meer codeerinformatie uit de volledige video dan wat het geval was in het scenario met foutieve alignatie in composities. Door op deze manier de codeerbeslissingen van de persoonlijke weergave te sturen met codeerinformatie geëxtraheerd uit de codering van de volledige video, worden complexiteitsreducties tussen 96,5% en 97,5% bereikt voor overheads in bitsnelheden tussen 8% en 20%. Deze overheads zijn nog steeds kleiner dan de overheads die gepaard gaan met het gebruik van de traditionele oplossing voor levering van panoramische en 360-graden video waarbij een vast aantal encoders het volledige beeld coderen als afzonderlijke tegels. Aangezien de gepersonaliseerde

encoders bovendien vrij zijn van de structurele vertraging van de tegelgebaseerde methode, kunnen deze gepersonaliseerde encoders in de toekomst een alternatief voor deze laatste worden.

Het derde scenario, in Hoofdstuk 5, beschouwt dan een geval waarin de eigenschappen van de originele video verschillen: een versie van de video is gefilmd met een hoog dynamisch bereik (HDR), terwijl de andere versie geoptimaliseerd is voor weergave op televisies die enkel lagere dynamische bereiken (LDR) ondersteunen. Als zodanig wordt het dynamisch bereik van de video gepersonaliseerd naargelang de capaciteiten van het doeltoestel en moet een video provider dus verschillende versies van dezelfde video coderen, wat dus de vereiste computationele complexiteit vermenigvuldigt met het aantal verschillende versies. Echter, wanneer zowel een HDR als LDR versie van dezelfde video gecodeerd worden, kunnen de codeerbeslissingen die gemaakt worden tijdens het coderen van HDR, gebruikt worden om de beslissingen van de LDR versie te sturen, waardoor de codeercomplexiteit van deze laatste sterk vermindert. Aangezien beide versies dezelfde video-inhoud bevatten met enkel verschillen in pixelwaarden, kan – in tegenstelling tot de uitgesneden of foutief gealigneerde inhoud uit de vorige scenario's – alle codeerinformatie van één versie naar de andere gekopieerd worden. Door vervolgens de kwantisatieparameter van de LDR codering te kiezen volgens het model beschreven in dit proefschrift, wordt de correlatie tussen de twee versies gemaximaliseerd, waardoor de overhead van de bitsnelheid verkleind wordt. Door dit model te gebruiken, wordt de LDR versie van de video gecodeerd met een complexiteitsreductie van 99,7%, terwijl de overhead van bitsnelheid 12,4% is. Met andere woorden, met behulp van gestuurde codering kan een video provider meerdere versies van dezelfde video met een verschillend dynamisch bereik gelijktijdig coderen voor dezelfde computationele complexiteit als voor het coderen van één enkele versie.

Daar waar de vorige scenario's veranderingen van de invoervideo behandelden zoals foutieve alignering, verschillende uitsnijdingen en verschillende dynamische bereiken, beschouwt het vierde en laatste scenario dat beschreven wordt in Hoofdstuk 6, het gebruik van verschillende codeerparameters om bitstromen te produceren die gecodeerd zijn met verschillende bitsnelheden. Bijvoorbeeld, in use cases waarbij real-time live-streaming vereist is, zoals het geval is in virtuele klaslokalen en videoconferenties, heeft iedere gebruiker een verschillende hoeveelheid beschikbare bandbreedte naargelang de toestand van zijn eigen netwerk. Als er dan een vaste hoeveelheid encoders gebruikt wordt, wordt elke gebruiker verbonden met een encoder die een versie van de video codeert die een lagere bitsnelheid heeft dan de beschikbare bandbreedte. Bijgevolg wordt de bandbreedte van de cliënt niet ten volle benut, wat betekent dat de geleverde bitstroom niet noodzakelijk gecodeerd is met de hoogste kwaliteit die ondersteund wordt door de gebruiker. Bovendien zullen cliënten met een fluctuerende bandbreedte extra vertraging ondervinden door het constant wisselen tussen videostromen. Om een dergelijk scenario te voorkomen, moet iedere gebruiker voorzien worden van een gepersonaliseerde encoder die een versie genereert waarvan de bitsnelheid het best overeenkomt met de huidige bandbreedte van de cliënt. Aangezien de computatio-

nele complexiteit van een dergelijk niet-geoptimaliseerd systeem lineair stijgt met het aantal cliënten, concentreert het werk in dit proefschrift zich op een extreme reductie van de complexiteit die iedere cliënt aan het systeem toevoegt. Om dit te bereiken worden de talrijke gepersonaliseerde encoders gestuurd door de codeerbeslissingen die berekend worden door gespecialiseerde modules. Deze modules bepalen de optimale codeerbeslissingen wanneer de video aan zekere, vaste bitsnelheden wordt gecodeerd. De gestuurde encoders moeten dan enkel nog het residuele beeld van de video coderen door alle codeerbeslissingen die gegenereerd worden door één van deze modules, te kopiëren. Door een systeem te creëren met slechts zes codeerinformatieberekeningsmodules die het volledige gewenste bereik van bitsnelheden omvatten, kan het systeem individuele encoders voorzien met een complexiteitsreductie van 99,2% met een gemiddelde overhead van bitsnelheid van 11,8% voor het gehele systeem. Hierdoor resulteert het toevoegen van honderd extra gebruikers aan het systeem voor een lagere stijging van totale computationele complexiteit dan het toevoegen van één extra gebruiker aan een soortgelijk systeem dat geen gestuurde codering gebruikt.

Door de bovenstaande vier scenario's in acht te nemen, kan er geconcludeerd worden dat gestuurde codering het mogelijk maakt om videostromen die gepersonaliseerd worden voor elke individuele gebruiker, veel sneller te coderen. Afhankelijk van het scenario kan de codereerinformatie tussen encoders efficiënter gedeeld worden. Hierbij zorgen uitsnijding en foutieve alignatie voor de grootste verschillen in codeerinformatie tussen encoders. Daartegenover laten kleine veranderingen in pixelwaarden en bitsnelheid het toe om een systeem te creëren waarbij gepersonaliseerde encoders alle codeerinformatie volledig kunnen delen met slechts een kleine boete in de vorm van overhead van bitsnelheid als gevolg. Gezien deze resultaten zou het dus interessant zijn voor toekomstige standaardisatie op gebied van videocompressie om deze bevindingen in rekening te brengen door codeerbeslissingen en het residueel beeld als aparte delen van de bitstroom te coderen met het oog op het vereenvoudigen van het gebruik van gestuurde codering.

1 Introduction

Imagine that you put on your virtual reality (VR) glasses to transport yourself and your living room couch to the spectator stands of a sports match. Now imagine thousands of other people having the same idea. For the people who have to film the match on location and transport this to the viewers at home, this is no dream. Instead, it is a nightmare. After all, they need to provide an appropriate video to each user sitting at home, and also to people who are on the go and want to follow the match on their smartphones. Forget for a moment about the future with VR-glasses, and come back to the present, where you have so many different devices with all kinds of different specifications, and people with all kinds of different internet connections. At this point, content providers might actually start to long for the past.

After all, in the past, people watched video only in movie theaters and on television sets. These simple videos were delivered to the televisions over traditional broadcast networks. However, as the years progressed, the internet was born and expanded quickly. Combined with the advent of devices such as laptops, smartphones, tablets, VR-glasses, and many more devices with displays, this expansion of the internet gave rise to a myriad of ways to view video, such as streaming videos from video sharing sites, watching live video streams of events, participating in interactive remote classrooms, making video calls to other people, and playing video games on remote devices. As a result, video now has an even stronger presence in the lives of people than ever before.

Since the devices on which video is consumed and the networks over which video is transported have become very diverse, so have the requirements for the

properties of the video itself. For example, as a higher bit rate is required in order to transfer a high-resolution video, it would be a waste of bandwidth capacity if a high-resolution video is transported over the network to a low-resolution display. A version with a low resolution should be transmitted instead. Similarly, if a television set does not support High-Dynamic-Range (HDR) video, the video sent to this display should be adapted to the capabilities of this device in order to provide the best quality of experience. The way in which the video is encoded should thus be adapted to the user depending on his device and network capabilities, providing him with an optimal, personalized experience.

1.1 Approaches to video content delivery

Content providers have two main ways to adapt the encoding of a video to the user: by using a fixed number of encoders (using adaptive streaming techniques), or by using completely personalized encoders to provide an individual bitstream to each user by using a protocol such as Web Real-Time Communication (WebRTC). The former, adaptive streaming method is traditionally used by content providers with many viewers. By providing a fixed number of different bitstream representations of an encoded video, each user will receive the version that is best suited for his situation. An example of such different bitstream representations is shown in Figure 1.1, where five bitstream representations are available, spread across three different video resolutions of 720p (1280×720 pixels), 1080p (1920×1080 pixels), and 2160p (3840×2160 pixels). Depending on their bandwidth capacity, each user receives an appropriate representation with a bit rate below their bandwidth capacity. If the available bandwidth changes while watching the video (e.g. due to varying network conditions), the user is switched to another, more suitable representation. For example, in Figure 1.1, if a user receiving a video with a 1080p resolution at a bit rate of 15 Mbps has his connection dropping to 12 Mbps, he will be switched to the 10 Mbps version instead. Note, however, that this switching is only possible at the start of a new video segment, which typically has a duration between two and ten seconds depending on the application.

This technique of adaptive streaming can be expanded further to also provide both HDR and Low-Dynamic-Range (LDR) versions of the same video in order to adapt to the dynamic range capabilities of the device of the user. Furthermore, adaptive streaming can also be used by dividing an ultra-high-resolution 360-degree or panoramic video into independently encoded tiles in order to send only the appropriate tiles to the users, since the actual viewport of users is typically smaller than the entire video in such applications.

Each extra bitstream representation that must be created for adaptive streaming comes at a large computational cost. However, the greatest advantage of this technique is that the amount of representations that are created is known on before-

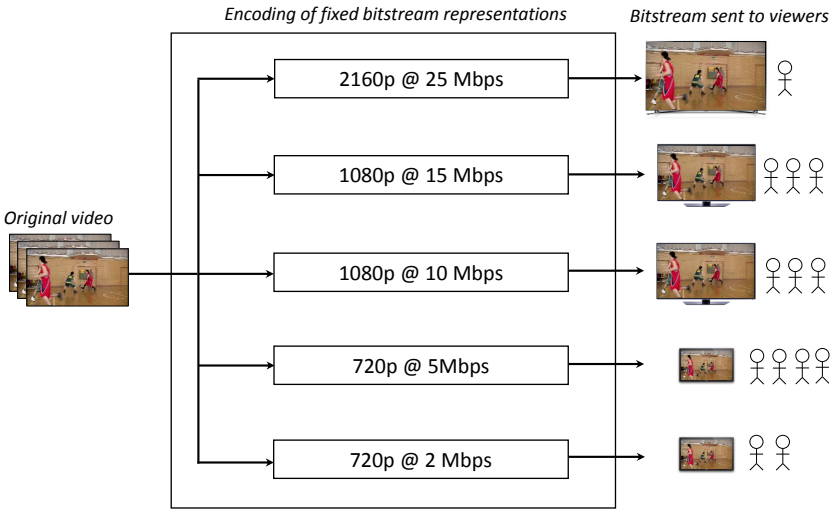


Figure 1.1: In the case of fixed bitstream representations, the amount of video encoders is limited and multiple clients receive the exact same bitstream.

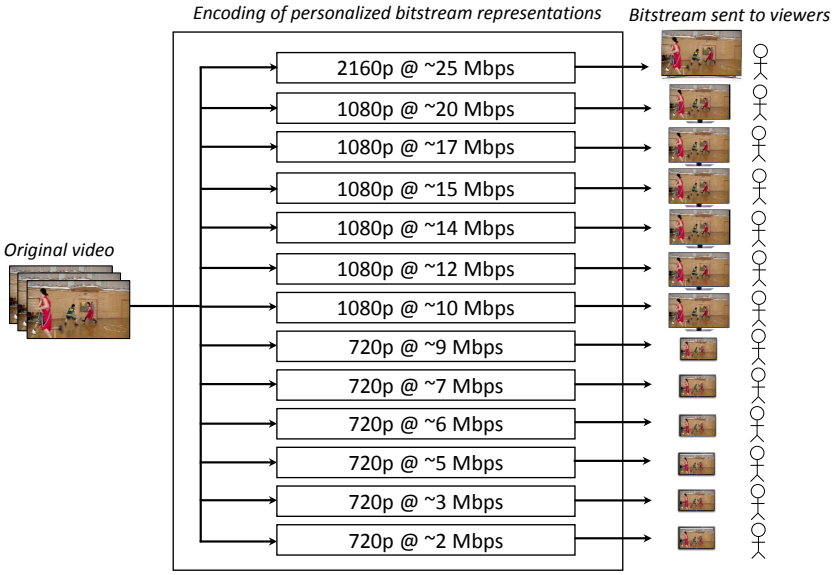


Figure 1.2: In the case of personalized bitstream representations, the amount of video encoders equals the number of clients. Each client receives a bitstream tailored to his needs.

hand and thus fixed, resulting in a system that scales well. After all, the content provider has full control over the number of computationally expensive encoders that he provides, no matter how many viewers connect to the system.

In contrast, as seen in Figure 1.2, the method using completely personalized encoders requires a number of encoders equal to the number of current viewers. As a result, if the network conditions of a user change, the target bit rate of the encoder is automatically adjusted in order to match the new conditions. Similarly, in the case of 360-degree or panoramic video, only the current view is transmitted to the user, which for example results in a video with a bit rate of 4.2 Mbps instead of a full 360-degree video of 23.4 Mbps, which is a reduction of the required bandwidth by more than 80%[1]. The greatest advantage of such a personalized approach compared to a fixed amount of encoders is the low latency of the system, since users do not need to wait for the beginning of a new segment until they can switch to a different bitstream representation. Additionally, it can also be used in situations where the amount of possible representations is very high, such as in the case when each user needs to receive their own video composition.

However, as is also seen in Figure 1.2, the amount of encoders for the same amount of users as in Figure 1.1 has increased dramatically. Consequently, personalized encoders are currently only used in practice when the maximum amount of clients is expected to be very small. This is for example the case in an industrial context where different compositions of videos have to be created depending on the device on which they have to be displayed (see also Chapter 3), or in low-latency communications such as video conferencing with a very small amount of users. In order to make personalized encoders more feasible, the computational complexity of each individual encoder should thus be decreased.

1.2 Guided encoding for personalized video

At its root, a video encoder is a piece of software or hardware that takes a video as an input and compresses it into a bitstream. As seen in Figure 1.3, such a compressed bitstream consists of coding decisions (such as block partitionings and motion vectors in the case of block-based video compression standards), and residual pictures, which partially compensate for the difference between original frames and the frames predicted by the coding decisions. If more bits are assigned to the residual, the resulting video quality will be higher, but the compression ratio of the resulting bitstream will be lower, whereas applying stronger quantization to the residual (and thus assigning less bits), results in a lower video quality, but also in a more compressed bitstream.

[1] These numbers have been obtained using the KiteFlite_8192x4096_30fps video sequence, with a quantization parameter of 28 for both the full video as well as for a 1080p crop of the center, using the x265 encoder software with the slow preset.

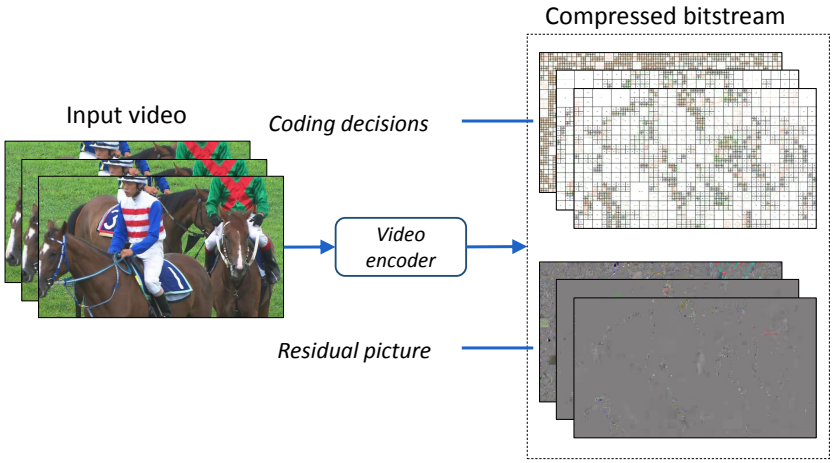


Figure 1.3: When an input video is encoded by a video encoder, it is compressed as a combination of coding decisions and residual pictures.

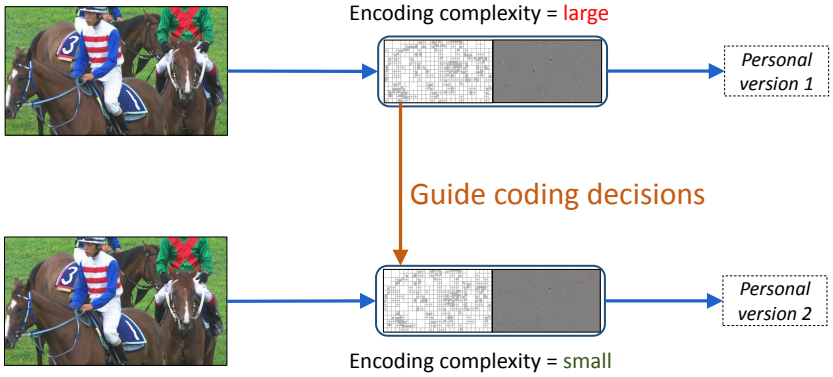


Figure 1.4: A video encoder is very computationally complex. However, when many personal versions of a similar video are created, the coding decisions of one video encoder can be guided by the decisions of another encoder, thus greatly reducing the computational complexity of the guided encoder.

In order to have a higher video quality for the same compression ratio, it is important for a video encoder to make coding decisions that will result in the smallest difference between the original frames and the predicted frames. However, obtaining these optimal coding decisions is a very computationally intensive task, hence content providers prefer to limit the amount of encodings using techniques such as adaptive streaming as mentioned in the previous section. If they supplied each viewer with a personalized video instead, the amount of required video encoders, and consequently the total computational complexity of their system as well, would increase greatly with each user.

However, since different versions of the same video are all still based on the same original, there remains a similarity between the coding decisions used during the encoding of each personalized bitstream. As such, the part of a video encoder that calculates the coding decisions can be accelerated by exploiting this similarity between the coding decisions of encodings of different personalized versions. In this dissertation, this technique is called guided encoding, as illustrated in Figure 1.4. By using this technique in order to greatly reduce the computational complexity of personalized encoders, providing each user with a personalized bitstream becomes more feasible.

1.3 Outline

As mentioned in the previous section, this dissertation focuses on reducing the computational complexity of personalized encoders by exploiting the similarity between coding decisions of similar encodings. All techniques mentioned in this dissertation can potentially be adapted to most modern block-based video compression standards such as Advanced Video Coding (H.264/AVC), VP9, and High Efficiency Video Coding (HEVC). Although H.264/AVC is currently the most widely used standard, its successor, HEVC, was standardized in 2013. Since this standard was created with very-high resolution video in mind, and was also extended to support HDR, both which feature in one of the use cases in this dissertation, the main focus of this dissertation lies on HEVC. However, others have already performed much research in order to reduce the complexity of video encoding with HEVC in the case of having only a single encoder. Therefore, Chapter 2 gives a high-level overview of the coding decisions made by an HEVC encoder and the performance of the state-of-the-art HEVC fast-encoding algorithms.

Chapter 2 is followed by the main body of this dissertation. In this main body, four different personalization scenarios are investigated in which similarity between different bitstreams is exploited to guide encoding decisions. The first scenario, in Chapter 3, focuses on spatially misaligned video sequences. In this chapter, a video is spatially shifted by several pixels since it has to be inserted into a personalized video composition. As illustrated in Figure 1.5, the similarity between the original and shifted video is then exploited in order to guide and accelerate the encoding of the shifted version.

The second scenario discussed in this dissertation is described in Chapter 4 and focuses on ultra-high resolution video. Since each user might be looking at a different location in the ultra-high resolution video, they each receive a personalized, cropped view. However, providing a separately encoded view for each user is a computationally expensive nightmare. Therefore, as seen in Figure 1.6, the encoding of these personalized views is accelerated by exploiting the correlation with the coding information of the corresponding region from the full encoding of the ultra-high resolution video.

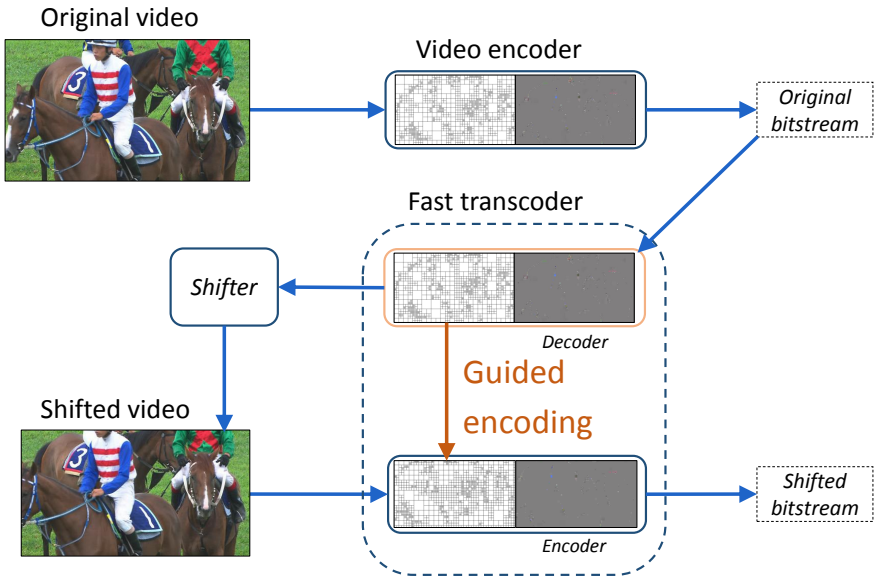


Figure 1.5: Guided encoding is applied to transcoding of a shifted picture by using the coding information obtained during decoding of the original bitstream to accelerate the encoding of the shifted video.

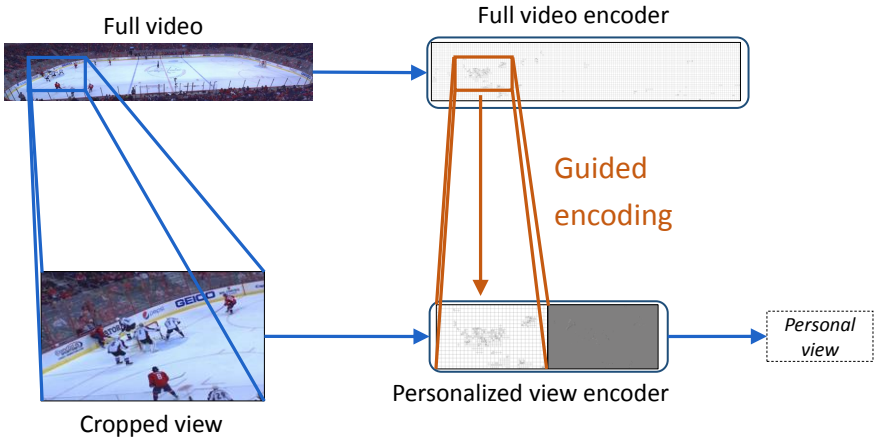


Figure 1.6: Guided encoding is applied to the encoding of personalized, cropped views by using the coding information of the corresponding region from the full video encoder to accelerate the personalized view encoder.

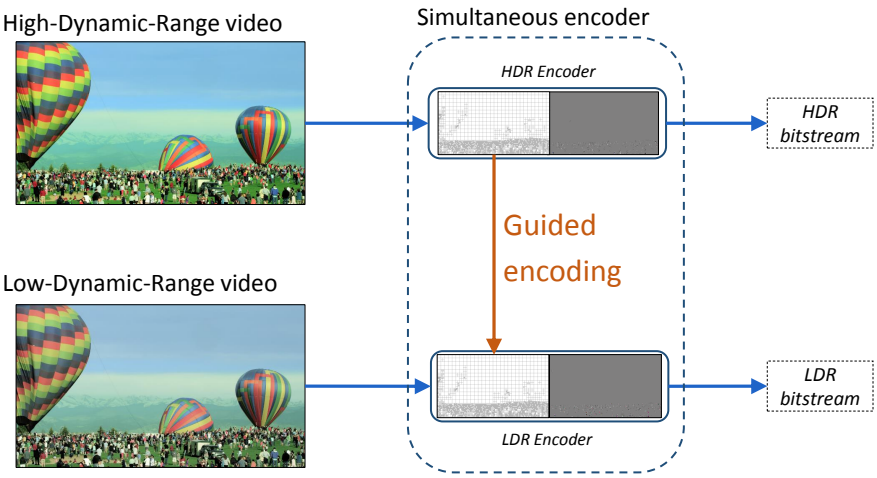


Figure 1.7: Guided encoding is applied to the simultaneous encoding of both an HDR and LDR version of the same video by using the coding information of the HDR version to also encode the LDR version.

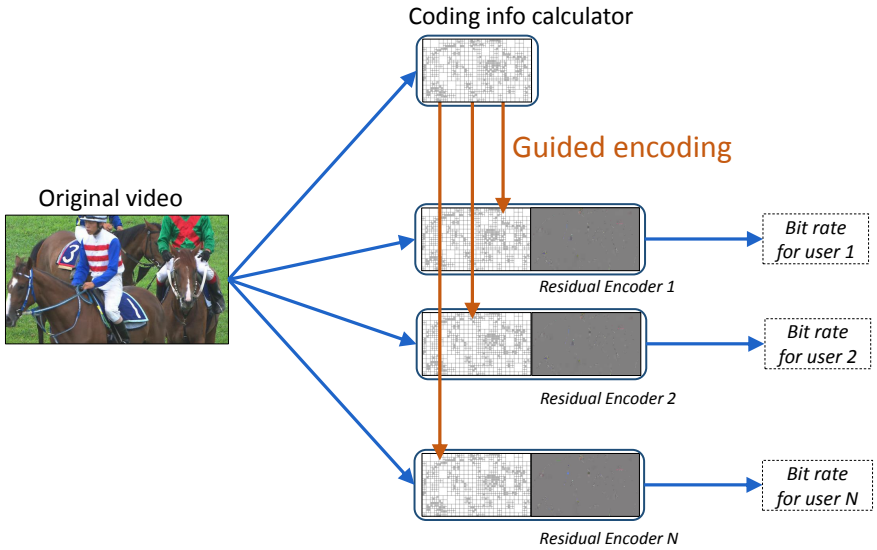


Figure 1.8: Guided encoding is applied to personalized bitstream representations by using dedicated coding information calculators that generate coding information for many residual encoders. These encoders attain the required bit rate by encoding the the residual picture with different quantization parameters.

Whereas the first two scenarios both focus on a spatial displacement of video (respectively spatial shifting and cropping), the third scenario considers different versions of the source video that have to be encoded. These source videos differ in terms of dynamic range, with one video having a higher dynamic range while the other has a lower dynamic range. Instead of separately encoding both an HDR version and LDR version of the video, the technique described in Chapter 5 investigates the correlation between the encodings of these different versions. By exploiting this correlation, a simultaneous encoder is created as shown in Figure 1.7, which guides the encoding decisions for the version with a lower dynamic range by using the coding decisions of the HDR version.

Finally, in Chapter 6, the fourth scenario focuses on providing a personalized bitstream representation which adapts to the current network condition of each user. As such, each user requires a personalized encoder that encodes the video at a bit rate similar to the available network bandwidth. In order to realize such a system, as illustrated in Figure 1.8, coding information is calculated by dedicated coding information calculators. These calculators pass the coding information to residual encoders, which have a very low computational complexity. These residual encoders then adapt to the bit rate of the user to which they are paired by adapting the quantization parameter used to quantize the residual picture.

This dissertation then comes to a close with the conclusion in Chapter 7.

1.4 Publications

1.4.1 Publications in international journals

1. L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Performance analysis of machine learning for arbitrary downsizing of pre-encoded HEVC video*. IEEE Trans. Consum. Electron., 61(4):507–515, Nov. 2015.

2. A. J. Diaz-Honrubia, J. De Praeter, G. Van Wallendael, J. L. Martinez, P. Cuenca, J. M. Puerta, and J. A. Gamez. *CTU splitting algorithm for H.264/AVC and HEVC simultaneous encoding*. J. Supercomputing, (online first: Feb. 2016).

3. 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 Trans. Multimedia, 18(3):364–378, Mar. 2016.

4. J. De Praeter, H. Swimberghe, G. Renard, G. Van Wallendael, and P. Lambert. *Dynamic encoder profile optimisation for real-time video streaming applications*. Electron. Lett., 52(13):1116–1118, June 2016.

5. A. J. Diaz-Honrubia, J. De Praeter, J. L. Martinez, P. Cuenca, and G. Van Wallendael. *Reducing the Complexity of a Multiview H.264/AVC and HEVC Hybrid Architecture*. J. Signal Process. Syst., (online first: June 2016).

6. J. De Praeter, G. Van Wallendael, T. Vermeir, J. Slowack, and P. Lambert. *Spatially misaligned HEVC transcoding with computational-complexity scalability*. J. Visual Commun. Image Representation, 40, Part A:149 – 158, Oct. 2016.

7. J. De Praeter, A. J. Diaz-Honrubia, T. Paridaens, G. Van Wallendael, and P. Lambert. *Simultaneous Encoder for High-Dynamic-Range and Low-Dynamic-Range Video*. IEEE Trans. Consum. Electron., 63(4):pp, Nov. 2016.

8. J. De Praeter, G. Van Wallendael, J. Slowack, and P. Lambert. *Video Encoder Architecture for Low-Delay Live-Streaming Events*. submitted to IEEE Trans. Multimedia.

1.4.2 Publications in international conferences

1. J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle. *Efficient Picture-in-Picture Transcoding for High Efficiency Video Coding*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 514–515, Sept. 2013.

2. 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 Proc. IEEE Int. Conf. Image Process. (ICIP), pages 2494–2498, Oct. 2014.

3. L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Machine learning for arbitrary downsizing of pre-encoded video in HEVC*. In Proc. IEEE Int. Conf. Consum. Electron. (ICCE), pages 406–407, Jan. 2015.

4. N. Van Kets, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Fast encoding for personalized views extracted from beyond high definition content*. In Proc. IEEE Int. Conf. Broadband Multimedia Syst. and Broadcast. (BMSB), pages 1–7, June 2015.

5. L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Out-of-the-loop information hiding for HEVC video*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 3610–3614, Sept. 2015.

6. A. J. Diaz-Honrubia, J. De Praeter, S. Van Leuven, J. De Cock, J. L. Martinez, and P. Cuenca. *Using Bayesian classifiers for low complexity multiview H.264/AVC and HEVC hybrid architecture*. In Proc. IEEE Int. Conf. Mach. Learn. Signal Process. (MLSP), pages 1–6, Sept. 2015.

7. J. De Praeter, A. J. Diaz-Honrubia, N. Van Kets, G. Van Wallendael, J. De Cock, P. Lambert, and R. Van de Walle. *Fast simultaneous video encoder for adaptive streaming*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 1–6, Oct. 2015.

8. L. Pham Van, J. De Praeter, G. Van Wallendael, P. R. Alface, and P. Lambert. *Intra-frame sharing for low-complexity decoding of SHVC video*. In Proc. IEEE Int. Conf. Consum. Electron. (ICCE), pages 297–298, Jan. 2016.

9. G. Cebrian-Marquez, A. J. Diaz-Honrubia, J. De Praeter, G. Van Wallendael, J. L. Martinez, and P. Cuenca. *A motion vector re-use algorithm for H.264/AVC and HEVC simultaneous video encoding*. In Proc. ACM International Conference on Advances in Mobile Computing and Multimedia, pages 241–245, Dec. 2015.

10. J. De Praeter, P. Duchi, G. Van Wallendael, J. F. Macq, and P. Lambert. *Efficient Encoding of Interactive Personalized Views Extracted from Immersive Video Content*. In Proc. ACM 1st International Workshop on Multimedia Alternate Realities, pages 25–30, Oct. 2016.

11. C. Van Goethem, J. De Praeter, T. Paridaens, G. Van Wallendael, and P. Lambert. *Multistream Video Encoder for Generating Multiple Dynamic Range Bitstreams*. In Picture Coding Symposium (PCS), pages 1–5, Dec. 2016.

12. J. De Praeter, J. Van de Vyver, N. Van Kets, G. Van Wallendael, and S. Verstockt. *Moving Object Detection in the HEVC Compressed Domain for Ultra-High-Resolution Interactive Video*. In Proc. IEEE Int. Conf. Consum. Electron. (ICCE), Jan. 2017.

2

Low-Complexity Encoding of High Efficiency Video Coding

2.1 Introduction

High Efficiency Video Coding (HEVC) is the most recently standardized video compression standard, designed by the Joint Collaborative Team on Video Coding (JCT-VC) of the ISO/IEC Moving Picture Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG) [1]. This standard provides twice the compression efficiency of its predecessor, H.264/AVC, for the same subjective quality thanks to a series of new or improved coding tools. However, these tools also result in a much higher coding complexity compared to H.264/AVC. Therefore, even without considering a system with an individual encoder per client, many efforts have been made to reduce the computational complexity of HEVC encoders.

In order to better contextualize the complexity reductions achieved in the applications in the following chapters of this dissertation, this chapter will provide an overview of related work that reduces the coding complexity of a single encoder, without exploiting coding information from other encodings. First, a short overview is presented of the coding decisions made by an HEVC encoder in Section 2.2. Next, Section 2.3 gives an overview of the accelerations that can be achieved by accelerating certain encoding decisions and compares the related work to the maximum attainable complexity reduction. Finally, the conclusion of this chapter is presented in Section 2.4.

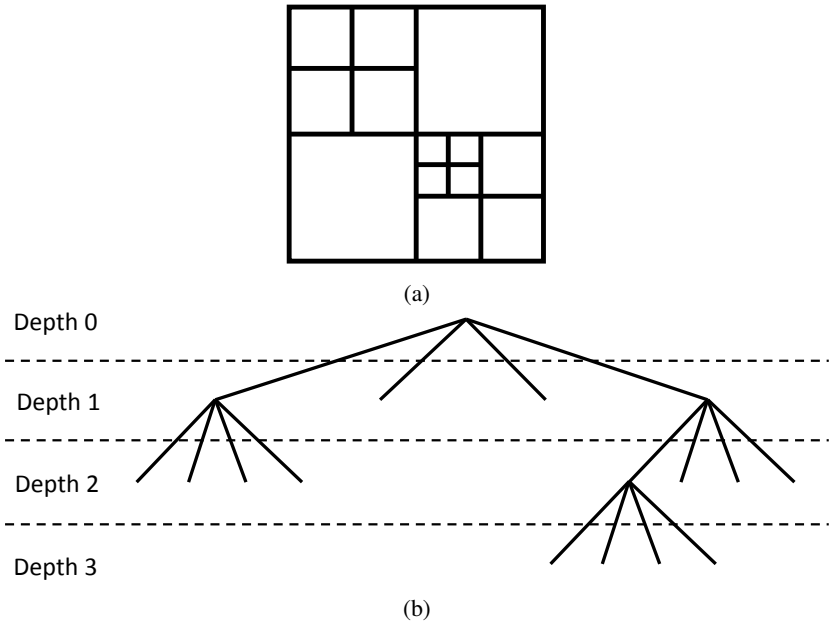


Figure 2.1: Block representation (a) and quadtree representation (b) of a CTU. For a CTU of 64×64 pixels, depths of 0, 1, 2, and 3 respectively correspond with CUs of 64×64, 32×32, 16×16, and 8×8 pixels.

2.2 High Efficiency Video Coding

As mentioned in the previous section, the improved compression efficiency of HEVC comes at a cost of a large computational complexity. In order to better understand the source of this increased complexity, and to also identify the coding decisions that can be accelerated in an HEVC encoder, a high-level overview of the most computationally-complex coding tools is presented here.

The first of these coding tools is the coding unit (CU) splitting process. In HEVC, a frame is divided in coding tree units (CTUs) which are typically 64×64 pixels and should be considered as such in the rest of this dissertation. These CTUs are recursively split into CUs according to a quadtree structure down to a size of 8×8 pixels. Whether or not a block (and thus the corresponding node in the quadtree) should be split, is signaled by a split flag.

Note that in this dissertation, the depth of a CU refers to its depth in the quadtree, hence depth 0, 1, 2, and 3 respectively correspond with CUs of 64×64, 32×32, 16×16, and 8×8 pixels. This relation between the quadtree depth and the block representation of a CTU is illustrated in Figure 2.1. Furthermore, due to the flexibility of recursively splitting a CTU into these different CU sizes, each CTU

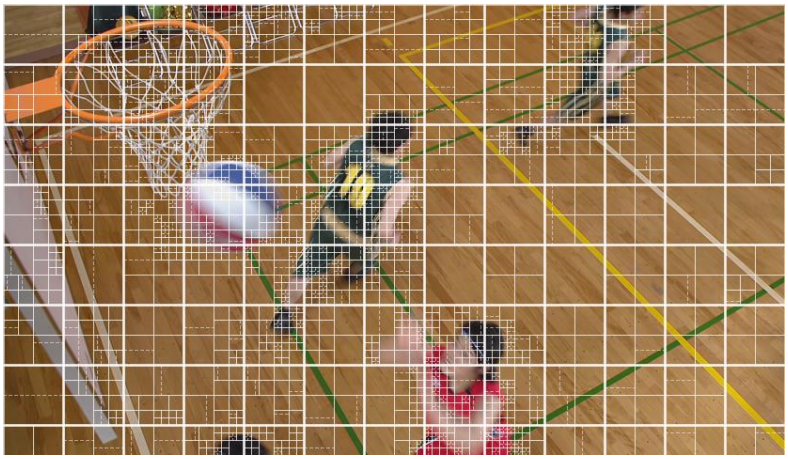


Figure 2.2: Example of CU splitting (solid white lines) and PU splitting (dashed lines) of a frame. Regions with more details and/or movement are assigned smaller block partitions.

has 83,522 possible CU structures which have to be evaluated by the encoder in terms of compression efficiency.

For each CU, the optimal prediction mode has to be decided. This mode is either intra- or inter-prediction. Depending on the mode, a CU can be further partitioned into Prediction Units (PUs). For intra-mode, a CU can be assigned one of two PU partitioning modes, whereas an inter-predicted CU has to decide the optimal PU partitioning out of eight possible modes. An example of CU and PU splitting is shown in Figure 2.2, where it can be seen that regions with more details and/or movement are assigned smaller block partitions.

If a PU is intra-predicted, 35 possible intra-prediction directions have to be evaluated for luma-components, whereas five possible modes (derived from the luma direction) are evaluated for the chroma-components. If a PU is inter-predicted, motion estimation is carried out to find the best-matching block in one of the reference pictures. In case of B-frames, both unidirectional and bidirectional prediction are tested.

Besides motion estimation, inter-predicted PUs are also evaluated with merge- and skip-mode. Merge-mode indicates that the PU has the same motion vector as the best motion vector predictor candidate. These candidates can be the motion vectors of spatially or temporally neighboring blocks and are used to calculate the motion vector difference during motion estimation. In case of merge-mode, this difference equals zero, and only the index of the best motion vector predictor candidate is encoded into the bitstream. Skip-mode then differs from merge-mode in a sense that no residual picture is encoded and that it can only be used if a CU only has one PU (a partitioning mode of $2N \times 2N$).

Finally, each CU is also recursively split according to a quadtree structure into Transform Units (TUs) for transformation and quantization of the residual picture. The smallest size of these TUs is 4×4 pixels.

To determine the most compression-efficient combination of possible coding parameters, encoders can evaluate all possible configurations during the rate-distortion optimization (RDO) process. However, since this RDO process is computationally intensive, several accelerations have been proposed to reduce the amount of configurations to be evaluated. These possible accelerations are discussed and compared in the next section.

2.3 Accelerating HEVC

In this section, related work on accelerating HEVC is compared to each other. In order to facilitate such comparisons, these works all attempt to use similar test conditions and evaluation metrics. These conditions and metrics are presented first, followed by the actual comparison.

2.3.1 Test conditions and metrics

All encoder-acceleration algorithms proposed in the related literature are evaluated using the HEVC reference software (HM) [2]. Although this software is meant as a reference, depending on the version, there might be small discrepancies if the algorithm were to be implemented in a different version. However, for the purpose of a high-level comparison, as is aimed for in this dissertation, these differences should be considered negligible.

In addition to using the same reference software, implemented algorithms are also evaluated on a standardized test set of video sequences [3]. These videos are divided into six classes depending on their spatial resolution or other characteristics. Class A contains videos with a resolution of 2560×1600 pixels with a bit depth of 10 bits for two of the four videos in this class and a bit depth of 8 bits for the remaining two. All other classes only contain videos with a bit depth of 8 bits. Class B, C, and D then contain videos with a resolution of respectively 1920×1080, 832×480, and 416×240 pixels. Class E consists of video content at a resolution of 1280×720 pixels, with the special characteristic of this class being that the videos always contain a close-up of one or two people speaking against a static background. Finally, class F consists of special content such as a computer-generated video and screen captures of slide shows. Note that due to the special natures of class E and F, and the high resolution of class A, these classes are sometimes not included in the evaluation of algorithms in the related literature.

When encoding the video, four different configurations can be used depending on the intended scenario that is evaluated:

- *All-intra* is a configuration that encodes the video with only intra-frames. This configuration is used to test accelerations that only target intra-coded CUs.

- The *low-delay P* configuration encodes the video with an intra-frame, followed by P-frames until the end of the video sequence. The frames are encoded in display order with each frame only using previous frames as reference frames. Since there is no structural delay of having to wait for future frames when decoding, this configuration is suitable for low-delay scenarios.

- The *low-delay* configuration differs from the low-delay P configuration by encoding all P-frames as B-frames instead. As with the P-frames, these B-frames only use previous frames as reference frames. However, since each PU in a B-frame can have up to two motion vectors instead of one, both the compression efficiency and the computational complexity of encoding these frames is higher than those of P-frames. Quantitatively, the low-delay configuration reduces the bit rate overhead for the same quality as the low-delay P configuration by 7.5% on average. However, this comes at a cost of an average computational complexity increase of 48.9%.

- *Random access* contains both B-frames and I-frames, with an I-frame at the start of each intra-period to provide random access to the video stream. This intra-period is defined as approximately one second. Additionally, hierarchical B-frames are used, meaning that, contrary to the low-delay configurations, the frames are not encoded in display order.

Using (subsets of) the above test conditions, the algorithms in the related literature are then evaluated in terms of compression efficiency and reduction of computational complexity compared to a non-accelerated encoding of the same video under the same test conditions. The compression efficiency is evaluated as Bjøntegaard-Delta rate (BD-rate) [4], which is the average bit rate overhead of the fast encoding algorithm compared to the non-accelerated encoding for the same quality. In order to calculate this metric, four rate-distortion (RD) points are required for both the fast encoding and the non-accelerated encode. These points are acquired by using four different quantization parameter (QP) values (22, 27, 32 and 37) and measuring the resulting bit rate and distortion, with the distortion expressed as the Peak Signal-to-Noise Ratio (PSNR). Finally, the BD-rate is measured as the average relative bit rate difference between the curves resulting from the interpolation of the two sets of four RD-points.

Computational complexity reduction is usually expressed as time saving (TS):

$$TS = \frac{T_{ref} - T_{fast}}{T_{ref}}\,, \tag{2.1}$$

with T_{ref} being the encoding time of the non-accelerated encoding with the HM encoder, while T_{fast} is the encoding time of the accelerated encoding under the same test conditions. Additionally, speed-up is also used as a measure:

$$Speed\text{-}up = \frac{T_{ref}}{T_{fast}} = \frac{1}{1 - TS} \tag{2.2}$$

This measure equals the amount of fast encoders that are needed to reach the same computational complexity as a single non-accelerated encoder. In this chapter, all complexity reductions reported using speed-up in the related literature have been converted to TS using the above equation.

2.3.2 Comparison

Tables 2.1 to 2.4 show the state-of-the-art fast encoding algorithms in the literature for respectively the all-intra, low-delay P, low-delay, and random access configurations. These methods are sorted according to increasing times saving. Note, however, that these numbers should only be considered as a rough indication of the performance of the algorithms, since they have been implemented on different versions of HM, with some also omitting evaluation on the test sequences in class A, E and/or F.

Additionally, Figures 2.3 to 2.6 compare the time savings of these methods to the maximum computational complexity reduction that can be achieved when skipping certain coding decisions. These maxima were determined by first performing a non-accelerated encoding for all test sequences and configurations, and then using the coding decisions generated by these encodings to skip these decision in an accelerated encoding of the same videos.

By examining Table 2.1, it is seen that all but two of the fast-encoding methods accelerate CU splitting decisions. The motivation for accelerating these decisions is that all other coding decisions presented in Section 2.2 are evaluated for each CU. As such, even if decisions concerning the intra direction are accelerated for each possible CU structure, this optimal CU structure still needs to be determined. On the other hand, if only a limited amount of CU structures are evaluated, the optimal intra direction decision only needs to be found for each of these CU structures. Besides CU splitting decisions, only PU, TU and intra direction decisions are accelerated for the all-intra configuration, since motion estimation and merge- and skip-mode are only used for inter-coded PUs.

When comparing the time saving of methods presented in Table 2.1 to the maximum time savings shown in Figure 2.3, it becomes clear that the highest time saving achieved by the algorithms in the literature is similar to the maximum time saving that can be achieved by accelerating all CU coding decisions (dashed line (1) in the figure). This is achieved with a complexity reduction of 63.5% by the method of S. Cho that uses an early-splitting and early-stopping condition in order

to evaluate only one CU structure for each CTU. The overhead in terms of bit rate for this acceleration is 3.5%. The method of M. Yang achieves a greater complexity reduction of 65.4% with a BD-rate of only 1.3%. However, although both CU, TU, and intra directions decisions are accelerated, the resulting complexity reduction is still significantly lower than the time saving of 77.6% when accelerating all CU and PU decisions (dashed line (2) in Figure 2.3) or the highest possible time saving of 94.5% when accelerating all coding decisions (dashed line (4) in the same figure).

Similar observations as with the all-intra configuration can be made for low-delay-P (Table 2.4), low-delay (Table 2.5) and random access (Table 2.4). Again, CU coding decisions are most commonly accelerated, since all other coding decisions are made on a sub-CU level. Besides CU decisions, PU decisions are also often accelerated, since decreasing the number of PU modes to be evaluated (eight in the case of inter-coded CUs) has a greater effect than in the case of intra-coded CUs (only one or two possible PU sizes). Skip-mode is frequently accelerated as well, since a successful early detection of skip-mode implies that no further decisions need to be made for that CU concerning PUs, TUs and motion estimation.

Despite many techniques for accelerating the encoding of inter-coded frames, with the highest reported time savings for the low-delay-P, low-delay, and random access configurations respectively being 67.6%, 68%, and 69%, neither of these methods reaches the maximum time saving that can be achieved when accelerating all CU decisions. This is more clearly illustrated in Figures 2.4 to 2.6, where none of the techniques achieve a higher time saving than indicated by the dashed line (1). The reason for this is that, for example in the case of accelerating CU decisions, fast-encoding will almost never accelerate all of these decisions, since making non-optimal decisions comes at the cost of increasing BD-rates. Since they do not accelerate all CU decisions, they can never break past line (1) by only accelerating CU decisions. Therefore, in order to reach higher complexity reductions, accelerating other coding decisions is necessary. However, as with CU decisions, they never skip all coding decisions, meaning that it is impossible for conventional fast-encoding algorithms to reach line (4) in the figure.

Figure 2.6 also further illustrates how greater time savings come at a cost of higher BD-rates, since accelerating coding decisions typically means that less optimal coding decisions will be chosen than with a non-accelerated encoding. Because fast-encoding algorithms are meant to reduce the computational complexity of an encoder while preserving the compression efficiency as much as possible, time savings above the maximum possible time saving of accelerating all CU decisions have not been investigated further in the context of fast encoding. However, as seen in Table 2.5 where the time saving values for accelerating all CU decisions and accelerating all coding decisions have been converted to speed-up values, state-of-the-art fast-encoding algorithms speed up the HM reference soft-

Method	Accelerated decisions	TS (%)	BD-rate (%)
Y. Lin [5]	CU	13.4	0.0
M. Fini [6]	intra direction	14.4	0.1
Y. Cen [7]	CU	15.9	2.8
S. Wang [8]	CU + intra direction	26.6	0.5
D. Ruiz [9]	intra direction	29.7	0.4
F. Belghith [10]	CU	31.3	0.8
H. Ding [11]	CU	35.8	2.3
I. Marzuki [12]	CU + PU + TU + intra direction	40.0	1.3
M. Ramezanpour [13]	CU + PU + intra direction	43.0	0.5
N. Hu [14]	CU + PU + intra direction + other	43.9	0.3
Y. Lin [15]	CU	44.1	1.5
A. Oztekin [16]	CU	45.1	4.6
M. Ramezanpour [17]	CU + PU + intra direction	45.8	1.5
L. Shen [18]	CU	47.0	1.1
L. Zhao [19]	CU + TU + intra direction	50.0	0.5
B. Min [20]	CU	52.3	0.8
D. Ruiz [21]	CU	53.2	2.2
K. Lim [22]	CU + PU	53.5	2.0
W. Zhao [23]	CU + PU + intra direction	54.0	2.5
H. Kim [24]	CU	54.2	1.0
Y. Zhang [25]	CU	55.6	1.3
S. Park [26]	CU + PU	57.0	0.6
Y. Song [27]	CU + PU + intra direction	57.2	0.9
H. Zhang [28]	CU + intra direction	60.0	1.0
S. Cho [29]	CU	63.5	3.5
M. Yang [30]	CU + TU + Intra direction	65.4	1.3

Table 2.1: Comparison of state-of-the-art acceleration for the all-intra configuration.

Method	Accelerated decisions	TS (%)	BD-rate (%)
H. Zhang [28]	CU + intra direction	15.0	0.9
S. Yang [31]	motion	19.0	0.2
A. Jimenez-Moreno [32]	CU	26.4	0.9
S. Wang [8]	CU + motion + intra direction	40.9	0.4
J. Xiong [33]	CU	42.8	1.9
H. Kim [24]	CU	48.5	0.6
Q. Zhang [34]	CU	49.6	1.8
Z. Liu [35]	CU + PU	56.7	1.1
Q. Hu [36]	CU + PU + skip	58.1	1.1
W. Zhao [23]	CU + PU + intra direction	67.6	2.4

Table 2.2: Comparison of state-of-the-art acceleration for the low-delay-P configuration.

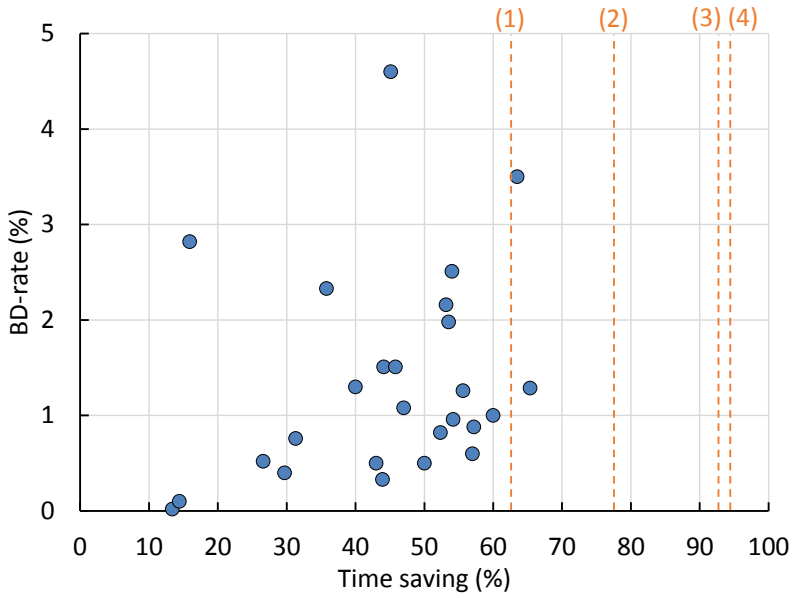


Figure 2.3: Comparison of state-of-the-art accelerations for the all-intra configuration with the maximum complexity reduction when accelerating all coding decisions related to CU (1) + PU (2) + TU (3) + intra direction (4).

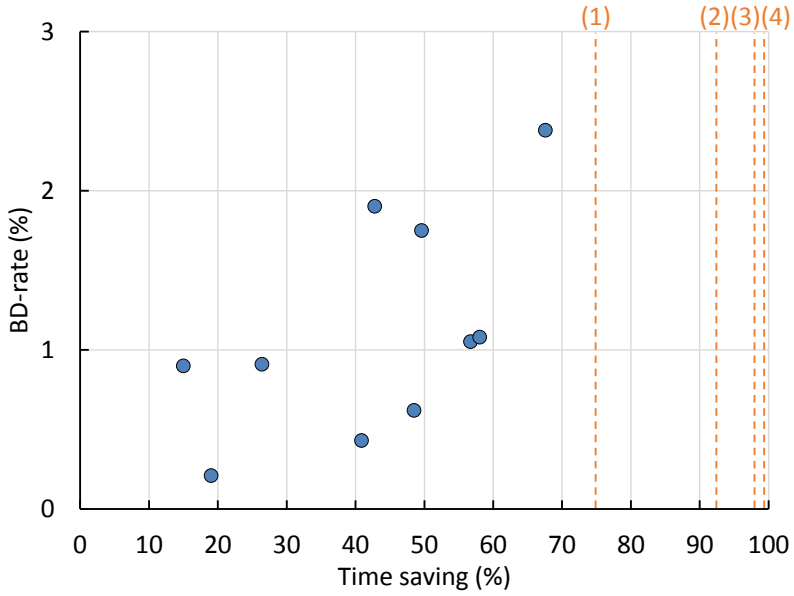


Figure 2.4: Comparison of state-of-the-art accelerations for the low-delay P configuration with the maximum complexity reduction when accelerating all coding decisions related to CU (1) + PU (2) + merge + skip (3) + motion + TU + intra direction (4).

Method	Accelerated decisions	TS (%)	BD-rate (%)
L. Shen [37]	TU	13.0	0.5
H. Zhang [28]	CU + intra direction	15.0	0.8
B. Kim [38]	merge + skip	30.0	2.2
X. Huang [39]	PU	35.2	1.9
K. Goswami [40]	PU + skip	39.6	0.5
J. Xiong [33]	CU	40.3	2.2
L. Shen [41]	CU	41.0	1.1
S. Ahn [42]	CU + skip	42.7	1.0
Q. Zhang [34]	CU	46.9	2.0
H. Kim [24]	CU	48.4	0.6
T. Lin [43]	CU	49.9	0.7
L. Shen [44]	CU + skip	51.8	0.9
H. Tan [45]	CU	56.0	0.8
Z. Liu [46]	CU	57.0	0.8
Y. Ahn [47]	CU + PU	57.8	2.0
J. Lee [48]	CU + merge + skip	68.0	2.5

Table 2.3: Comparison of state-of-the-art acceleration for the low-delay configuration.

Method	Accelerated decisions	TS (%)	BD-rate (%)
L. Shen [37]	TU	13.0	0.6
H. Zhang [28]	CU + intra direction	17.0	1.1
H. Lee [49]	Skip	33.3	0.0
B. Kim [38]	merge + skip	35.4	0.8
G. Correa [50]	CU	37.0	0.3
X. Huang [39]	PU	39.5	2.0
K. Goswami [40]	PU + skip	40.4	0.4
L. Shen [41]	CU	42.0	1.5
Z. Liu [46]	CU	49.0	0.8
L. Shen [44]	CU + skip	49.1	0.7
S. Ahn [42]	CU + skip	49.6	1.4
J. Lee [51]	CU + PU	51.7	2.6
H. Kim [24]	CU	53.6	0.7
A. Lee [52]	CU + PU	54.9	0.9
T. Lin [43]	CU	55.0	0.5
H. Tan [45]	CU	55.0	1.2
Y. Ahn [47]	CU + PU	57.5	1.4
Z. Liu [35]	CU + PU	59.8	1.0
Q. Hu [36]	CU + PU + skip	65.1	1.3
G. Correa [53]	CU + PU + TU	65.3	1.4
W. Zhao [23]	CU + PU + intra direction	68.4	1.9
J. Lee [48]	CU + merge + skip	69.0	3.0

Table 2.4: Comparison of state-of-the-art acceleration for the random-access configuration.

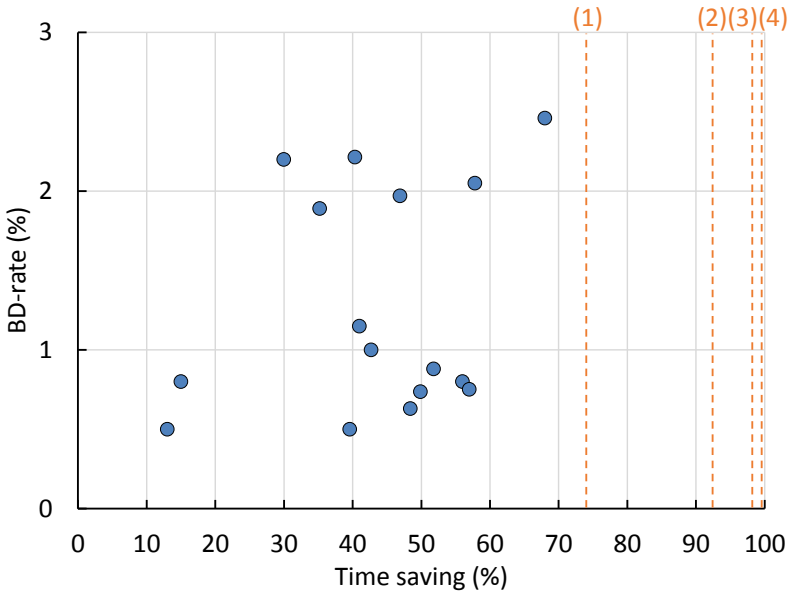


Figure 2.5: Comparison of state-of-the-art accelerations for the low-delay configuration with the maximum complexity reduction when accelerating all coding decisions related to CU (1) + PU (2) + merge + skip (3) + motion + TU + intra direction (4).

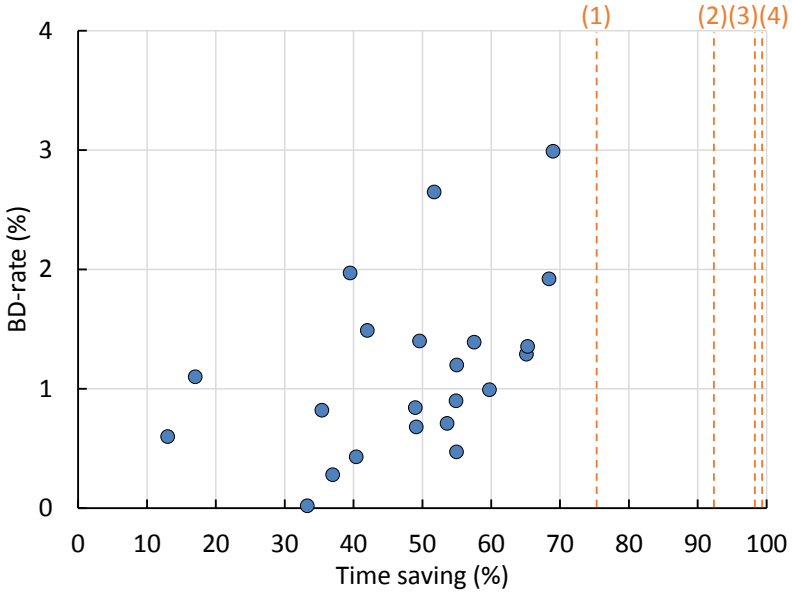


Figure 2.6: Comparison of state-of-the-art accelerations for the random access configuration with the maximum complexity reduction when accelerating coding decisions related to CU (1) + PU (2) + merge + skip (3) + motion + TU + intra direction (4).

	Time saving (%)		Speed-up	
Configuration	CU	All	CU	All
All-intra	62.7	94.5	2.7	18.3
Low-delay-P	74.9	99.4	4.0	159.4
Low-delay	74.1	99.6	3.9	227.8
Random access	75.3	99.4	4.1	167.6

Table 2.5: Maximum time savings and speed-ups when accelerating only CU decisions versus accelerating all coding decisions, for all configurations.

ware by at most a factor of about 2.7 for the all-intra configuration, and about 4 for all other configurations. Although the maximum possible speed-up for the all-intra configuration is 18.3, the speed-ups of the other configurations reach values of more than 150, meaning that accelerating all coding decisions can allow a content provider to run more than 150 fast encoders for the same computational complexity as a non-accelerated encoder. Although achieving this number might not seem realistic for traditional fast encoders, since they will always have to execute some calculations in order to determine the coding information, these number are achievable by guided, personalized encoders in certain scenarios, as will be shown in the further chapters of this dissertation.

2.4 Conclusion

This chapter presented an overview of the coding decisions taken by an HEVC encoder and illustrated how the state-of-the-art fast encoding methods only achieve a limited complexity reduction of the encoder, since they aim at reducing the computational complexity of a single fast encoder while retaining the compression efficiency. However, in scenarios with personalized encoders, this complexity reduction is insufficient, since each extra user of the system will linearly increase the overall computational complexity. Therefore, in this dissertation, the focus lies on achieving very high speed-ups for a variety of scenarios with personalized encoders, while utilizing the similarities between the coding information of these encoders.

References

[1] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) Standard.* IEEE Trans. Circuits Syst. Video Technol., 22(12):1649–1668, Dec. 2012.

[2] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description Update 2*. Technical Report JCTVC-T1002, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Feb. 2015.

[3] F. Bossen. *Common test conditions and software reference configurations*. Technical Report JCTVC-L1100, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2013.

[4] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

[5] Y.-C. Lin and J.-C. Lai. *Feature-based fast coding unit partition algorithm for high efficiency video coding*. J. Appl. Research Technol., 13(2):205–219, 2015.

[6] M. R. Fini and F. Zargari. *Two stage fast mode decision algorithm for intra prediction in HEVC*. Multimedia Tools Appl., pages 1–18, 2015.

[7] Y.-F. Cen, W.-L. Wang, and X.-W. Yao. *A fast CU depth decision mechanism for HEVC*. Inf. Process. Lett., 115(9):719–724, 2015.

[8] S. Wang, F. Luo, S. Ma, X. Zhang, S. Wang, D. Zhao, and W. Gao. *Low complexity encoder optimization for HEVC*. J. Visual Commun. Image Representation, 35:120–131, 2016.

[9] D. Ruiz, G. Fernández-Escribano, J. L. Martínez, and P. Cuenca. *Fast intra mode decision algorithm based on texture orientation detection in HEVC*. Signal Process. Image Commun., 44:12–28, 2016.

[10] F. Belghith, H. Kibeya, M. A. B. Ayed, and N. Masmoudi. *Fast coding unit partitioning method based on edge detection for HEVC intra-coding*. Signal Image Video Process., pages 1–8, 2015.

[11] H. Ding, X. Huang, and Q. Zhang. *The fast intra CU size decision algorithm using gray value range in HEVC*. Optik - Int. J. Light Electronc Optics, 127(18):7155–7161, 2016.

[12] I. Marzuki, J. Ma, Y.-J. Ahn, and D. Sim. *A context-adaptive fast intra coding algorithm of high-efficiency video coding (HEVC)*. J. Real-Time Image Process., pages 1–17, 2015.

[13] M. Ramezanpour and F. Zargari. *Fast CU size and prediction mode decision method for HEVC encoder based on spatial features*. Signal Image Video Process., pages 1–8, 2016.

[14] N. Hu and E. H. Yang. *Fast Mode Selection for HEVC Intra-Frame Coding With Entropy Coding Refinement Based on a Transparent Composite Model*. IEEE Trans. Circuits Syst. Video Technol., 25(9):1521–1532, Sept. 2015.

[15] Y.-C. Lin, J.-C. Lai, and H.-C. Cheng. *Coding unit partition prediction technique for fast video encoding in HEVC*. Multimedia Tools Appl., pages 1–24, 2015.

[16] A. Öztekin and E. Erçelebi. *An early split and skip algorithm for fast intra CU selection in HEVC*. J. Real-Time Image Process., pages 1–11, 2015.

[17] M. Ramezanpour and F. Zargari. *Fast HEVC I-frame coding based on strength of dominant direction of CUs*. J. Real-Time Image Process., pages 1–10, 2015.

[18] L. Shen, Z. Zhang, and Z. Liu. *Effective CU Size Decision for HEVC Intracoding*. IEEE Trans. Image Process., 23(10):4232–4241, Oct. 2014.

[19] L. Zhao, X. Fan, S. Ma, and D. Zhao. *Fast intra-encoding algorithm for high efficiency video coding*. Signal Process. Image Commun., 29(9):935–944, 2014.

[20] B. Min and R. C. C. Cheung. *A Fast CU Size Decision Algorithm for the HEVC Intra Encoder*. IEEE Trans. Circuits Syst. Video Technol., 25(5):892–896, May 2015.

[21] D. Ruiz, G. Fernández-Escribano, V. Adzic, H. Kalva, J. L. Martínez, and P. Cuenca. *Fast CU partitioning algorithm for HEVC intra coding using data mining*. Multimedia Tools Appl., pages 1–34, 2015.

[22] K. Lim, J. Lee, S. Kim, and S. Lee. *Fast PU Skip and Split Termination Algorithm for HEVC Intra Prediction*. IEEE Trans. Circuits Syst. Video Technol., 25(8):1335–1346, Aug. 2015.

[23] W. Zhao, T. Onoye, and T. Song. *Hierarchical Structure-Based Fast Mode Decision for H.265/HEVC*. IEEE Trans. Circuits Syst. Video Technol., 25(10):1651–1664, Oct. 2015.

[24] H. S. Kim and R. H. Park. *Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule*. IEEE Trans. Circuits Syst. Video Technol., 26(1):130–138, Jan. 2016.

[25] Y. Zhang, S. Kwong, G. Zhang, Z. Pan, H. Yuan, and G. Jiang. *Low Complexity HEVC INTRA Coding for High-Quality Mobile Video Communication*. IEEE Trans. Ind. Informat., 11(6):1492–1504, Dec. 2015.

[26] S. J. Park. *CU encoding depth prediction, early CU splitting termination and fast mode decision for fast HEVC intra-coding*. Signal Process. Image Commun., 42:79–89, 2016.

[27] Y. Song, Y. Zeng, X. Li, B. Cai, and G. Yang. *Fast CU size decision and mode decision algorithm for intra prediction in HEVC*. Multimedia Tools Appl., pages 1–17, 2016.

[28] H. Zhang and Z. Ma. *Fast Intra Mode Decision for High Efficiency Video Coding (HEVC)*. IEEE Trans. Circuits Syst. Video Technol., 24(4):660–668, Apr. 2014.

[29] S. Cho and M. Kim. *Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding*. IEEE Trans. Circuits Syst. Video Technol., 23(9):1555–1564, Sept. 2013.

[30] M. Yang and C. Grecos. *Fast intra encoding decisions for high efficiency video coding standard*. J. Real-Time Image Process., pages 1–10, 2014.

[31] S. H. Yang and K. S. Huang. *HEVC fast reference picture selection*. Electron. Lett., 51(25):2109–2111, Dec. 2015.

[32] A. Jimenez-Moreno, E. Martinez-Enriquez, and F. Diaz-de-Maria. *Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard*. IEEE Trans. Multimedia, 18(4):563–575, April. 2016.

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

[34] Q. Zhang, J. Zhao, X. Huang, and Y. Gan. *A fast and efficient coding unit size decision algorithm based on temporal and spatial correlation*. Optik - Int. J. Light Electronc Optics, 126(21):2793–2798, 2015.

[35] Z. Liu, T.-L. Lin, and C.-C. Chou. *Efficient prediction of CU depth and PU mode for fast HEVC encoding using statistical analysis*. J. Visual Commun. Image Representation, 38:474–486, 2016.

[36] Q. Hu, X. Zhang, Z. Shi, and Z. Gao. *Neyman-Pearson-Based Early Mode Decision for HEVC Encoding*. IEEE Trans. Multimedia, 18(3):379–391, Mar. 2016.

[37] L. Shen, Z. Zhang, X. Zhang, P. An, and Z. Liu. *Fast TU size decision algorithm for HEVC encoders using Bayesian theorem detection*. Signal Process. Image Commun., 32:121–128, 2015.

[38] B.-G. Kim. *Fast coding unit (CU) determination algorithm for high-efficiency video coding (HEVC) in smart surveillance application*. J. Supercomputing, pages 1–22, 2016.

[39] X. Huang, Q. Zhang, X. Zhao, W. Zhang, Y. Zhang, and Y. Gan. *Fast inter-prediction mode decision algorithm for HEVC*. Signal Image Video Process., pages 1–8, 2015.

[40] K. Goswami, J.-H. Lee, and B.-G. Kim. *Fast algorithm for the High Efficiency Video Coding (HEVC) encoder using texture analysis*. Inf. Sci., 364:72–90, 2016.

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

[42] S. Ahn, B. Lee, and M. Kim. *A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding*. IEEE Trans. Circuits Syst. Video Technol., 25(3):422–435, Mar. 2015.

[43] T.-L. Lin, C.-C. Chou, Z. Liu, and K.-H. Tung. *HEVC early termination methods for optimal CU decision utilizing encoding residual information*. J. Real-Time Image Process., pages 1–17, 2016.

[44] L. Shen, Z. Zhang, and Z. Liu. *Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations*. IEEE Trans. Circuits Syst. Video Technol., 24(10):1709–1722, Oct. 2014.

[45] H. L. Tan, C. C. Ko, and S. Rahardja. *Fast Coding Quad-Tree Decisions Using Prediction Residuals Statistics for High Efficiency Video Coding (HEVC)*. IEEE Trans. Broadcast., 62(1):128–133, Mar. 2016.

[46] Z. Liu, T.-L. Lin, and C.-C. Chou. *HEVC coding-unit decision algorithm using tree-block classification and statistical data analysis*. Multimedia Tools Appl., pages 1–22, 2016.

[47] Y.-J. Ahn and D. Sim. *Square-type-first inter-CU tree search algorithm for acceleration of HEVC encoder*. J. Real-Time Image Process., pages 1–14, 2015.

[48] J. Lee, S. Kim, K. Lim, and S. Lee. *A Fast CU Size Decision Algorithm for HEVC*. IEEE Trans. Circuits Syst. Video Technol., 25(3):411–421, Mar. 2015.

[49] H. Lee, H. J. Shim, Y. Park, and B. Jeon. *Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality*. IEEE Trans. Broadcast., 61(3):388–397, Sept. 2015.

[50] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz. *Fast coding tree structure decision for HEVC based on classification trees*. Analog Integr. Circuits and Signal Process., 87(2):129–139, 2016.

[51] J.-H. Lee, K. Goswami, B.-G. Kim, S. Jeong, and J. S. Choi. *Fast encoding algorithm for high-efficiency video coding (HEVC) system based on spatio-temporal correlation*. J. Real-Time Image Process., pages 1–12, 2015.

[52] A. Lee, D. Jun, and J. S. Choi. *Fast motion estimation using priority-based inter-prediction mode decision method in high efficiency video coding*. J. Real-Time Image Process., pages 1–9, 2015.

[53] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz. *Fast HEVC Encoding Decisions Using Data Mining*. IEEE Trans. Circuits Syst. Video Technol., 25(4):660–673, Apr. 2015.

3

Encoding Complexity Reduction of Personalized Video Compositions

3.1 Introduction

A first personalization scenario in which guided encoding can be used to accelerate personalized encoders occurs when creating video compositions. In many industries, these compositions often consist of visual information such as surveillance footage. This information is gathered in a control room where it is displayed as a composition of multiple input streams on a video wall. These individual video sequences are either tiled next to each other, or partially overlap. However, (part of) this information is sent to other devices such as desktop computers, laptops, and other handheld devices. All of these devices receive a personalized composition of input bitstreams. This composition must be freely arrangeable, meaning that each separate sequence can be dragged across the screen. Since decoding all these input bitstreams requires multiple decoders, which are not available on the client device, the composition is created in the network by specialized hardware, as seen in Figure 3.1. This new bitstream is then decoded by a single decoder at the receiver.

The same approach is used for sending personalized advertisements embedded in the picture during live video broadcasts. Depending on the location of the viewers, broadcasters insert different advertisements into the encoded video stream of the content providers without requiring the client device to decode both the video and advertisement.

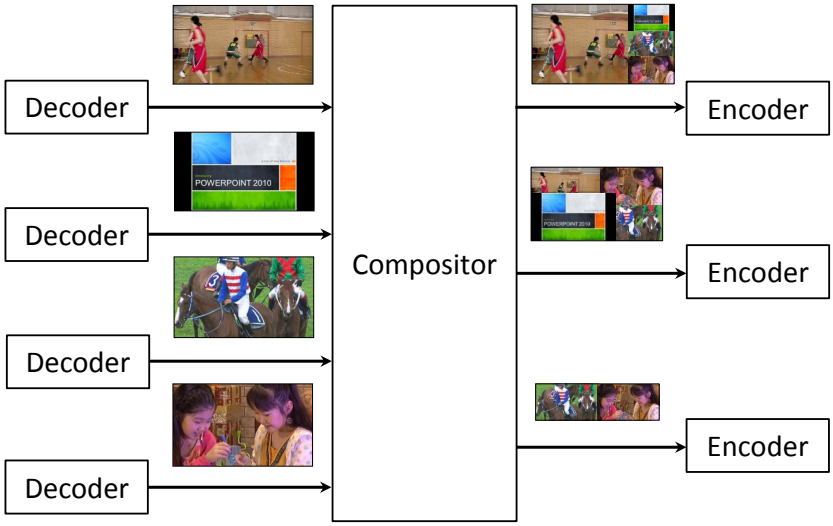
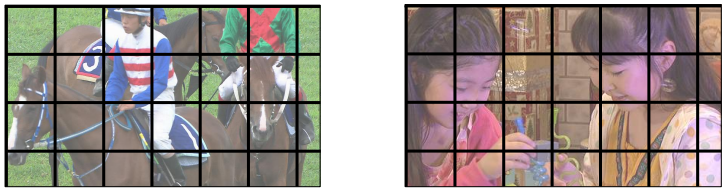


Figure 3.1: Depending on the needs of the client, a personalized composition (right) is created from several input videos (left).

Another similar scenario occurs during video conferencing between more than two locations. If a participant wants to display all video streams on his handheld device, the power and memory of the decoder might be insufficient for decoding the multiple input bitstreams.

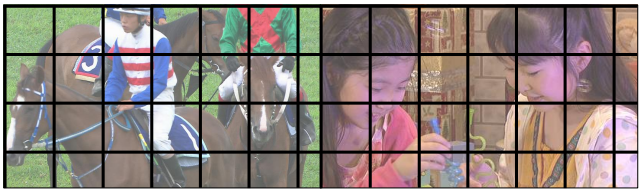
In the above use cases, each different, personalized composition needs to be encoded. This encoding step requires much computational power. Moreover, current displays support resolutions up to 3840×2160 pixels and will further evolve towards 7680×4320 pixels. With such high resolutions becoming common, both the input video sequences and the composition will be compressed using HEVC [1], since this compression standard offers a bit rate reduction of 50% for the same perceptual quality as its predecessor, H.264/AVC [2]. However, as was mentioned in Chapter 2, the computational complexity of the encoder is also higher compared to the predecessors of HEVC. Consequently, reducing this increased complexity is crucial in order to limit the resources necessary to generate compositions.

Previous work on compositions focuses on picture-in-picture insertion of video content in H.264/AVC by re-using encoding decisions from the original video sequences [3–5]. A similar approach was also used for HEVC in order to achieve high complexity reductions. In this approach, the block structures of the input bitstreams are merged and passed to the encoder, reducing the number of encoding decisions that need to be evaluated [6]. However, this solution is only applicable if the inserted content is constrained to a fixed grid based on the size of the CTUs. In compositions that do not constrain sequences to this grid, misalignment of the CTU-grids is introduced as seen in Figure 3.2.

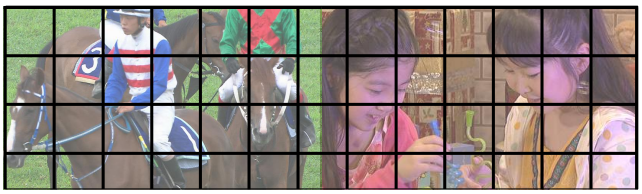


(a) CTU-grid of video 1

(b) CTU-grid of video 2



(c) Misalignment of grids



(d) Correct CTU-grid of composition

Figure 3.2: Illustration of CTU-grid misalignment in a composition. If the CTU-grids of (a) and (b) are naively combined in a composition (c), misalignment of the grids occurs in the middle. The correct CTU-grid of the entire composition should instead be (d).

In this figure, the CTU-grids of both video 1 and video 2 are depicted. When arranging both videos into a composition, the grid of video 2 (Figure 3.2c) becomes misaligned with the grid of the overall composition (Figure 3.2d). In fact, the correct CTU-grid of video 2 in the composition is shifted by 32 pixels to the right compared to the original CTU-grid. As such, in the remainder of this dissertation, a shift is defined as the number of pixels in the x- and y-direction that a CTU-grid of a new version of the video is shifted compared to the CTU-grid in the original version of the video.

Since coding information in HEVC is determined for each CTU, simply copying this coding information in order to achieve fast encoding of a composition is not possible if the CTU-grid is shifted for one or more videos. As such, in order to support flexible, personalized compositions, a more intelligent solution is necessary for fast re-encoding of misaligned content in HEVC.

The problem of handling misaligned video sequences is classified as transcoding, which is defined as converting one compressed input bitstream to an outgoing bitstream. A naive cascaded transcoding approach consists of decoding the input

bitstream, applying the desired operation, and then re-encoding the video. However, this re-encoding step is computationally complex, whereas all of the coding information of the input bitstream is discarded. One way to accelerate this computationally intensive task is by determining the coding information of the re-encode in parallel [7, 8]. Another way would be to not completely discard the coding information of the original input bitstream. Therefore, transcoding techniques focus on exploiting the information of the input bitstream in order to skip encoding decisions in the re-encoding step [9–11]. Examples of transcoding operations include spatial downscaling [12, 13], frame rate reduction [14, 15], requantization [16, 17], and changing the compression format [18, 19].

This chapter proposes an efficient HEVC-based transcoder for spatially misaligned sequences that can be applied to any misalignment of an input sequence. To achieve this, both a trivial and a machine learning method are presented. The latter method also introduces a probability threshold, which allows the model to only apply decisions with a higher confidence than the threshold. This results in a transcoder that allows a trade-off between complexity and quality depending on the available resources.

The rest of the chapter starts with a description of related work on transcoding. Section 3.3 then describes the proposed methods followed in Section 3.4 by an in-depth analysis of the parameters for the machine learning method. The results are presented in Section 3.5. Finally, Section 3.6 ends with the conclusion.

3.2 Related work

In other works on transcoding, motion re-estimation is often used in order to reduce complexity during re-encoding [12, 18]. However, in HEVC, CTUs of 64×64 pixels can be recursively split into CUs which can be as small as 8×8 pixels. These CUs can then be split further into eight possible PU modes which are used for inter prediction (or two in the case of intra prediction) and are shown in Figure 3.3. These modes are $2N\times2N$, $2N\times N$, $N\times2N$, $N\times N$, and four asymmetrical partitioning modes, with the size of the entire CU being $2N\times2N$ pixels. With the first mode, the entire CU consists of one PU of size $2N\times2N$. For modes $2N\times N$, $N\times2N$, and the asymmetrical modes, the CU is split into two PUs of a size as described by the mode. For mode $N\times N$, the CU is split into four PUs. Consequently, even when motion re-estimation is applied, the most optimal block structure still needs to be determined.

Another way to reduce complexity is by using mode mapping in order to limit the amount of partitioning modes that the encoder needs to test. In related transcoding works, machine learning techniques have often been applied to find a correlation between coding information of the input bitstream and the partitioning modes of the output bitstream [20–28]. The applied techniques include

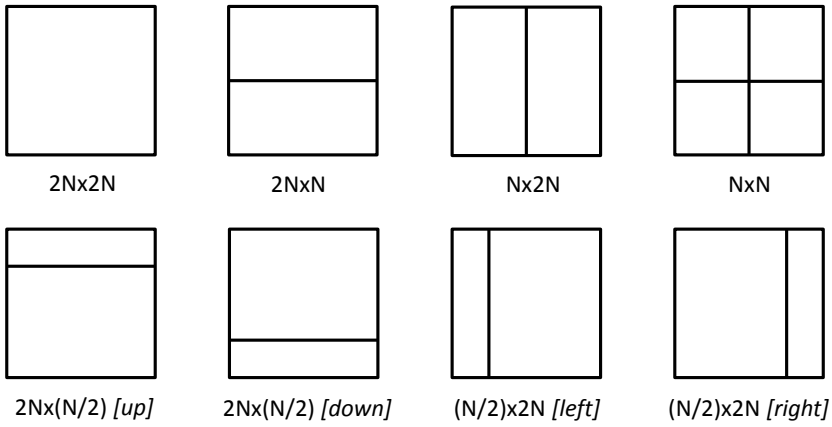


Figure 3.3: Different possible PU partitioning modes for a CU of size $2N \times 2N$ in case of inter prediction.

support vector machines (SVM) [20, 21], decision trees [22–26], and linear discriminant functions [27, 28]. Some of these algorithms have also been compared to each other, showing that a random forest performed the best in the case of spatial transcoding [29]. Several algorithms have been trained offline [20–26], i.e., using a subset of sequences for training and a separate set for evaluation. However, online-trained models, which are trained on the first k frames of a sequence and are then applied to the rest of the sequence, are better adapted to the content of the current sequence [27–29].

Many of the above papers focus on transcoding using older standards than HEVC. Applying these techniques on HEVC is impossible due to the more complex coding tools of HEVC compared to its predecessors. Despite this difference in coding tools, some research tries to bridge the gap between the different compression technologies by translating an older compression standard to HEVC [19, 27, 28]. However, because of the big difference in compression standards, techniques that try to learn these conversions are suboptimal to be used as transcoding techniques within the same standard.

Transcoding in HEVC itself has been considered for transrating [23] and spatial transcoding [29], but in these cases the encoder has been accelerated only by limiting CU partitioning decisions. Complexity in HEVC can be further reduced by also limiting PU partitioning modes, which is part of the proposed method in this chapter.

None of the above papers have considered transcoding of spatially misaligned sequences. This type of transcoding in HEVC has only been considered in previous work by creating a mode mapping model based on an offline-trained decision tree [26]. This tree was trained for pixel-shifts of 32, 16 and 8 pixels. However,

this means that the model is restricted to these shifts, which limits the applicability and does not allow the end user to freely arrange the videos in the composition. Moreover, only CU decisions were skipped, meaning that all possible PU partitioning modes are still evaluated for each predicted CU.

As their main novelty, the methods proposed in this chapter overcome the drawbacks of being restricted to certain shifts and of only accelerating CU decisions. This is achieved by proposing two transcoding methods that can be applied to all possible pixel-shifts and further improve complexity reduction by predicting PU information as well. Moreover, the second method achieves an improved trade-off between encoding complexity and compression efficiency.

3.3 Proposed transcoding methods

An efficient transcoder for misaligned sequences is a transcoder that reduces the complexity of re-encoding while minimizing the loss of compression efficiency. To achieve this, coding information from the input bitstream can be used in order to guide the CU structure and PU mode decisions of the shifted output bitstream. This information is then used during re-encoding to limit the number of evaluations of CU sizes and PU partitioning modes. However, if the predicted CU structures and PU modes are not optimal, this might result in a reduced compression efficiency. Therefore, it is important to have an accurate prediction model for any form of spatial misalignment.

In this section, the coding information extracted from the input bitstreams is presented first. This information is then used in two different methods that predict CU and PU information by detecting whether a block in the output bitstream should be split. The first is a trivial method based on the correlation between the input and output block structures and uses the same model for all shifts and sequences. On the other hand, the second method generates online-trained machine learning models, which means that a unique model is generated for each shifted sequence during transcoding.

3.3.1 Extraction of coding information

To determine the coding information of the shifted video, sixteen features are examined. These features are based on the coding information of the original video and are extracted from the input bitstream during transcoding. More specifically, they are extracted from the co-located input blocks, i.e., the blocks in the input bitstream that contain the same block of pixels as the block in the output bitstream for which coding information needs to be determined. For example, the region of co-located blocks in the shifted video in Figure 3.4a is marked in red for the original bitstream in Figure 3.4b. The features used in this chapter are the following:

- the fraction of the co-located blocks that are intra-coded (intraFraction)
- the mean, variance, maximum, and minimum CU depth (mean_CUdepth, var_CUdepth, max_CUdepth, and min_CUdepth)
- the mean, variance, maximum, and minimum TU depth (mean_TUdepth, var_TUdepth, max_TUdepth, and min_TUdepth)
- the mean, variance, maximum, and minimum PU depth (mean_PUdepth, var_PUdepth, max_PUdepth, and min_PUdepth)
- the variance of the input motion vectors (mvVariance)
- the mean and the variance of the transform coefficient variance (mean_DCTvar, var_DCTvar)

The first of the sixteen features is the intra fraction. This value indicates the percentage of pixels in the co-located input blocks that are intra-coded. This feature was selected in order to take possibly different behavior into account that may depend on the amount of intra-coded blocks in the input bitstream, since intra-blocks have only two possible PU partitioning modes contrary to inter-coded blocks, and they do not have any motion vectors either.

Twelve more features are based on the depth of the block structures in the input bitstream. The depth of a block refers to the number of times a CTU has been split. Therefore, for Coding Units CUs and Transform Units TUs, depth 0 refers to a block of 64×64 pixels, whereas depth 4 refers to a block of 4×4 pixels. For PUs, the depth d_{PU} is defined as

$$d_{PU} = \begin{cases} d_{CU} & \text{if } PU_{part\_size} = 2N\times2N \\ d_{CU}+1 & \text{other} \end{cases} \tag{3.1}$$

with d_{CU} being the depth of the CU to which the PUs belong, PU_{part_size} being the partitioning size of the PUs, and $2N\times2N$ being one of the eight possible partitioning sizes. For each block type (CU, PU, and TU), four features (mean, variance, maximum, minimum value) are calculated for the depths of the co-located input blocks. The use of these features is based on the assumption that the CU structure of a spatially shifted picture will show high correlations with the original block structures.

Another feature is the motion vector variance, which is defined as

$$\sigma_{mv}^2 = \sigma_x^2 + \sigma_y^2 \tag{3.2}$$

with σ_x^2 and σ_y^2 respectively being the variance of the x- and y-component of the motion vectors in the co-located input blocks. This feature is used to measure the similarity between the motion vectors of the co-located blocks, since it is assumed

that for a small σ^2_{mv}, a good match could be found for the current output block size, meaning that it will not need to be split. If motion vector variance information is unavailable (as is the case with intra-coded blocks), it is not used.

The last two features are the variance σ^2_{DCT} and the mean μ_{DCT} of the transform coefficient variance, defined as

$$\sigma^2_{DCT} = 16\sigma^2_y + \sigma^2_u + \sigma^2_v \tag{3.3}$$

and

$$\mu_{DCT} = \frac{4\mu_y + \mu_u + \mu_v}{6} \tag{3.4}$$

with σ^2_i and μ_i respectively the variance and mean of the i-component of the transform coefficient variance. These features are used since the transform coefficient variance will be zero if a block in the input bitstream is skipped. This information might be useful to combine with motion vector variance to predict splitting behavior of the output block.

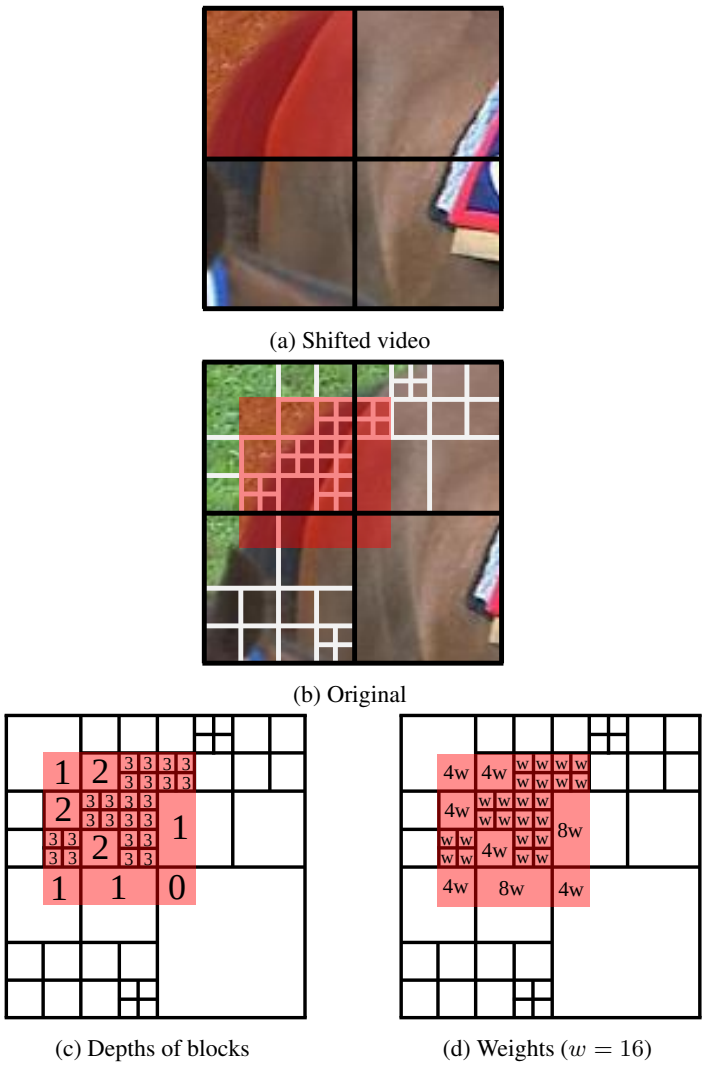
Due to misalignment with the CTU-grid, some of the above features that use mean and variance need to be weighted. This need for weighting is illustrated in Figure 3.4 in the case of a shift of 16 pixels in both x- and y-direction. In this figure, the block for which information needs to be predicted is not aligned with the CTU-grid of the original input bitstream (the red square does not align with the black lines). For example, to determine the mean CU depth of the co-located block, the depth of each CU within the co-located block is multiplied with a weight determined by the amount of overlap between the co-located block and the CUs in the original bitstream. The resulting value is then normalized. Consequently, the mean CU depth is 1.875 in the depicted example.

3.3.2 Trivial method

In the model described in [26], the CU structure is retained for blocks if the shift is a multiple of the block size for which the splitting behavior needs to be predicted. E.g. for a shift of 32 pixels, the CU structure of blocks of 32 pixels and smaller is retained. This rule is based on the high correlation between the CU structure of the input and output bitstream for certain shifts. However, this specific rule can only be applied for certain combinations of shifts and block sizes. Since the goal of this chapter is to transcode any spatially misaligned sequence, the rule based on correlation between CU structures should be generalized. This is done as follows.

For each CTU in the output bitstream, the CU structure is predicted based on the mean CU depth of the co-located blocks in the input bitstream. Only if this value is greater than the CU depth of the currently evaluated output block, the block is split. The same procedure is recursively applied for the new blocks.

After predicting the CU structure, a higher complexity reduction can be achieved by predicting PUs as well. For a subset of three sequences (*BasketballDrillText*,



(a) Shifted video

(b) Original

(c) Depths of blocks

(d) Weights ($w = 16$)

Figure 3.4: Shift of 16 pixels in both x- and y-direction. To find features for the CU of 64×64 *pixels in the shifted video, the info from the co-located block marked by the red square in the original is used. Note that this block is not aligned with the CTU-grid of the original (black lines in (b)).*

FourPeople, and *RaceHorses*) encoded with quantization parameters 22, 27, 32, and 37, the PU partitioning mode of each CU size is $2N \times 2N$ in at least 77% of the cases (Figure 3.5). For a full test set of 22 sequences (as shown in Table 3.3), this mode occurs at least 74%. Therefore, the greatest complexity reduction can be gained by predicting when a CU should be encoded with this PU mode.

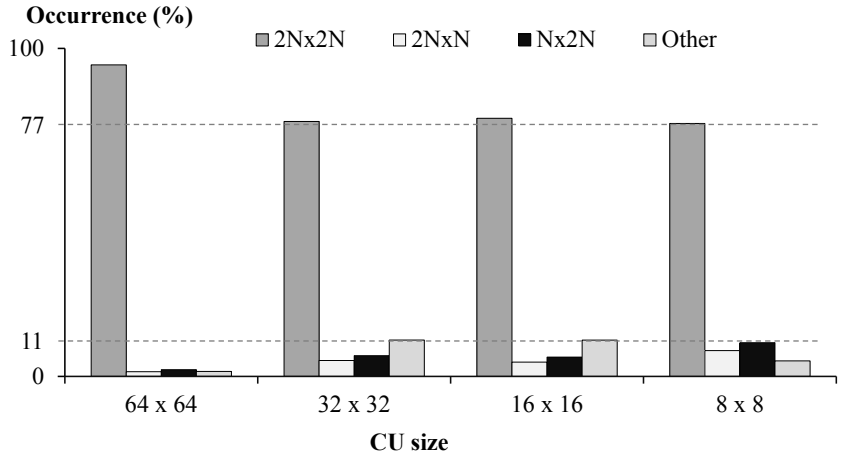


Figure 3.5: Relative occurrence of PU partitioning modes for each block size for a subset of three sequences. $2N \times 2N$ occurs in more than 77% of the cases for all block sizes.

Based on these observations, the trivial method is further expanded for PUs by examining the PU mode of the co-located input blocks if the output CU is not split. If the mode of all the co-located blocks of a predicted CU is $2N \times 2N$, only this mode is evaluated by the encoder. In all other cases, all PU modes are evaluated.

3.3.3 Machine learning method

One of the drawbacks in [26] and other transcoding algorithms based on an offline-trained model is the reliance on a diverse training set. This might cause the resulting model to be too general. This means that the model only takes features into account that correlate the best with all of the training data. However, the relative importance of features for determining splitting behavior might be different for each sequence. Moreover, offline-training is impractical to apply if all forms of spatial misalignment must be taken into account, since there are 64×64 possible combinations of shifts.

The second proposed method for predicting CU and PU information overcomes the above drawbacks by applying an online-trained machine learning algorithm, which is content-adaptive and applicable for all shifts. This machine learning model is trained on the first k inter-coded frames of a sequence, and creates a unique model that predicts the block structures of the remaining frames. Such a model consists of several sets of CU and PU classifiers, one for each block size in the output bitstream. A CU classifier determines whether a block of that size should be split (i.e., whether the split flag equals 1, as mentioned in section 2.2), whereas a PU classifier determines if only mode $2N \times 2N$ should be tested by the encoder for a CU that was not split.

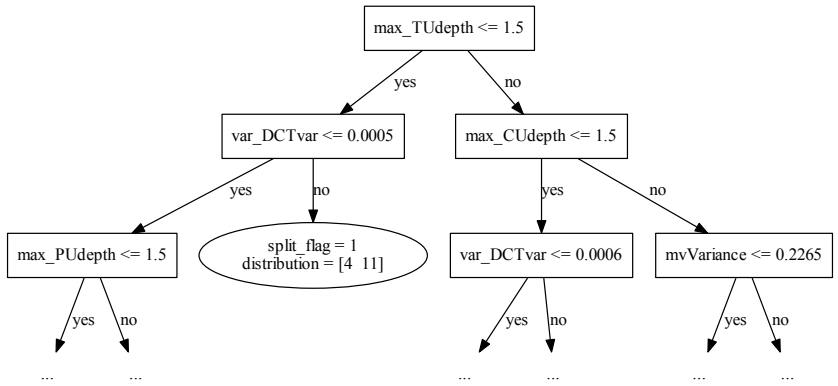


Figure 3.6: Example of a decision tree with 756 samples. At each inner node (rectangles), a rule is evaluated until a leaf node is reached. At the leaf node (ellipse), a decision is made based on the distribution of split and unsplit samples in the node. Since 11 out of 15 samples were split in the shown leaf node, any test samples that end up in this node will be split with a confidence of 73%.

The classifiers are trained based on the random forest algorithm [30], since this algorithm was shown to have the best performance out of several machine learning algorithms in the case of spatial transcoding [29]. The random forest algorithm generates multiple decision trees. Each of these decision trees uses only a random subset of all candidate features and a random subset of all training samples. Such a single decision tree is built by determining rules in each node with the goal of maximizing entropy reduction for the given samples [31]. In this chapter, a node is not split any further if a split would result in a node containing less than 1% of the total number of samples used in the tree. An example of such a tree is shown in Fig. 3.6. Given an input sample with $max\_TUdepth \leq 1.5$ and $var\_DCTvar > 0.0005$, the tree would predict that the sample is split with a probability of 73%, since the class distribution shows that 4 training samples were not split and 11 were split. Implementation-wise, predicting the class of a sample with a tree can thus be thought of as a large set of if-else statements.

A single tree can be sensitive to noise and outliers in the data. On the other hand, a random forest has a higher noise robustness due to averaging over the different trees. Moreover, other machine learning algorithms may have a high computational complexity, such as SVM [32] with a training complexity of the order $O(n^3 \cdot m)$, with n the number of samples and m the number of features. On the other hand, random forests have fast training times of the order $O(p \cdot m \cdot n^\dagger \cdot log(n^\dagger))$, with p the number of trees. Note that for random forests $n^\dagger$ is used since each tree only uses a random subset of the total number of samples. Due to this low complexity, random forests are a good fit to use in an online-training algorithm.

In order to predict the output class of a new input sample, after determining the probability of each possible output class in each of the decision trees, the outcomes of the trees are combined by using either a voting or an averaging strategy. Since in this chapter an averaging strategy is used, the probabilities of each tree are averaged to determine the outcome of the random forest with the highest probability. Due to this averaging, the random forest is less prone to overfitting than a single decision tree.

Since the implementation of the random forest algorithm used in this chapter averages the probabilistic prediction of each tree for each class instead of using the voting strategy, a probability p is obtained for each class [33]. By comparing this probability with a threshold, only decisions that are more certain than this threshold will be enforced in the encoder. If the probability is lower than the threshold, the encoder will evaluate all options. The aim of this threshold is to allow a trade-off between complexity reduction and quality.

3.4 Parameter analysis for machine learning

In order to obtain good results for the machine learning method presented in the previous section, it is necessary to analyze the effect of certain parameters on the performance of the algorithm. This section describes the test conditions used to evaluate the performance of the transcoding algorithms in this chapter. Under these conditions, the parameters of the machine learning algorithm are then tested.

3.4.1 Test conditions

Both methods described in Section 3.3 were implemented in version 12.0 of the HEVC reference software [34]. However, since the algorithms work on a high level to determine CU and PU structures, they can also be applied to other HEVC encoders in order to achieve complexity reduction. To evaluate the methods, 22 sequences of varying content and resolution [35] were pre-encoded using a quantization parameter of 22, 27, 32, and 37, with the low-delay configuration using only P-frames and with a constant quantization parameter for each frame. The full test set is also shown in Table 3.3. It should be noted that both class E and F contain special types of content, with E focusing on video conferencing with static backgrounds, and F containing sequences with special characteristics such as screen content. All bitstreams were decoded, shifted, and re-encoded with the same configuration and quantization parameter as the original encoding.

Each test was conducted on a single core of machines running on a dual-socket octa-core Intel Xeon Sandy Bridge (E5-2670) processor @ 2.6 GHz. For class A, the transcoding process had access to 6 GB RAM, whereas up to 2 GB RAM was allowed for transcoding sequences from other classes. Since all tests are executed under the same conditions, time can be used as a relative measure for complexity.

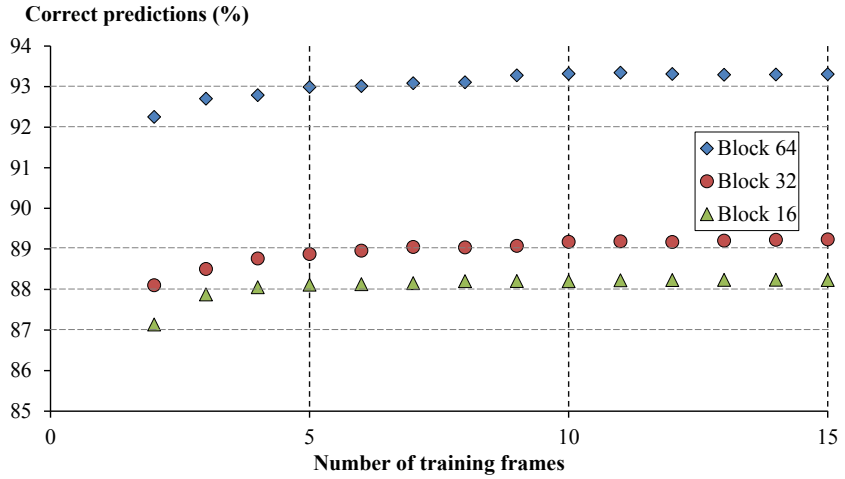


Figure 3.7: Relationship between number of training frames and correct predictions of the split flag for each block size, averaged over four shifts for three sequences. Between two and five training frames, the correctness increases by one percent, whereas it stops increasing after ten frames.

Complexity reduction is determined by comparing the encoding time of the fast encoder T_{fast} to the encoding time of the reference encoder T_{ref} in terms of time saving as defined in equation (2.1).

The difference in compression efficiency is expressed in BD-rate [36]. This metric shows the average increase in bit rate of a sequence re-encoded with the fast encoder compared to a sequence re-encoded with the reference encoder for the same PSNR. The PSNR for a sequence is calculated by decoding a shifted bitstream and comparing it to the original, uncompressed video.

To shift a sequence L pixels in a certain direction, the picture is first padded with L pixels at either the top or left edge. Then, rows and/or columns of CTUs containing padding are cropped in order to avoid any influence of the artificial padding on the results. Unless stated otherwise, a shift of L pixels refers to shifting the picture by L pixels in both x- and y-direction.

3.4.2 Parameter analysis

Using the above conditions, the following parameters are analyzed: the size of the training set, the number of trees in the random forest, and the number of relevant features. In this analysis, the sixteen features described in Section 3.3 are used as an initial feature set.

Since more data beats a cleverer algorithm [37], it is important to select a sufficiently large training set. However, since each training frame is fully re-encoded,

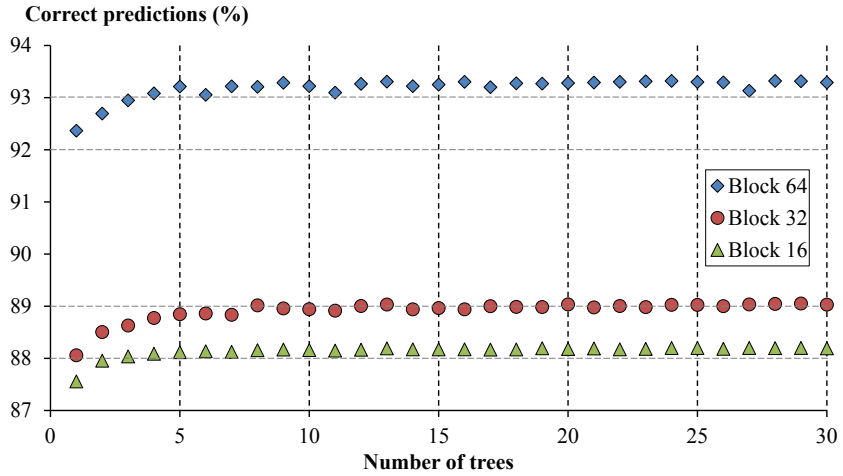


Figure 3.8: Relationship between number of trees in the forest and correct predictions of the split flag for each block size, averaged over four shifts for three sequences. After a certain number of trees in the forest, the correctness converges to a limit.

the training set should be kept as small as possible. In order to determine the optimal number of training frames, a CU classifier was trained for each block size for the sequences *BasketballDrillText*, *FourPeople*, and *RaceHorses*. These sequences were selected from the full set of sequences due to their diverse characteristics. Four different shifts were applied to each sequence: 0, 8, 16, and 32 pixels in both x- and y-direction. The number of trees was fixed at 100, and the initial set of sixteen features was used (mentioned in Section 3.3.1). For each classifier, the relative amount of correct predictions of the split flag was calculated for the test frames (Figure 3.7). The correct predictions of the classifier for each block size increase by 1% between using two[1] and five training frames. In terms of compression efficiency and complexity, this means that the average BD-rate is reduced by 0.33%, whereas the time saving is only reduced by 0.28% due to fully re-encoding three extra frames. Between five and ten training frames, the increase in prediction correctness is smaller. It is most noticeable for a block size of 64, since the amount of available samples in each frame is smaller for larger block sizes. The average BD-rate is only further reduced by 0.1%, whereas the time saving is again reduced by 0.28%. Beyond ten frames, the increase in correctness becomes negligible compared to the reduction of the time saving. Therefore, ten training frames will be used.

[1]Note that the results for one frame are not displayed in the figure because the percentage of correct predictions is very low. This is because the first inter-frame of the *BasketballDrillText* sequence is identical to the first (intra-)frame of the video, meaning that the frame is entirely coded using CUs with the lowest possible depth (using skip-mode), resulting in irregular behavior for one training frame.

An important parameter of the random forest algorithm is the number of trees in the forest [38]. If the number of trees grows, both the performance and computational complexity of the algorithm increase. However, after a certain number of trees, there is no further significant gain in performance, whereas the complexity continues increasing. The optimal number of trees is determined similarly to the number of frames, using ten training frames and the same features. On average, it is observed that the correctness of the predictions converges to a limit after five trees (Figure 3.8). However, since small fluctuations in the correctness were observed more often when the number of trees is smaller, twenty trees are used in the rest of this chapter[2].

The performance of a machine learning algorithm is also affected by the number of features [37, 39]. If there are many irrelevant features, the algorithm will perform worse because of the introduced noise. However, using only a small amount of features might introduce a larger generalization error. Consequently, it is important to identify a sufficient number of relevant features. The initial set of sixteen features was used as a starting point to train the CU and PU classifiers. Additionally, the same sequences and shifts were used as for the analysis of the number of frames and trees.

The random forest algorithm allows the calculation of the relative feature importance during training [30]. This value is calculated for each tree in the forest as the expected fraction of the samples a feature contributes to. Consequently, features used in higher decision nodes of the tree will have a greater importance. The calculated value is then averaged over all the trees in the forest. Finally, the relative importance of each feature is averaged over all block sizes and tested shifts (as shown in Table 3.1).

The importances of the features for the CU classifier show that the CU structures of the co-located blocks have a high correlation with the output CU structure. Other features related to block structures have a high importance as well. There is less than 1% difference between the importance of mvVariance, max_TUdepth, and var_DCTvar. However, there is a difference of more than 1% in importance between both the sixth and seventh feature, and ninth and tenth feature. Therefore, the performance was evaluated for using all sixteen features, as well as for both nine and six features. On average, for the three tested sequences with four shifts, the time saving was 71.5% for either amount of features, whereas the BD-rate was 2.74% for sixteen features, 2.76% for nine features, and 3.06% for six features. Since the compression efficiency for nine and sixteen features is similar, the top nine features will be used to train the CU classifier for the full test set.

For the PU classifier, motion vector variance appears to have the greatest importance in determining whether a PU in the output bitstream should be encoded

[2] In terms of training time, this increases from about 500ms for one tree to 750ms for 30 trees, with some fluctuations due to the shared nature of the used computing cluster.

CU classifier		PU classifier	
Feature	Importance %	Feature	Importance %
mean_CUdepth	15.3	mvVariance	28.7
max_CUdepth	12.7	mean_PUdepth	18.0
min_CUdepth	11.4	max_PUdepth	14.9
mean_PUdepth	11.0	min_PUdepth	14.7
max_PUdepth	9.1	mean_DCTvar	5.7
mean_TUdepth	8.2	var_DCTvar	3.8
mvVariance	6.6	mean_CUdepth	2.7
max_TUdepth	5.9	mean_TUdepth	2.7
var_DCTvar	5.7	var_PUdepth	2.2
min_TUdepth	3.7	var_TUdepth	1.8
mean_DCTvar	3.7	var_CUdepth	1.4
var_PUdepth	3.5	max_CUdepth	1.0
var_TUdepth	1.3	max_TUdepth	0.9
min_PUdepth	1.1	min_CUdepth	0.8
var_CUdepth	0.8	min_TUdepth	0.6
intraFraction	0.1	intraFraction	0.2

Table 3.1: Relative importance of each feature for a subset of three sequences when training the CU and PU classifier.

as $2N \times 2N$. Most likely, if the motion vectors in the co-located blocks are similar, there is no need to split the CU further into PUs. Information about the PU structure of the input bitstream also appears to be important in order to determine the PU partitioning in the output. Since there is a difference of 9% in importance between the fourth and fifth most important feature, only the top four features will be used to train the PU classifiers for the full test set.

3.5 Results

In this section, the trivial method and machine learning method are evaluated under the same test conditions as in the previous section. The compression efficiency of different shifts is examined, followed by an evaluation of the trade-off between complexity reduction and coding performance.

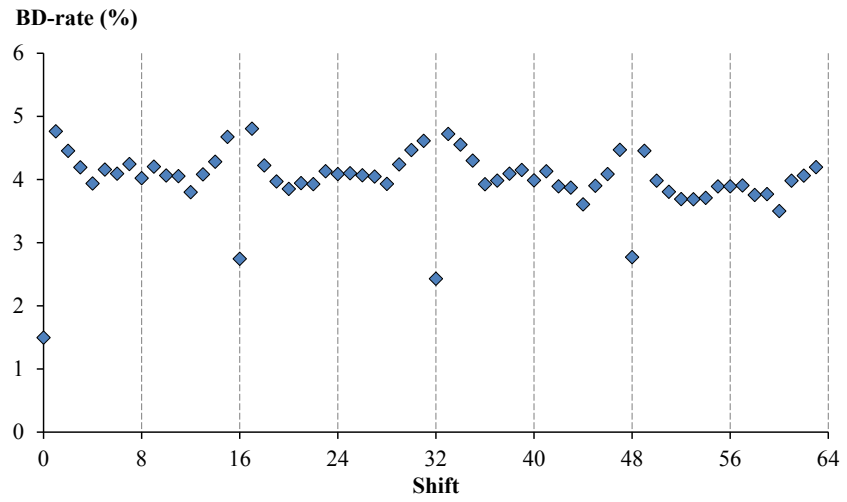


Figure 3.9: BD-rate (%) for 64 different shifts applied to the sequence BQTerrace. The behavior between shifts of 0 and 32 appears to resemble a mirrored version of the behavior between shifts of 32 and 64.

3.5.1 Compression efficiency of shifts

In the following set of experiments, the behavior of predicting the CU structure is evaluated for all possible shifts, up to a shift of 63 pixels, since 64 pixels equals the block size of a CTU. This means that all shifts above 64 pixels have an equivalent shift between 0 and 63 pixels. Similarly, negative shifts are not considered either. Moreover, as seen with an example sequence (Figure 3.9), the BD-rates of shifts between 33 and 63 pixels resemble a mirrored version of the BD-rates between 1 and 32 pixels. This behavior indicates that a shift of L pixels is equivalent to a shift of $64 - L$ pixels. Since a shift of $64 - N$ pixels is similar to applying a shift of $-L$, this means that the direction of a shift has no effect on the compression efficiency. Therefore, only shifts between 0 and 32 pixels will be considered in the rest of the experiments.

The compression efficiency of the models depends on the shifts (Figure 3.10). For both the trivial and machine learning model, no shift at all performs the best in terms of BD-rate, followed by a shift of 32 pixels, and 16 pixels. Since these shifts better retain the alignment with a grid, the CU structure is preserved better [26]. For example, in case of a shift of 32 pixels, blocks of 32×32 pixels and smaller might retain their CU structure after shifting. This results in a higher correlation between the input and output bitstream, and therefore results in a better prediction model. Since a better model will predict more efficient encoding decisions, the BD-rate is lower for these shifts.

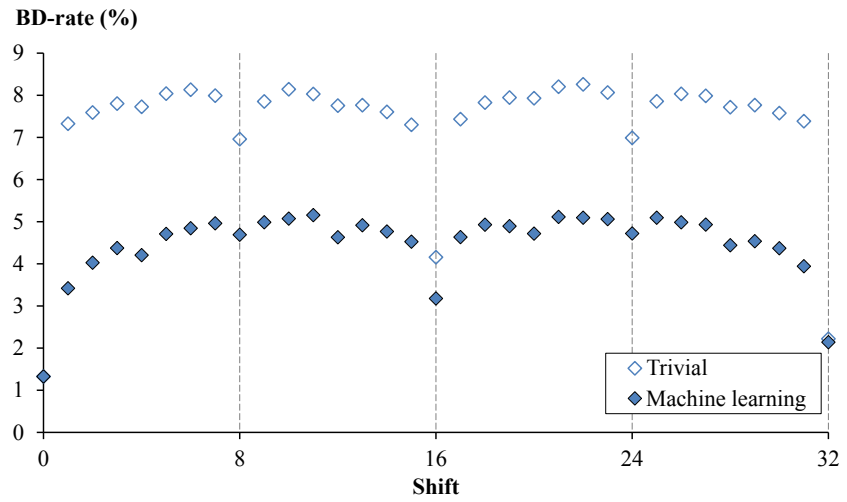


Figure 3.10: Average compression efficiency in BD-rate (%) for the trivial and machine learning model. Shifts that cause more misalignment with the CTU-grid result in a higher BD-rate. For these shifts, the machine learning model performs better than the trivial model.

Compared to the trivial model, the machine learning model results in a significantly lower BD-rate for shifts that cause more misalignment with the original grid of CUs (Figure 3.10). This shows that for such shifts, a complex model is necessary in order to maintain a good compression efficiency. For shifts of 0 or 32 pixels, the trivial model performs similar to the machine learning model. In these cases, usage of the trivial model might be preferred due to its simplicity in implementation.

For both the trivial and machine learning model, a shift of 0 pixels shows an average BD-rate increase of 1.33%. This is caused by some optimizations in the HM encoding software that speed up the encoding process [40]. These optimizations require information that is calculated during CU evaluations that are skipped by the models. Since this information is not available, this will result in a lower compression efficiency.

In the previous experiments, shifts of L pixels in both x- and y-directions were considered. However, in many scenarios it is necessary to shift a sequence asymmetrically with (L, M) pixels, which consists of an x-shift of L pixels and a y-shift of M pixels with $L{\neq}M$. As seen in Table 3.2 for the machine learning model, a shift of $(8, 32)$ pixels results in a BD-rate of 4.26%, whereas a shift of $(32, 8)$ pixels results in a BD-rate of 4.12%. On average, the difference in BD-rate between an (L, M) shift and an (M, L) shift is smaller than 0.2%, suggesting that both are equivalent. Moreover, the direction of the shift that causes the greatest

	y-shift				
x-shift	0	8	16	24	32
0	1.33	4.05	2.92	4.11	1.96
8	4.05	4.69	4.48	4.72	4.26
16	2.74	4.42	3.18	4.38	2.93
24	4.17	4.71	4.49	4.72	4.27
32	1.76	4.12	3.01	4.20	2.14

Table 3.2: Average BD-rate (%) for asymmetric shifts with the machine learning model.

misalignment with the grid appears to affect the BD-rate the most. For example, the BD-rate of a shift of $(16, 32)$ is 2.93%. This increases to 4.42% for a shift of $(16, 8)$. On the other hand, between a shift of $(16, 32)$ and $(16, 16)$, the BD-rate only increases to 3.18%. For the trivial model, similar behavior was observed.

In all of the above results, the transcoding model predicts the complete CU structure. This means that the time saving is similar for all shifts, since only a single CU structure is evaluated by the encoder. Small variations between shifts may occur if the model predicts large CU sizes more often, since this reduces the number of evaluated CUs. However, the time saving does not show a correlation with the shifts, contrary to the BD-rate (see also Table 3.3).

3.5.2 Complexity-scalable prediction

The previous subsection showed that several shifts are equivalent to one another in terms of compression efficiency. It was also determined that the shifts such as 0, 32, and 16 pixels that retain alignment with a grid result in a better coding performance. Moreover, asymmetric shifts are mostly influenced by the direction of a shift causing the greatest misalignment. Therefore, in this subsection, the experiments are only conducted for symmetric shifts of 0, 32, 16, and 8 pixels.

As described in Section 3.3, the machine learning method allows setting a threshold for the confidence of predictions. This allows to trade encoder speed-up for compression efficiency by only skipping decisions if the confidence is higher than the threshold. Both a threshold t_{CU} for CU predictions, and a threshold t_{PU} for PU predictions can be set.

In a first set of experiments, only CU information is predicted, meaning that only t_{CU} is modified. This threshold was varied from 0.9 to 0.5 with steps of 0.1. When $t_{CU} = 0.5$, all predictions are accepted, which results in the highest amount of speed-up. However, a confidence close to 0.5 means that the decision has a high probability to be incorrect, which results in a lower compression efficiency. For a

Table 3.3: Results for selected shifts for the full test set. Only CU information is predicted, with $t_{CU} = 0.5$.

				BD-rate (%)				Time saving (%)			
Class	Sequence	Resolution	Frames	Shift 0	Shift 32	Shift 16	Shift 8	Shift 0	Shift 32	Shift 16	Shift 8
A	PeopleOnStreet	3840×2160	150	1.02	1.30	1.77	3.74	64.68	65.93	64.60	67.06
	Traffic	3840×2048	300	1.53	2.25	3.13	4.84	75.30	74.65	73.53	74.55
	Average			1.28	1.77	2.45	4.29	69.99	70.29	69.06	70.80
B	BasketballDrive	1920×1080	500	1.06	1.68	2.95	4.79	66.96	67.57	69.25	71.75
	BQTerrace	1920×1080	600	1.50	2.43	2.75	4.02	71.84	71.78	71.86	72.88
	Cactus	1920×1080	500	1.19	1.95	3.57	5.67	70.85	70.93	71.66	73.55
	Kimono1	1920×1080	240	1.05	1.81	3.89	3.67	71.22	71.56	71.70	74.53
	ParkScene	1920×1080	240	1.41	2.03	2.93	4.64	72.20	72.15	72.24	72.57
	Average			1.28	2.06	3.28	4.50	71.53	71.61	71.86	73.39
C	BasketballDrill	832×480	500	0.81	1.36	2.33	4.67	71.60	71.18	69.09	71.54
	BQMall	832×480	600	1.20	2.01	2.70	5.05	72.92	72.61	71.35	71.59
	PartyScene	832×480	500	0.66	0.91	1.62	3.47	69.33	69.60	68.61	68.46
	RaceHorses	832×480	300	1.00	1.37	2.13	3.43	65.80	65.88	65.11	67.60
	Average			0.92	1.41	2.20	4.16	69.91	69.82	68.54	69.80

Continued on next page

Table 3.3 – *Continued from previous page*

Class	Sequence	Resolution	Frames	BD-rate (%)				Time saving (%)			
				Shift 0	Shift 32	Shift 16	Shift 8	Shift 0	Shift 32	Shift 16	Shift 8
D	BasketballPass	416×240	500	0.86	1.06	1.88	3.35	67.29	68.37	68.11	66.23
	BlowingBubbles	416×240	500	0.73	1.07	1.84	2.99	68.53	68.33	69.57	66.23
	BQSquare	416×240	600	0.73	1.38	1.85	2.47	70.97	70.99	71.66	67.56
	RaceHorses	416×240	300	0.93	1.54	2.20	3.61	65.21	65.83	66.24	63.44
	Average			0.81	1.26	1.95	3.10	68.00	68.38	68.89	65.86
E	FourPeople	1280×720	600	1.89	2.66	4.48	7.15	79.58	78.40	77.93	77.83
	Johnny	1280×720	600	2.94	4.56	4.63	5.49	80.20	79.20	79.01	78.81
	KristenAndSara	1280×720	600	2.34	3.42	5.00	5.65	79.01	78.16	78.11	78.26
	Average			2.00	2.98	4.02	5.35	76.70	76.03	75.99	75.19
F	BasketballDrillText	832×480	500	0.82	1.19	2.15	4.82	71.59	71.48	69.78	71.11
	ChinaSpeed	1024×768	500	1.69	2.40	3.24	4.48	58.21	61.83	65.67	67.80
	SlideEditing	1280×720	300	1.87	3.30	5.99	5.73	81.34	80.69	81.52	80.93
	SlideShow	1280×720	500	1.96	5.38	6.89	9.47	74.43	74.73	75.71	75.75
	Average			1.58	3.07	4.56	6.13	71.39	72.18	73.17	73.90
Average for all sequences				1.33	2.14	3.18	4.69	71.32	71.45	71.47	71.82

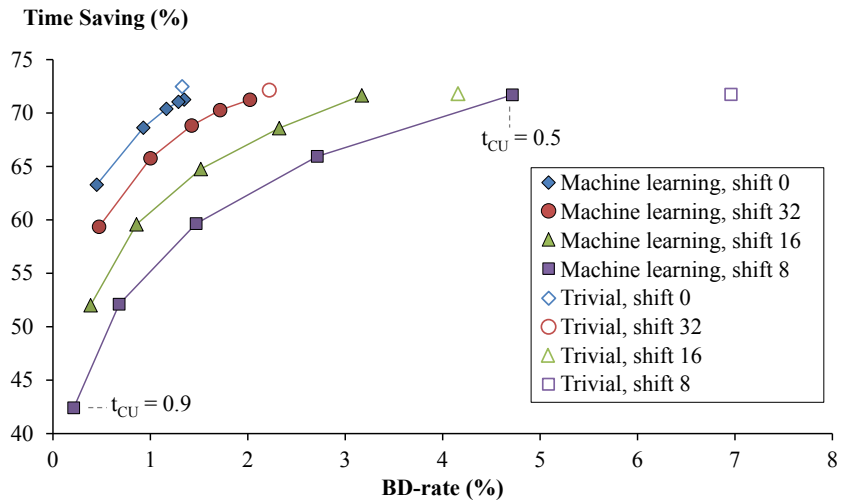


Figure 3.11: Average trade-off between BD-rate (%) and Time Saving (%) for CU prediction. Contrary to the trivial method, the machine learning method can trade encoder speed-up for compression efficiency by modifying the confidence threshold t_{CU}.

shift of 8 pixels, these low confidences appear to occur more often. For example, at $t_{CU} = 0.5$ both shift 8 and shift 16 have a similar time saving, whereas at $t_{CU} = 0.9$ shift 8 has a time saving of 42% while shift 16 has a time saving of 52% (Figure 3.11). This implies that the machine learning algorithm can predict more CUs with a high confidence for a shift of 16 pixels. This is likely caused by the worse misalignment of shift 8 compared to shift 16, which makes it harder for the algorithm to find a good correlation between input information and the output CU structure. For shifts 16, 32, and 0, the complexity reduction for $t_{CU} = 0.9$ is progressively higher, indicating that predictions for these shifts have a higher confidence.

Figure 3.11 also shows that for a shift of 0 and 32 pixels, the trivial CU prediction method has a slightly higher speed-up than the machine learning method. However, the latter method only requires half the bit rate overhead for a penalty in speed-up of only 8% for shift 0 and 5% for shift 32. This makes the machine learning method useful if bit rate constraints are stricter than the complexity reduction requirements.

In a second group of experiments, prediction of PU information is allowed as well. In addition to t_{CU}, t_{PU} is now also varied. When comparing the machine learning method to the trivial method with both CU and PU prediction, it is observed that the former can achieve a higher compression efficiency for a similar speed-up if the shift is no multiple of 16 pixels (as seen in Figure 3.12).

When predicting both CU and PU information, time saving increases up to 84% for the machine learning method and 82% for the trivial method compared to

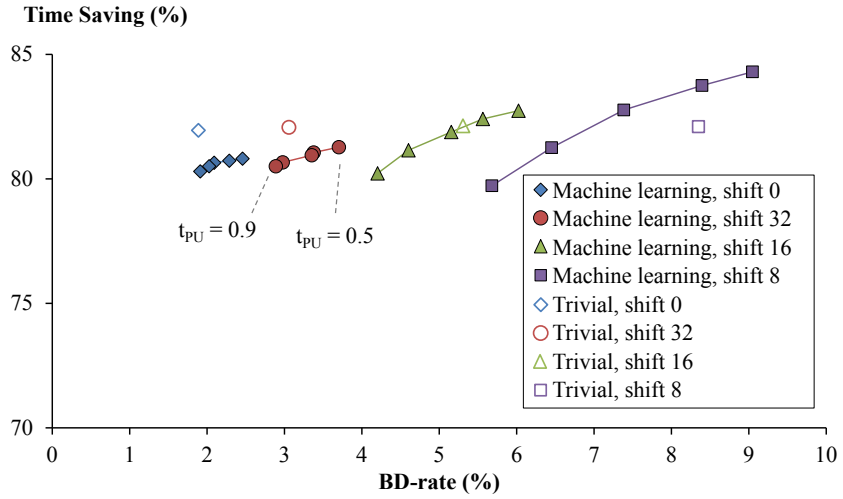


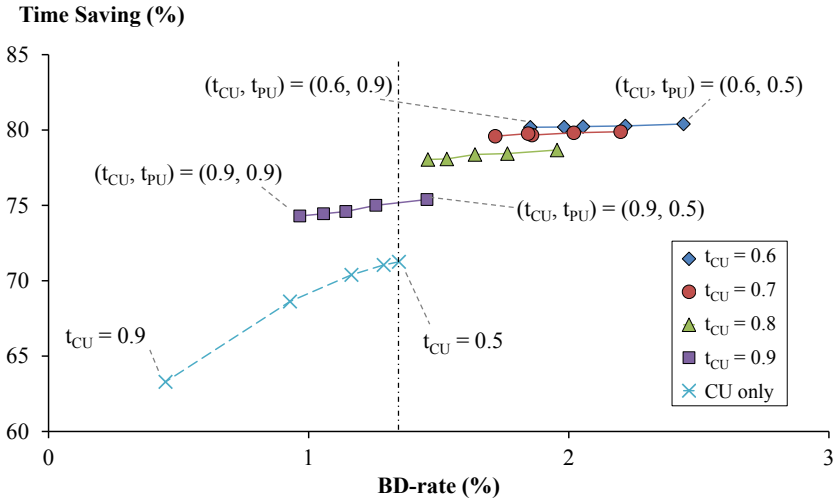
Figure 3.12: Average trade-off between BD-rate (%) and Time Saving (%) for prediction of both CUs and PUs when modifying the confidence threshold t_{PU}. $t_{CU} = 0.5$ for the machine learning method. This method achieves a higher speed-up for the same compression efficiency as the trivial method for shifts that are no multiple of 16 pixels.

the original encoder (Figure 3.12). However, the BD-rate also increases compared to predicting only CU information (e.g. in Figure 3.13 from 3.17% BD-rate to 4.20% between $t_{CU} = 0.5$ and $(t_{CU}, t_{PU}) = (0.5, 0.9)$ for shift 16). An explanation for this increase might be that forcing a PU partition of $2N \times 2N$ limits the possibilities for the encoder to correct a non-optimal decision for the CU structure. If only the CU information is predicted wrongly, the encoder can still correct this by selecting a PU partitioning that will produce a good match for the motion estimation.

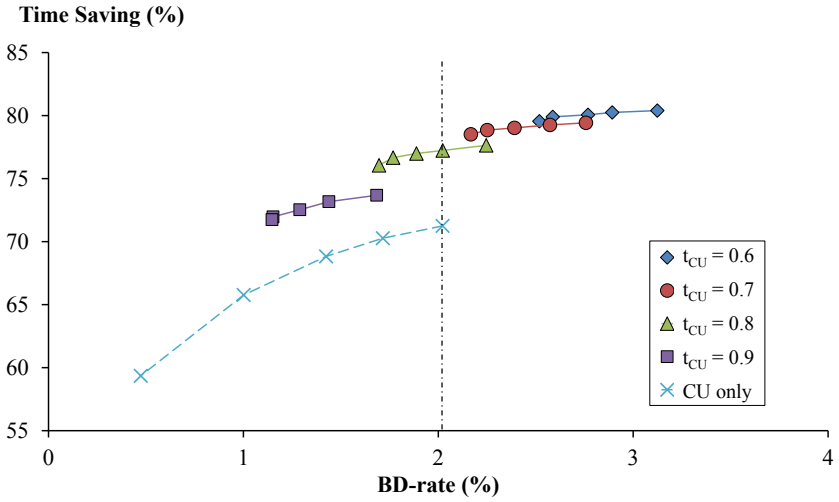
Although predicting PU information increases the BD-rate, given the right combination of PU threshold and CU threshold, the complexity-scalable PU prediction can achieve a higher complexity reduction for the same BD-rate as the CU prediction (Figure 3.13). This suggests that in order to achieve the best performance for the machine learning method, both CU and PU information should be predicted.

3.5.3 Comparison with existing work

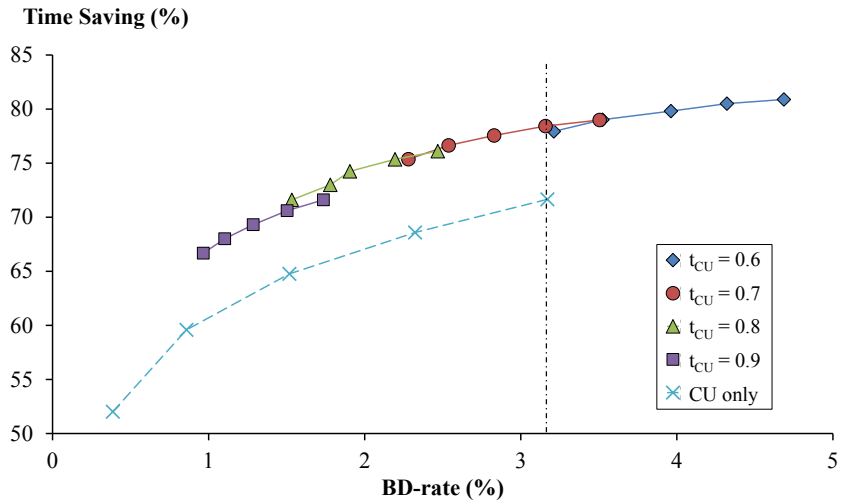
In order to show the effectiveness of the proposed complexity-scalable method for transcoding spatially misaligned sequences in HEVC, the results are compared to the existing work based on an offline-trained decision tree in [26]. Since this model only supports transcoding for shifts that are a multiple of 8, the results are compared for shifts of 32, 16 and 8 pixels.



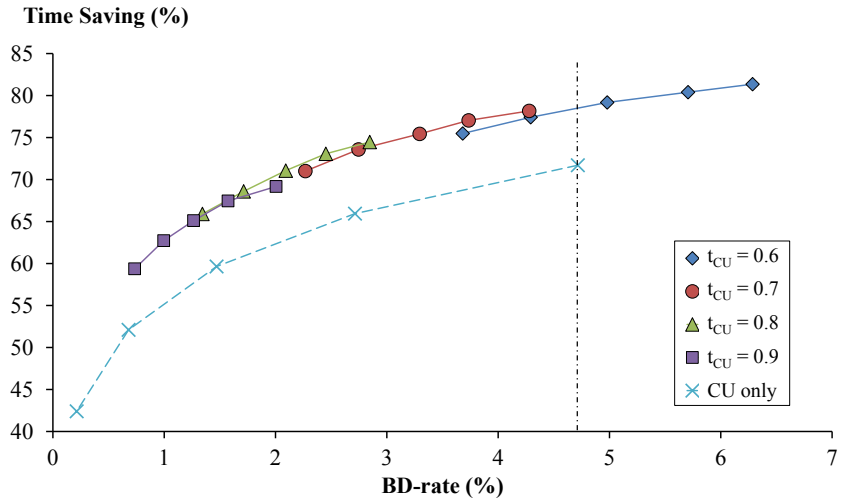
(a) Shift 0



(b) Shift 32



(c) Shift 16



(d) Shift 8

Figure 3.13: Comparison between predicting only CU information and both CU and PU information with the machine learning method when modifying confidence thresholds t_{CU} and t_{PU}. Predicting both CU and PU information results in a higher speed-up for the same compression efficiency regardless of the shift.

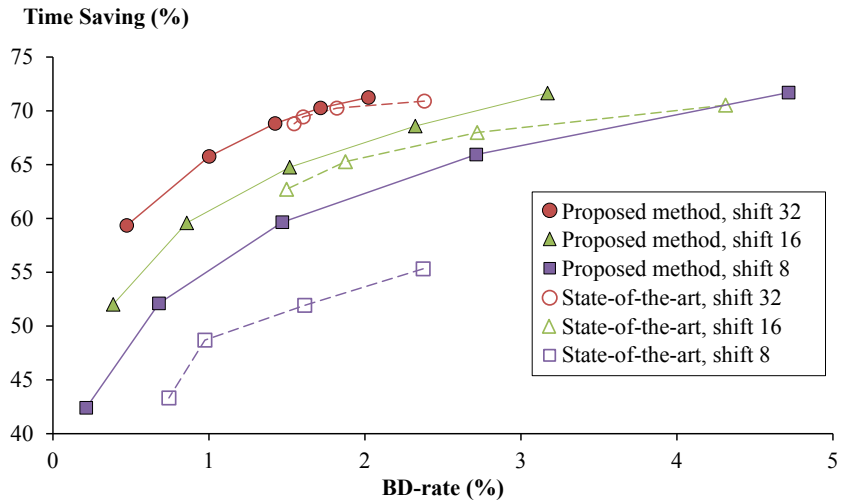


Figure 3.14: Comparison between the proposed method and state-of-the-art [26]. For any threshold in the existing work, a better point in terms of time saving or BD-rate can be found for the proposed method.

The existing work also supports complexity control with the splitting threshold t_{split} and stopping threshold t_{stop}. Threshold t_{split} was fixed at 0, since this results in a good trade-off between BD-rate and time saving, while t_{stop} was varied from 0.25 to 1 with steps of 0.25. As seen in Figure 3.14, the proposed model outperforms the existing work for all shifts when predicting only CUs.

Additionally, for a shift of 32 pixels, the difference between [26] and the proposed model is smaller than for a shift of 16 or 8 pixels. This illustrates that the proposed model can adapt better to more difficult shifts.

3.6 Conclusion

In this chapter, two methods were proposed in order to transcode any spatially shifted video sequence. First, a trivial method focuses on simplicity of implementation and results in a general model that can be applied to any shift. A second, machine learning method uses the random forest algorithm to create a unique model for each misaligned video sequence. The coding decisions made by either method are guided by the coding decisions made by the non-shifted version of the video in order to speed up transcoding by skipping CU and PU decisions during the re-encoding step of the shifted sequence.

Evaluation of the methods shows a better performance when the shifts align with a grid. Additionally, the bit rate increase of asymmetric shifts is mainly determined by the direction that causes the most misalignment. For achieving the

highest complexity reduction of 82% with a BD-rate smaller than 3%, the trivial method can be used for shifts of 0 and 32 pixels. However, for other shifts, the performance of the machine learning method is better. Moreover, the machine learning method performs best when predicting both CU and PU information, achieving high complexity reductions for limited bit rate overheads.

In conclusion, using coding decisions from a non-shifted version of the video in order to guide the encoding of a spatially misaligned sequence succeeds in enabling fast encoding for any amount of misalignment of the input videos with the CTU-grid of the entire composition.

References

[1] G.J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) Standard*. IEEE Trans. Circuits Syst. Video Technol., 22(12):1649–1668, Dec. 2012.

[2] J. Ohm, G.J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand. *Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC)*. IEEE Trans. Circuits Syst. Video Technol., 22(12):1669–1684, Dec. 2012.

[3] C.-H. Li, H. Lin, C.-N. Wang, and T. Chiang. *A fast H.264-based picture-in-picture (PIP) transcoder*. In Proc. IEEE Int. Conf. Multimedia Expo (ICME), pages 1691–1694, June 2004.

[4] Y. Michalevsky and T. Shoham. *Fast H.264 picture in picture (PIP) transcoder with B-slices and direct mode support*. In Proc. IEEE Mediterranean Electrotechnical Conf. (MELECON), pages 862–867, April 2010.

[5] D. Grois, M. Loants, O. Hadar, R. Ohayon, and N. Amram. *Ultra-fast live video-in-video insertion for H.264/AVC*. In Proc. IEEE Int. Conf. Consum. Electron. (ICCE), pages 635–636, Jan. 2013.

[6] J. De Praeter, J. De Cock, G. Van Wallendael, S. Van Leuven, P. Lambert, and R. Van de Walle. *Efficient Picture-in-Picture Transcoding for High Efficiency Video Coding*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 514–515, Sept. 2013.

[7] C. Yan, Y. Zhang, J. Xu, F. Dai, J. Zhang, Q. Dai, and F. Wu. *Efficient Parallel Framework for HEVC Motion Estimation on Many-Core Processors*. IEEE Trans. Circuits Syst. Video Technol., 24(12):2077–2089, Dec. 2014.

[8] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, and F. Wu. *A Highly Parallel Framework for HEVC Coding Unit Partitioning Tree Decision on Many-core Processors*. IEEE Signal Process. Lett., 21(5):573–576, May 2014.

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

[10] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang. *Video transcoding: an overview of various techniques and research issues*. IEEE Trans. Multimedia, 7(5):793–804, Oct. 2005.

[11] J. Xin, C.-W. Lin, and M.-T. Sun. *Digital Video Transcoding*. Proc. IEEE, 93(1):84–97, Jan. 2005.

[12] Y.-P. Tan and H. Sun. *Fast motion re-estimation for arbitrary downsizing video transcoding using H.264/AVC standard*. IEEE Trans. Consum. Electron., 50(3):887–894, Aug. 2004.

[13] V. Patil, R. Kumar, and J. Mukherjee. *A Fast Arbitrary Factor Video Resizing Algorithm*. IEEE Trans. Circuits Syst. Video Technol., 16(9):1164–1171, Sept. 2006.

[14] C.-T. Hsu, C.-H. Yeh, C.-Y. Chen, and M.-J. Chen. *Arbitrary Frame Rate Transcoding Through Temporal and Spatial Complexity*. IEEE Trans. Broadcast., 55(4):767–775, Dec. 2009.

[15] C.-H. Yeh, S.-J. Fan Jiang, C.-Y. Lin, and M.-J. Chen. *Temporal Video Transcoding Based on Frame Complexity Analysis for Mobile Video Communication*. IEEE Trans. Broadcast., 59(1):38–46, Mar. 2013.

[16] 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, Apr. 2010.

[17] 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 Trans. Multimedia, 18(3):364–378, Mar. 2016.

[18] Q. Tang and P. Nasiopoulos. *Efficient Motion Re-Estimation With Rate-Distortion Optimization for MPEG-2 to H.264/AVC Transcoding*. IEEE Trans. Circuits Syst. Video Technol., 20(2):262–274, Feb. 2010.

[19] W. Jiang, Y. Chen, and X. Tian. *Fast transcoding from H.264 to HEVC based on region feature analysis*. Multimedia Tools and Applications, pages 1–22, Sept. 2013.

[20] X. Jing, W.-C. Siu, L.-P. Chau, and A.G. Constantinides. *Efficient inter mode decision for H.263 to H.264 video transcoding using support vector machines*. In Proc. IEEE Int. Symposium Circuits Syst. (ISCAS), pages 2349–2352, May 2009.

[21] 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, Oct. 2012.

[22] J.L. Martinez, G. Fernandez-Escribano, H. Kalva, W. A. C. Fernando, and P. Cuenca. *Wyner-Ziv to H.264 video transcoder for low cost video encoding*. IEEE Trans. Consum. Electron., 55(3):1453–1461, Aug. 2009.

[23] L. Pham Van, J. De Cock, G. Van Wallendael, S. Van Leuven, R. Rodriguez-Sanchez, J.L. Martinez, P. Lambert, and R. Van de Walle. *Fast transrating for high efficiency video coding based on machine learning*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 1573–1577, Sept. 2013.

[24] G. Fernandez-Escribano, J. Bialkowski, J.A. Gamez, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Kaup. *Low-Complexity Heterogeneous Video Transcoding Using Data Mining*. IEEE Trans. Multimedia, 10(2):286–299, Feb. 2008.

[25] G. Fernandez-Escribano, H. Kalva, J.L. Martinez, P. Cuenca, L. Orozco-Barbosa, and A. Garrido. *An MPEG-2 to H.264 Video Transcoder in the Baseline Profile*. IEEE Trans. Circuits Syst. Video Technol., 20(5):763–768, May 2010.

[26] 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 Proc. IEEE Int. Conf. Image Process. (ICIP), pages 2494–2498, Oct. 2014.

[27] T. Shanableh, E. Peixoto, and E. Izquierdo. *MPEG-2 to HEVC Video Transcoding With Content-Based Modeling*. IEEE Trans. Circuits Syst. Video Technol., 23(7):1191–1196, July 2013.

[28] E. Peixoto, T. Shanableh, and E. Izquierdo. *H.264/AVC to HEVC Video Transcoder Based on Dynamic Thresholding and Content Modeling*. IEEE Trans. Circuits Syst. Video Technol., 24(1):99–112, Jan. 2014.

[29] L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Performance analysis of machine learning for arbitrary downsizing of pre-encoded HEVC video*. IEEE Trans. Consum. Electron., 61(4):507–515, Nov. 2015.

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

[31] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[32] C. Cortes and V. Vapnik. *Support-vector networks*. Machine Learning, 20(3):273–297, 1995.

[33] 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 E. Duchesnay. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825–2830, Oct. 2011.

[34] Il-Koo Kim, Ken McCann, Kazuo Sugimoto, Benjamin Bross, Woo-Jin Han, and Gary Sullivan. *High Efficiency Video Coding (HEVC) Test Model 12 (HM12) Encoder Description*. Technical Report JCTVC-N1002, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Aug. 2013.

[35] F. Bossen. *Common test conditions and software reference configurations*. Technical Report JCTVC-L1100, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Jan. 2013.

[36] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

[37] P. Domingos. *A Few Useful Things to Know About Machine Learning*. Commun. ACM, 55(10):78–87, Oct. 2012.

[38] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas. *How many trees in a random forest?* In Machine Learning and Data Mining in Pattern Recognition, pages 154–168. Springer, 2012.

[39] I. H. Witten and F. Eibe. *Transformations: Engineering the Input and Output*. In Data Mining: Practical Machine Learning Tools and Techniques, chapter 7, pages 285–343. Morgan Kaufmann, 2005.

[40] I.-K. Kim, W.-J. Han, J. H. Park, and X. Zheng. *CE2: Test results of asymmetric motion partition (AMP)*. Technical Report JCTVC-F379, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), July 2011.

4

Personalized Views Extracted from Ultra-High-Resolution Video

4.1 Introduction

A second type of personalization of video bitstreams focuses on guiding coding decisions of video encoders that each encode a personalized view extracted from ultra-high-resolution video in order to create a more immersive experience for each viewer. Such an immersive experience is difficult to create in current consumer set-ups. After all, watching a live broadcast of an event such as a sports match or a music concert through traditional television is a static experience. Contrary to sitting in the stadium or concert hall, the viewer is restricted to a single viewpoint. As a result, viewers are merely observers who are disconnected from the actual event. However, with the advent of new technology such as VR glasses, this static experience will evolve into a more immersive one, making viewers feel as if they are actually present at the event.

In order to allow users to truly experience this immersion, panoramic or 360-degree video content is necessary. This will allow the user to look around in the scene, unrestricted by a single viewpoint, and will thus increase the level of immersion. However, delivery of such video content to the home is not trivial, since the resolution of this video is far beyond 3840×2160 pixels, which is the highest resolution consumer displays and distribution channels currently support. Resolutions beyond this ultra-high resolution lead to even higher bit rates, which cannot be transported over the limited bandwidth capacity of users at home.

A common approach to the above problem is to split the high-resolution video into tiles, which are separately encoded video streams. A subset of these tiles is then sent to the client depending on the desired view. However, this system still requires either an extra component that will join the separate video streams into one for decoding with a standard decoder, or the client needs multiple decoders to decode the multiple streams. Moreover, depending on the size of the separate tiles, some tiles will contain pixel data that is not displayed within the current view, resulting in a waste of bandwidth capacity.

A better solution is to send only the required view to the client. For example, sending an entire 360-degree video with a resolution of 8192×4096 pixels at 30 frames per second encoded with HEVC results in a bit rate of 23.4 Mbps, whereas sending only a view of 1920×1080 pixels results in a bit rate of 4.2 Mbps, which is less than one-fifth of the original bit rate. However, since each viewer may be looking at a different region of the video, a separate, personalized video is encoded for each client. Since encoding is computationally complex, such a system requires a high amount of resources. Therefore, as the first main contribution in this chapter, a fast personalized-view approach is proposed in order to reduce the complexity of each individual encode by using information obtained from a pre-analysis of the entire ultra-high-resolution video. Such an approach typically results in some bit rate overhead. As such, the second main contribution in this chapter is to investigate how the overhead of the personalized-view approach compares to the tile-based approach.

The remainder of this chapter first gives an overview of the state-of-the-art in Section 4.2. Section 4.3 then describes the overall architecture of a system that provides personalized views. The fast encoding of the personalized views is then presented and evaluated in Section 4.4, with further reductions of complexity being proposed in Section 4.5. The latter method is then compared to the tile-based approach in Section 4.6. Finally the conclusion is given in Section 4.7.

4.2 Related work

The traditional approach to provide users with more interactive video, the tile-based method, has already been thoroughly discussed in the literature [1–4]. This technique was mostly applied to the H.264/AVC codec. In this approach, the high-resolution video is downsampled at the server to different resolutions (including a thumbnail) in order to provide zooming by having multiple resolution layers. These different layers are then subdivided into a grid of non-overlapping tiles and encoded. At the user side, the user selects his Region of Interest (RoI) view based on a thumbnail, which is a low-resolution overview of the entire video, or by using other input methods such as VR glasses. Next, the tiles contained within and intersecting with the RoI boundary for the requested resolution are streamed from

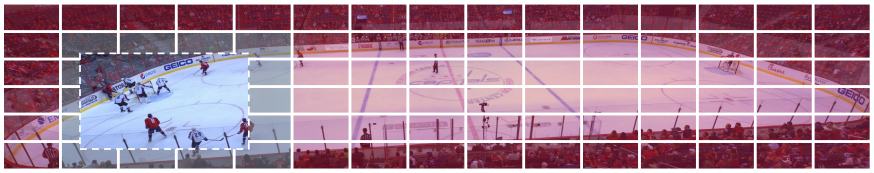


Figure 4.1: Illustration of the tile-based method. The tile borders are indicated by the white lines. The red areas are not transmitted to the user. The gray areas indicate the pixels that are not displayed on the user side.

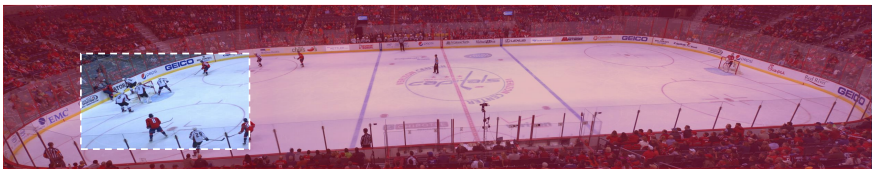


Figure 4.2: Illustration of the personalized-view method. The red area is not transmitted to the user. Only the required view is sent to the user side.

the server. These tiles are rendered at the user side and cropped to the appropriate resolution of the display if necessary.

The tile-based method has several disadvantages. A first disadvantage is that tiled streaming results in a bit rate overhead due to sending additional pixels outside the RoI that are not displayed at the user side, as illustrated in Figure 4.1. This is because some tiles may partially overlap with the RoI, since the RoI is unlikely to be aligned with tile boundaries. To reduce these wasted bits, one can reduce the dimension of the tiles. However, since each tile is encoded independently, small tiles lead to a lower compression ratio, increasing the number of bits needed for the RoI [5]. Another disadvantage is that the user also requires a customized video player to decode, combine and synchronize the tiled streams, which makes this approach harder to deploy. A third disadvantage is that the tiles need to have an encoding structure that allows random access, since the tiles can only be decoded starting from an intra-coded frame. Moreover, the intra-period must be small in order to allow low-delay panning, which leads to an increase in bit rate due to the lower compression efficiency of intra-coded frames. Finally, a structural delay is introduced, since repositioning the RoI to another region is only possible after decoding all frames within an intra-period. This structural delay can have a negative impact on very low-delay interactive applications such as viewing the video through VR glasses.

As a solution to the above disadvantages, a monolithic streaming method has been investigated by Quang Minh Khiem et al. [6]. In this method, only H.264/AVC macroblocks that belong to the RoI are sent to the viewer, together with the blocks

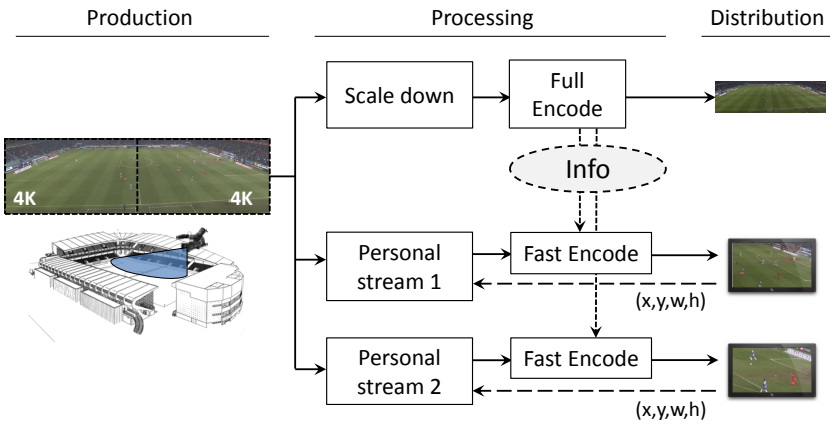


Figure 4.3: System architecture of a multi-viewpoint fast-encoding solution.

in previous frames that are used as a reference for the sent macroblocks. This results in more bandwidth-efficient streaming compared to the tile-based method. However, a random access coding structure is still used, meaning that the structural delay and the bit rate overhead of intra-frames remain.

4.3 System architecture

Due to many disadvantages of the tile-based method, a personalized-view technique is proposed in this chapter. This technique differs from the tile-based method by encoding the selected RoI of each user on the fly. Consequently, no extra pixel overhead is sent to the user, as illustrated in Figure 4.2. Another advantage of the personalized-view approach is that the user can use a standard decoder and has very flexible digital pan/tilt/zoom possibilities. A disadvantage is that the method is not scalable to a large number of users. In order to alleviate this problem, the encoding complexity of the individual encodes was lowered by reusing coding information of an encoded full ultra-high-resolution video in order to speed up the encoding process of the RoI of each user. An overall system architecture that incorporates the creation, processing and distribution of these personalized views is described and depicted in Figure 4.3.

The main goal of this system is to deliver a high amount of simultaneously encoded videos with different output resolutions, zoom levels and cropping parameters. In the scenario shown in Figure 4.3, multiple users can request different personalized views from within the same ultra-high-resolution video. The system then delivers these personalized HEVC streams encoded by fast encoders to the appropriate client. The coding decisions of these fast HEVC encoders are guided by the coding information from an HEVC encoder that encodes the entire panoramic

or 360-degree video. The overall system consists of three major parts: production, processing and distribution, with the main focus and contribution of this chapter lying within the processing step. In the following three paragraphs, each of the parts of the system will be highlighted briefly.

Production This part of the system describes the creation of the input content. In Figure 4.3, the content consists of two stitched 4K (3840×2160 pixels) camera images. This results in content of about 7680 by 2160 pixels. The creation and stitching [7] of this content goes beyond the scope of this dissertation. In general, any content creation process that results in a very high resolution image can be used as input to this system. Applications may range from sports as depicted in Figure 4.3, to surveillance and more industrial applications. The output of this step is directly linked to the processing step.

Processing By using coordinates provided by different users as input to the system, the processing step creates different views and encodes them as HEVC bitstreams ready for distribution to the users. Because a full encode of each view would result in a computationally complex system that requires one extra full encoder per extra viewer, this chapter describes techniques to reduce this overall complexity. The key idea behind these complexity reducing techniques is the use coding information extracted from a full encode of the ultra-high-resolution video in order to guide (and thus accelerate) the coding decisions of the encoders of the individual, personalized views of the same content.

Distribution The last step in the process is the video distribution. This is the part where the HEVC encoded bitstream is sent to the consumer devices. Because each user receives his own personalized view, the video streams can be optimized based on the device parameters, e.g. tablet users might request content with a full high-definition (HD) resolution of 1920×1080 pixels, while users wearing VR glasses require a different resolution. Note that the architecture requires bi-directional communication between the distribution and processing step due to the cropping parameters (x- and y- coordinates, and width w and height h of the cropped view in the figure) being sent from the distribution to the processing. Also note that due to the interactive nature of the system, the client and his personal encoder should be synchronized[1].

[1]For example, this synchronization could be done through the use of timestamps sent together with the video packets, and another timestamp sent from the client to the server to make sure that the selection of the view gets ignored if some past instructions arrive after future instructions due to network delays.

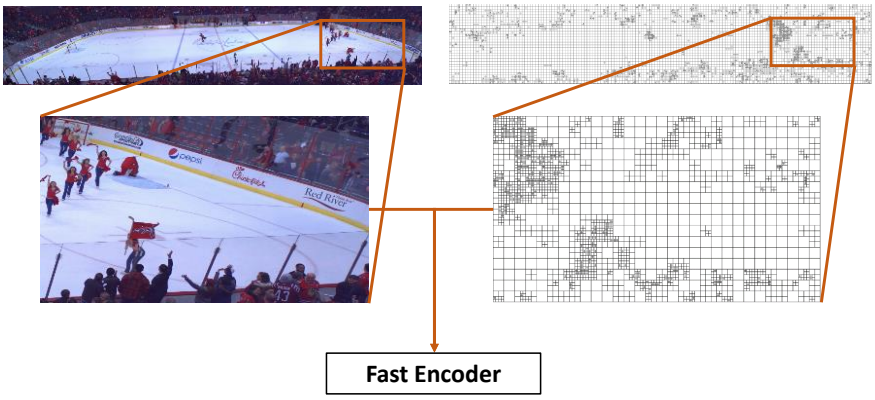


Figure 4.4: Accelerated personalized-view encoding. A part of the coding information (on the right) of the full ultra-high-resolution video is fed to the encoder together with the cropped view in order to accelerate the encoding of this view.

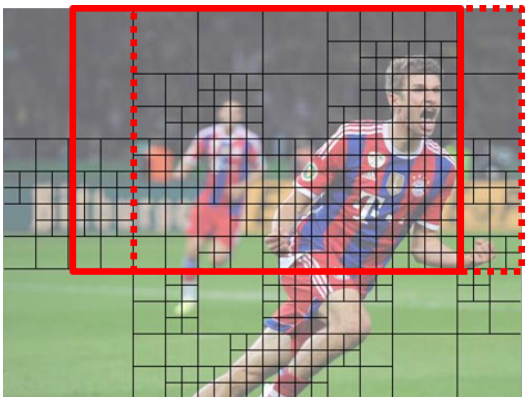
4.4 Extraction and encoding of views

HEVC uses a block structure as described in Chapter 2 to divide a frame into multiple small coding blocks. Since determining the optimal block structure of a frame is computationally complex, the encoder complexity can be greatly reduced by limiting the structures that should be considered. This can be done by exploiting coding information from the encoding of the original ultra-high-resolution video. This is illustrated in Figure 4.4, where the CU structure is extracted from the same location in the ultra-high-resolution video as the personalized view. Both the personalized view and this coding information are then fed to a fast encoder which has to make less encoding decisions.

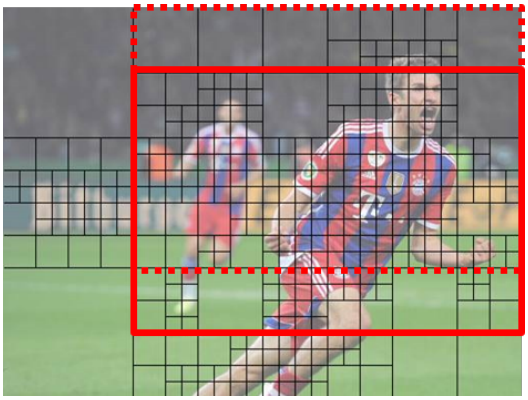
This section further investigates the viability of reusing CU structure decisions from the full ultra-high-resolution video in order to accelerate the encoding of the personalized views. The method is first introduced, followed by an evaluation and short discussion.

4.4.1 Method

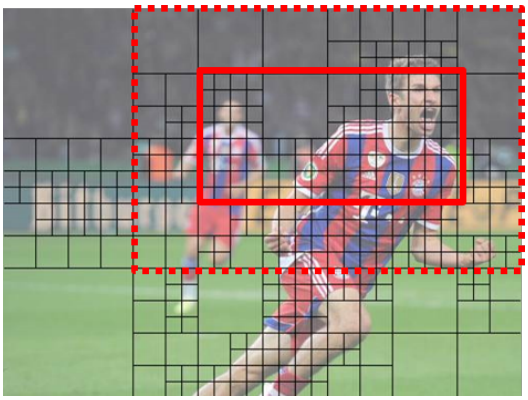
When allowing users to select a personalized view of an ultra-high-resolution video, three major movements within the video are defined. These movements are similar to the pan, tilt and zoom camera movement known from the field of image capturing. The first movement is panning within the content. As shown in Figure 4.5a, the pan operation retains the output size, but shifts the image left or right on the horizontal axis. The second movement depicted in Figure 4.5b corresponds with tilting. Tilting is similar to panning, except that the motion is upwards



(a) Pan

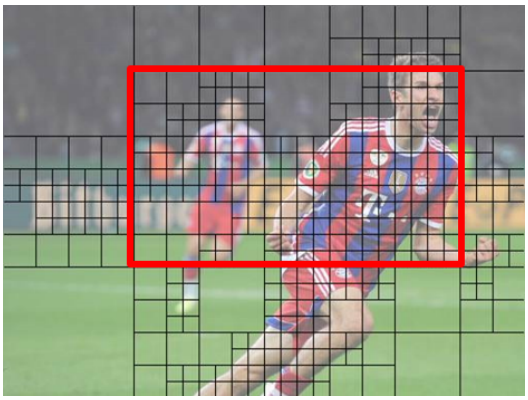


(b) Tilt

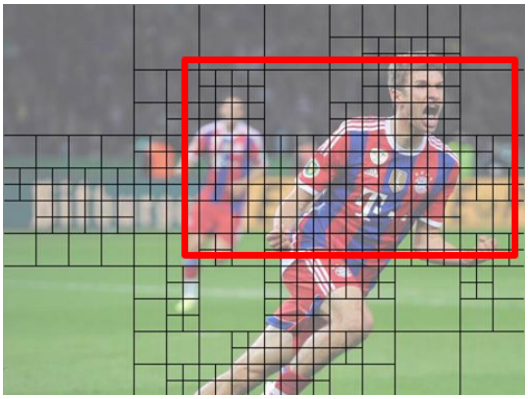


(c) Zoom

Figure 4.5: View adaptation conform to camera pan (a), tilt (b) and zoom (c).



(a) Alignment



(b) Misalignment

Figure 4.6: Difference between spatial alignment (a) and spatial misalignment (b) of selected views.

or downwards on the vertical axis. Finally, the third and last movement consists of zooming in or out on the content. As depicted in Figure 4.5c, the zoom operation creates a smaller image by cropping and shifting the original image while retaining the same aspect ratio.

Pan, tilt and zoom operations correspond to a combination of cropping, scaling and shifting the original picture to create a new viewpoint. The outcome of these operations can result in one of two scenarios for the personalized view: alignment or misalignment. In a first scenario, as depicted in Figure 4.6a, the view that the user selects is perfectly aligned with CTU-grid of the original image. In this case, the block structure of that part of the image can easily be reused, as was also the case in Chapter 3. However, similar to the creation of compositions in the previous

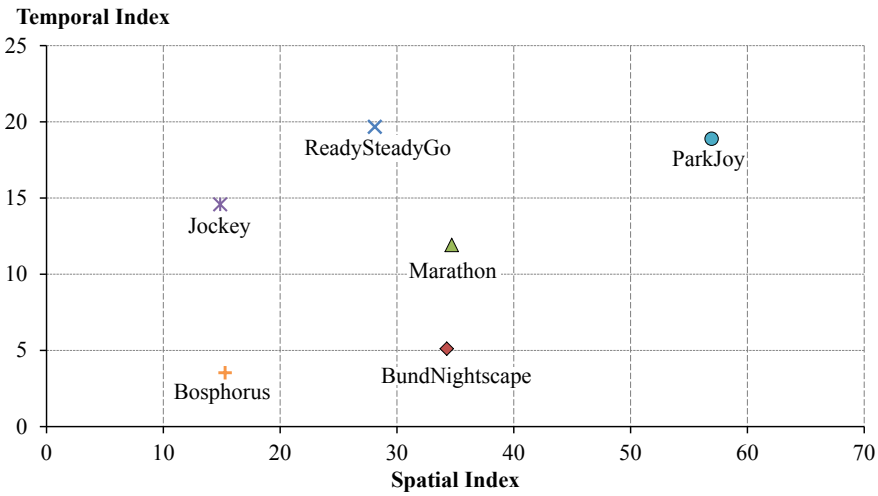


Figure 4.7: Description of the chosen test sequences in terms of spatial and temporal activity.

chapter, misalignment can also occur. In this case, as shown in Figure 4.6b, the boundaries of the personalized view do not align with existing block boundaries, making it impossible to simply copy the block structures of the full panoramic video. Nonetheless, since each view is created from the same input video, there is a certain amount of correlation between the personalized view and the original video, which should make it possible to predict CU structures for the personalized views based on the entire panoramic video.

In order to predict CU structures, a machine learning model is trained on the first 10 frames of each new view by using the Random Forest algorithm [8]. Since the problem of misalignment is similar to the case of fast transcoding of compositions, the same parameters and features are used as in the previous chapter. The machine learning algorithm thus creates an ensemble classifier from 20 decision trees, whereas the features are calculated based on the co-located blocks in the full ultra-high-resolution video.

4.4.2 Evaluation

In order to evaluate the algorithm for fast encoding of personalized views, version 16.3 of the HEVC reference software was modified [9]. The Ultra HD (UHD) sequences *Bosphorus*, *BundNightscape*, *Jockey*, *Marathon*, *ParkJoy* and *ReadySteadyGo*, which have a resolution of 3840×2160 pixels, were used as the original video from which personalized views are extracted [10]. These sequences were chosen from a larger pool of UHD sequences due to their diverse spatial and tem-

Sequence	Fps	Description
Bosphorus	120	Camera moves in parallel to follow boat
BundNightscape	30	Fixed camera and small moving objects
Jockey	120	Fast camera pan and zoom to follow horse
Marathon	30	Fixed camera and many moving objects
ParkJoy	50	Camera moves in parallel to follow people
ReadySteadyGo	120	Camera pans to follow action

Table 4.1: Short description of test sequences, including the frame rate in frames per second (fps). The total number of encoded frames equals ten times the frame rate with a maximum of 600 frames.

poral activity as measured using the Spatial Index (SI) and Temporal Index (TI) (Figure 4.7) described in the ITU-T Recommendation P.910 [11]. They are further described in Table 4.1 and shown in Figure 4.8.

Each UHD sequence was encoded with QP values of 22, 27, 32 and 37. Each of these versions was encoded using a low delay configuration, which consists of an I-frame followed by P-frames. This configuration results in a lower delay, which is a requirement for interacting with personalized views.

To evaluate the proposed algorithm, the differences in compression efficiency and encoding complexity reduction are measured. The difference in compression efficiency is expressed in BD-rate [12]. This metric shows the average increase in bitrate for the same PSNR of encoding a personalized view by reusing information from the original UHD sequence (fast encoder) compared to encoding this view without reusing information (reference encoder). Complexity reduction is determined in terms of time saving as given by equation (2.1).

In the following subsections, two scenarios are simulated and evaluated. First, a virtual zoom is simulated, followed by a pan- and tilt-scenario.

4.4.2.1 Simulating zoom

If a user with a 720p screen wants to view a part of the UHD video at the original resolution, the CU structure of the relevant part of the UHD video can be copied to encode this personalized view. However, if the user zooms out with a factor of two, the personalized view should reuse the information from a 1080p version of the original video. If the user zooms out even further to watch the complete video on the 720p screen, CU structure information for a 720p version needs to be determined. To simulate this scenario of zooming, the UHD video is encoded at three resolutions (2160p, 1080p and 720p) by using coding information of the UHD video encoded at 2160p.

(a) Bosphorus

(b) BundNightscape

(c) Jockey

(d) Marathon

(e) ParkJoy

(f) ReadySteadyGo

Figure 4.8: Ultra HD sequences used in the simulations.

The results in Table 4.2 indicate that performing a fast encode at resolutions other than 2160p does not differ greatly in compression efficiency (with the exception of *Jockey*) from performing a fast encode of the UHD sequence at its original resolution of 2160p. This seems to indicate that zoom levels of 1080p and 720p are indeed viable choices for providing personalized views. Additionally, all versions also show similar complexity reductions between 69% and 79%.

If the CU structure of encoding the UHD sequence at its original resolution is predicted, the machine learning algorithm reports a 100% accuracy since the encoder simply has to copy all of the original CU information. However, the simulation reported BD-rates between 2.4% and 4.7% (Table 4.2), which is far from the expected 0%. This behavior was investigated further and can be attributed to two encoder optimizations in the HM reference software.

The first encoder optimization speeds up inter-prediction of asymmetrical motion partitions [13]. This optimization uses the PU partitioning size of the parent CU to decide which asymmetrical motion partitions will be evaluated and whether full motion estimation will be performed in addition to merge estimation. If the evaluation of the parent CU is skipped, as is the case with the algorithm proposed in this section, its PU partitioning size is not known and the existing encoder optimization does not function correctly.

The second encoder optimization provides an extra candidate starting point for motion estimation [14]. This starting point is based on the motion vector of the most recently calculated $2N \times 2N$ PU with the same reference picture as the currently tested motion vector. However, if some CU sizes are not evaluated, this value can be incorrect, leading to less optimal encoder decisions.

4.4.2.2 Simulating pan and tilt

The effects of pans and tilts of the personalized view can be simulated by shifting the view on the UHD video in x- and y-direction. Since the size of CTUs in this dissertation is 64×64 pixels, shifts of k pixels are assumed to be equivalent to shifts of $k \mod 64$ pixels in terms of misalignment. Additionally, a shift of k pixels is also assumed to be equivalent to a shift of $-k$ pixels. Hence, only shifts of up to 32 pixels will be evaluated. To simulate these shifts, a view of 3808×2128 pixels is used on the UHD sequences. This view can shift up to 32 pixels in both x- and y-direction (see also Figure 4.9).

In a first experiment, the relation between shifts in different directions was investigated. As seen in Table 4.3 for the sequence *BundNightscape*, an (x, y) shift shows a similar compression efficiency as a (y, x) shift. For this sequence, the largest difference can be seen between e.g. a shift of (0,32) pixels with a BD-rate of 4.6% and (32,0) pixels with a BD-rate of 4.4%. Since the other sequences displayed similar behavior, only shifts in a single direction are used in the next experiment.

	BD-Rate(%)			Time Saving (%)		
Sequence	2160p	1080p	720p	2160p	1080p	720p
Bosphorus	4.3	4.6	5.1	78.8	77.8	77.4
BundNightscape	4.2	5.5	5.5	79.5	77.9	78.2
Jockey	4.7	8.2	9.3	78.0	77.6	77.5
Marathon	3.2	3.6	2.9	71.0	69.2	69.4
ParkJoy	2.4	2.5	2.1	72.1	69.2	69.2
ReadySteadyGo	4.5	5.6	5.5	77.1	74.3	73.8

Table 4.2: Results of the zoom simulation.

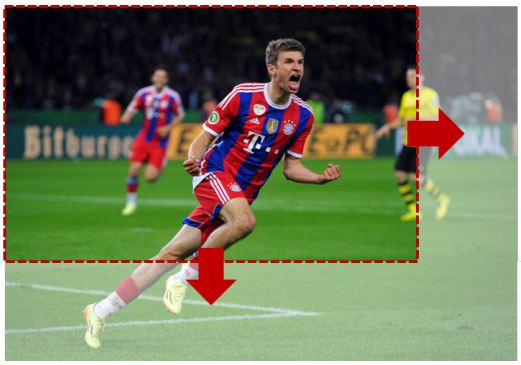


Figure 4.9: To simulate pan and tilt for shifts of up to 32 pixels in both x- and y-direction, the personalized view shows the complete video with 32 pixels subtracted from both its width and height.

	x-shift				
y-shift	0	8	16	24	32
0	4.2	5.7	5.1	5.7	4.4
8	5.7	6.0	6.0	6.2	5.6
16	5.2	5.9	5.4	6.0	5.1
24	5.8	6.2	6.2	6.2	5.9
32	4.6	5.7	5.2	5.7	4.5

Table 4.3: Effect on BD-rate when combining pan and tilt movements, for the sequence BundNightscape.

In a second experiment, the effect of misalignment was investigated in detail for all 32 shifts. As seen in Figure 4.10, some shifts perform better than others. Shifts of 0 and 32 pixels perform best since they respectively preserve alignment with the CTU-grid of 64×64 pixels and a CU-grid of 32×32 pixels. Shifts of 16 pixels perform slightly better than surrounding shifts, although this performance is worse than for shifts of 32 pixels. Depending on the sequence, shifts from 1 to 3 pixels and shifts of 30 and 31 pixels also perform generally better than other shifts. As in Chapter 3, this is most likely due to very small shifts only introducing a negligible amount of misalignment for some sequences.

In all of the above results, the fast encoder predicts the complete CU structure. As seen in Table 4.4, this means that the time saving is similar for all shifts, since only a single CU structure is evaluated by the encoder. Small variations between shifts may occur if the model predicts large CU sizes more often, since this reduces the number of evaluated CUs. Since the complexity reduction is similar for all shifts, only the compression efficiency should be taken into account to determine the shifts that should be allowed when selecting personalized views.

4.4.3 Discussion

Although the method proposed in this section allows accelerated encoding of personalized views by skipping CU decisions based on a predicted CU structure, the method has several drawbacks. First, the machine learning algorithm requires the use of training frames. If the content of the video changes greatly, this implies that the model needs to be retrained, meaning that 10 new frames of the personalized views need to be fully encoded. However, in the case of many users, this implies a large spike in computational complexity, which might not be desirable. In order to avoid these spikes caused by retraining, an offline-trained model might be used instead. However, even if a model for each possible combination of shifts and zoom levels is implemented in memory, the usage of these models would likely have a negative effect on the compression efficiency, as has already been demonstrated in the case of spatial transcoding where content-dependent online models outperform offline-trained models [15].

Second, by only predicting CU information, the complexity reduction is still limited. Similar to Chapter 3, the model could be extended to include PU information, but by creating a more complicated model, the computational complexity during retraining would also increase. Therefore, an alternative solution might be to directly copy coding information from the full panoramic video by always forcing alignment of the personalized view with the CTU-grid[2]. As illustrated in

[2]Note that in the previous chapter, alignment can also be forced. However, this would result in restricting users to a fixed grid of possible positions for the videos. Since current commercial products for control rooms generally allow complete freedom of placement, this would thus lead to a loss of a currently existing feature, which may not be welcomed by all customers.

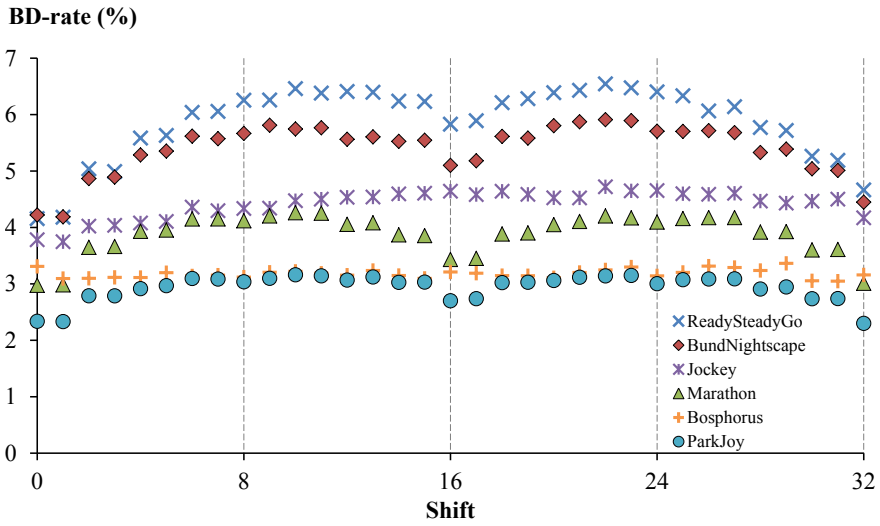


Figure 4.10: Effect on BD-rate when panning the virtual camera.

	Time Saving (%) (all shifts)	
Sequence	Average	Standard deviation
Bosphorus	79.0	0.2
BundNightscape	79.0	0.1
Jockey	77.4	0.1
Marathon	70.0	0.2
ParkJoy	71.2	0.3
ReadySteadyGo	76.1	0.2

Table 4.4: Time saving when panning the virtual camera.

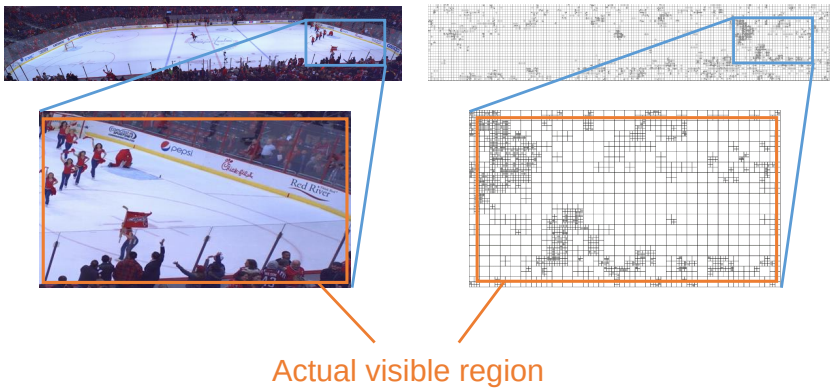


Figure 4.11: When copying coding information for a personalized view, alignment of the coding information with the CTU-grid can be forced by encoding a slightly larger view than the one that is actually shown to the viewer.

Figure 4.11, enforcing this alignment is possible by encoding extra pixels that are transmitted, but are not displayed on the user-side. In order to support zooming capabilities, the full panoramic video should also be encoded at different resolutions. If necessary, these different versions of the full panoramic video can also be encoded in an accelerated way by using information from the original in the same way as in Section 4.4.2.1. This alternative solution with a greater potential for complexity discussion is further investigated in the next section.

4.5 Further encoding complexity reduction

In the previous section the encoding complexity of the individual encodes was lowered by reusing coding information of an encoded full ultra-high-resolution video in order to speed up the encoding process of the RoI of each user. However, only CUs were reused from the ultra-high-resolution video, resulting in limited acceleration. As a result, the remaining complexity of each individual encode is still significant.

In order to further accelerate the individual encodes in the personalized-view approach, it is necessary to extract prediction mode, PU information, motion vectors, and merge information from the full encode of the ultra-high-resolution video and use this information in addition to CU information to speed up the encoding of the individual personalized views. As noted in Section 4.4.3, this is only possible after forcing alignment of the personalized view with the CTU-grid of the full panoramic video.

By feeding the encoder with more coding information such as PU information and motion vectors, more coding steps can be skipped. Consequently, copying

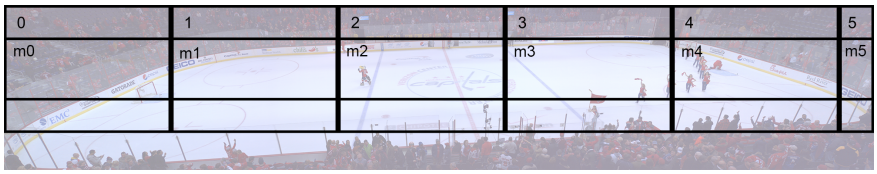


Figure 4.12: Selected 1088p RoIs for hockey1_1. The RoIs are marked with their corresponding notation on the figure. The middle views are indicated by their prefix m.

more coding information of the panoramic video lowers the coding complexity of the personalized views and thus speeds up the encoding process. However, this will also lead to less optimal coding decisions since the cropped view lacks the surrounding pixels, which impacts intra- and inter-prediction at the borders of the view.

4.5.1 Used content

Contrary to the previous section, where 4K-videos were used due to their greater abundance, a more realistic scenario with true panoramic content is considered in the rest of this chapter. The tested content consists of a hockey sports game, because this type of content has static areas such as the ice hockey field, moving areas such as the audience, and fast moving parts such as the hockey players. It is important to have a large range of spatial and temporal variability in the scenes, because this influences the complexity of the encoding. The hockey content consists of five sequences, split in two scenes. Only the first sequence of each scene was used for the evaluation, respectively named *hockey1_1* and *hockey2_1*. *Hockey1_1* contains a scene where cheerleaders enter the field, whereas *hockey2_1* is a scene during the match itself. The sequences have a duration of 10s each, at a rate of 60 frames per second. They have a resolution of 10.000×1880 pixels and have been 4:2:0 chroma subsampled.

From the two sequences, static views were chosen that contain different types of spatial and temporal activity. These views consist of regions that many users will choose to watch, such as the ice hockey field itself. These selected RoIs are shown in Figure 4.12. The top and middle views are indicated by their corresponding view numbers as shown in the figure. The middle views, which mostly show the ice hockey field, are specified by their prefix *m*. The views with view number five (*5* and *m5*) were ignored because these do not have the correct RoI resolution.

The RoIs each have a resolution of 1920×1088 pixels (1088p). The reason for the small deviation from 1080p is to have a multiple of the CTU size in HEVC in order to have perfect alignment with the CTU-grid of the full panoramic video. Only static views without zooming are considered in this section.

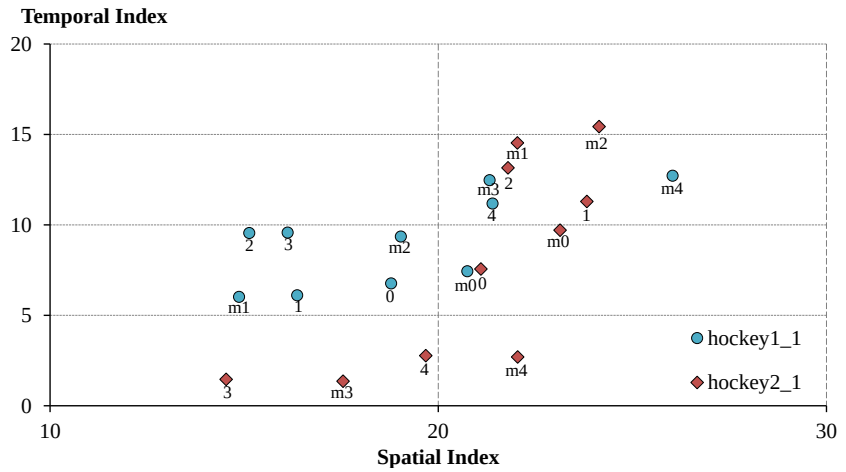


Figure 4.13: Spatial and temporal information for each view in sequence hockey1_1 and hockey2_1. The notes beneath the markers specify the particular view. The selected RoIs cover a variety of spatial and temporal activity.

In order to have a better indication on how much spatial and temporal information each view contains, the SI and TI values have been calculated for the different views. As seen in Figure 4.13, the RoIs have a large variety of TI/SI values, which corresponds with the assumption that views with different types of motion and spatial details are considered.

4.5.2 Evaluation

For the personalized-view method both the entire panoramic video and the RoIs were encoded. All encodings were done with version 16.5 of the HM software. Both the full panoramic video and all cropped views were encoded with four different QP values: 22, 27, 32 and 37.

All the views for both the non-accelerated reference encode and accelerated encodings were encoded with a *low-delay* configuration, meaning that the sequence is encoded with an I-frame, followed by all P-frames. This configuration was chosen since, contrary to the tiled-based approach which requires many random access points, each user has a personalized view and encoder instance. Therefore, the cropped region of the raw panoramic video (views) can be fed to one encoder instance for each user. It does not need I-frame refreshes because the user continues to use the same personalized stream. This configuration results in a lower delay, which is an important requirement for interacting with personalized views. For the tile-based method, a *random access* configuration would be needed because all the tiles can be retrieved by all users at any time and any position that corresponds with their selected RoI.

In order to evaluate the performance of the proposed personalized view method, bit rate overhead and encoder complexity reduction were evaluated. The bit rate overhead was again measured using the BD-rate, whereas the complexity reduction was calculated using the time saving metric.

Table 4.5 shows that by reusing only CU information the BD-rate is between 4.9% and 7.4% for the selected views with a time saving of around 79%. This time saving is comparable to the complexity reduction reported in Section 4.4. However, reusing CU info in combination with mode, PU and motion vector decisions increases this complexity reduction to more than 96.5%. In this case, the BD-rate is between 8.3% and 19.5%. How these values compare to a tile-based approach is further investigated in the next section.

Finally, copying merge (with skip) information results in irregular behavior for all views except for view *0*. Merge mode typically copies motion vectors from spatially or temporally adjacent blocks, as illustrated in Figure 4.14. In this figure, the blue block has a merge candidate list containing references to blocks D, B, C, A, and the temporal neighbor in this order. Note that this candidate list is not encoded into the bitstream, but is instead derived in both the encoder and decoder based on the availability of neighboring blocks. Only the merge index is actually stored in the bitstream as coding information. If this index is 0, this means that the motion vectors from block D are copied for the blue block. However, if the coding information of the entire panoramic video is used to encode a personalized view (the cropped region indicated by the orange rectangle in the figure), block D is no longer available, meaning that the copied merge index 0 will refer to the first available block, which is block B. Since motion vectors are then copied from the wrong block, the predicted block will differ a lot from the original picture. Moreover, in case of skip-mode when no residual picture is encoded, this predicted picture cannot be compensated by the residual. Due to this lack of residual, the block will resemble an incorrect area of the picture, eventually propagating throughout the video and leading to the afterimages shown in Figure 4.15. As a result, despite offering higher complexity reduction beyond 99%, merge information should not be copied in a personalized-view scenario.

4.6 Comparison with the tile-based method

The second main contribution of this chapter is to compare the tile-based method and the personalized-view method in terms of bit rate and PSNR for particular views. The bit rate should be low in order to make the interactive video system usable for clients with a limited bandwidth capacity. However, the PSNR should be high to have a good video quality for the RoI. In order to provide a fair comparison, several tile sizes of the tile-based approach were selected.

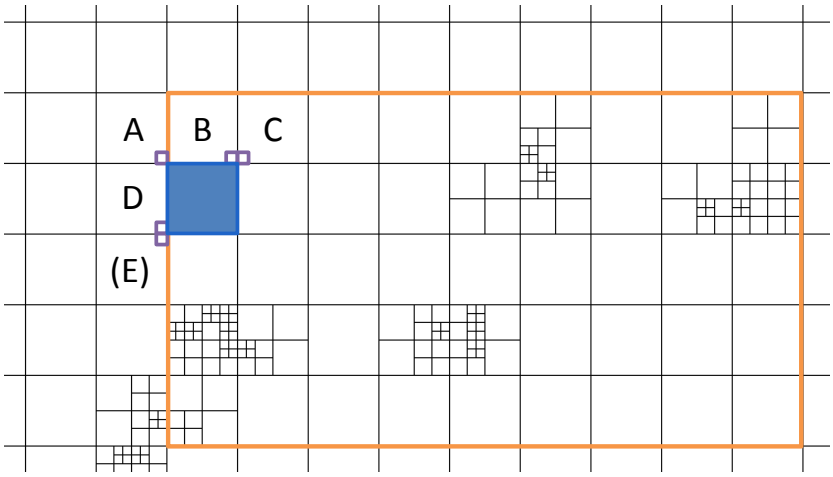


Figure 4.14: Under normal circumstances, if merge- or skip-mode is used for the blue block, it will have a merge candidate list consisting of [D,B,C,A,...]. If the merge index of this block is 0, the motion vector information of block D will be copied for this block. However, if the video is cropped (indicated by the orange rectangle), block A and D can no longer be a part of the merge candidate list and the merge candidate list will thus start with [B,C,...]. As a result, the blue block with merge index 0 thus mistakenly copy the motion vectors of B instead.

Figure 4.15: Illustration of afterimage-effect caused by copying unavailable blocks in the cropped views with merge/skip mode. Affected regions where afterimages are seen are marked with red rectangles.

		BD-rate (%)					Time saving (%)				
Sequence	View	CU	+ mode	+ PU	+ motion	+ merge	CU	+ mode	+ PU	+ motion	+ merge
hockey1_1	0	7.4	8.4	15.6	19.5	22.1	78.3	79.5	92.9	96.5	99.2
	m1	7.2	8.9	16.5	17.7	1164.1	80.7	81.0	93.1	97.3	99.5
	m4	6.1	8.0	15.0	16.2	278.6	78.1	79.3	92.6	96.6	99.2
hockey2_1	m1	5.5	7.3	13.2	13.7	862.9	79.2	79.7	92.6	96.8	99.3
	m3	4.9	6.0	9.5	8.3	-98.9	81.2	81.8	93.4	97.5	99.6

Table 4.5: BD-rates and Time Savings obtained by supplying different coding information. The columns incrementally represent the type of coding information that is reused from the panoramic video.

For the tile-based approach, the panoramic sequences were split into different tile sizes. The choice was made to pick 16:9 tile resolutions, because the RoIs are also close to 16:9 and this is a common aspect-ratio. The tested tile sizes were 1280×720, 1024×576, 640×360, 256×144 and 128×72 pixels. Next, these tiles were compressed using HM 16.5 as was done for the personalized-view method.

The tiles need to provide random access in order to allow changing of the RoI at any time as the tiles are pre-encoded on the server. Therefore, a *random access* configuration was chosen. This configuration consists of a structure of I-frames followed by B-frames that repeats every intra-period. This intra-period was selected as 32 frames, because this corresponds to a delay of 0.5s. Note that this is still a considerable amount of maximum delay when other tiles are selected, e.g. when another RoI is chosen. However, a smaller intra-period would result in higher bit rates due to worse compression of I-frames compared to B-frames. All tiles were encoded with the same four different QP values as in the previous section (22, 27, 32 and 37).

In the following subsections, the tile-based method and the personalized-view method are subsequently compared in terms of bit rate and PSNR, followed by a discussion.

4.6.1 Bit rate comparison

For the personalized-view method, the bit rates are retrieved from the encoding step with different coding information supplied to the encoder. For the tile-based method, the bit rates are calculated as the sum of the tiles of one particular tile size that (partially) overlap with the corresponding view. Figure 4.16 shows the bit rates of both methods and of the non-accelerated personalized-view encode for *hockey1_1* view *m4*. This view represents the plain white ice hockey field, the cheerleaders and the audience.

When both methods are compared, the bit rates of the personalized-view method are lower than the bit rates of the tile-based method. For example, in Figure 4.16, a bit rate of 6.83 Mbps is seen for QP 22 for which the cropped CU coding information of the panoramic video is reused, whereas 144p tiles have a bit rate of around 9 Mbps for QP 22. As seen for each QP, the bit rate of the personalized-view method remains below the bit rate of the tile-based method, even despite the overhead caused by accelerating the encoding of the personalized views. The reason for this difference is likely due to overhead caused by encoding a video with separate tiles. Because of the separate tiles, decisions such as motion estimation are constrained to the tile itself. As a result, if an object leaves the tile, it will be encoded less efficiently. Moreover, the tile boundaries will usually not match the boundaries of the selected view. As a result, extra pixels outside the view are also encoded, which further increases the bit rate overhead.

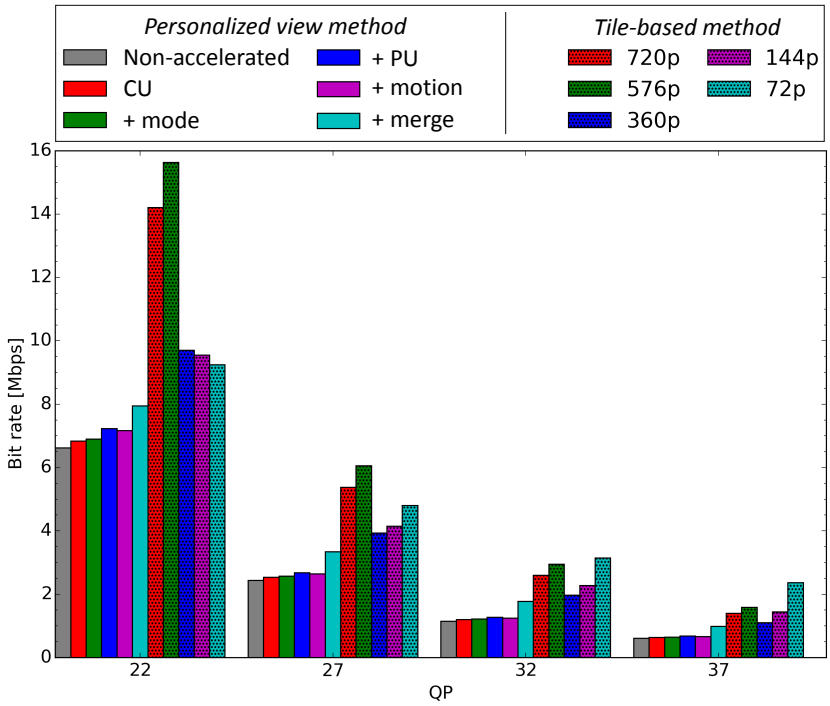


Figure 4.16: Comparison between the tile-based method with different tile sizes and the personalized-view method in terms of bit rate for hockey1_1 view m4 with the personalized-view method using a low-delay configuration.

In the above comparisons only static views are considered, but it is expected that the personalized-view method will further outperform the tile-based method in terms of bit rate when panning and tilting are taken into consideration. In such a scenario, all tiles covered by a dynamic view during the same intra-period need to be retrieved. If the user completely changes the RoI within this intra-period, the bit rate will at least double during this period since both the tiles corresponding to the old and new RoI will have been transmitted. Moreover, extra latency is introduced, since the current frame can only be decoded after all reference frames of the tiles corresponding to the new RoI have been retrieved.

4.6.2 PSNR comparison

Another important aspect to compare with is quality, which is measured in PSNR. The mean PSNR for the tile-based method was calculated by first transforming all the PSNR values back to the Mean Squared Error (MSE). Then, the average of all the MSE values, temporally and spatially corresponding to each tile size and to

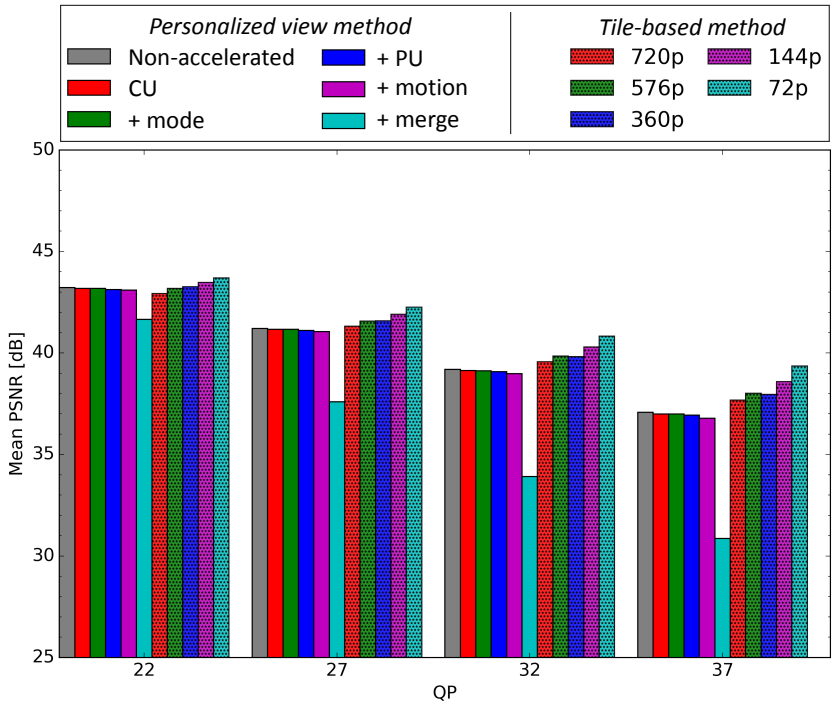


Figure 4.17: Comparison between the tile-based method and the personalized-view method in terms of PSNR for hockey1_1 view m4.

each QP that covers the view, was calculated and transformed back to PSNR. Since PSNR $\propto 10\log_{10}$(MSE), averaging MSEs instead of PSNR tends to penalize more if a single tile has a low PSNR. Consequently, minimizing such an average tends to enforce a more constant PSNR over the different tiles.

Figure 4.17 presents the PSNR of both methods for *hockey1_1* view *m4*. This figure shows that the tile-based method for all the tile sizes performs better in terms of PSNR than the personalized-view method. For QP 32, the PSNR is around 39 dB when the cropped CU, mode and PU coding information of the panoramic video is reused, whereas the 144p tiles have a PSNR of around 40 dB for QP 32. Similar behavior is seen for the other views. Note that for the personalized-view method the PSNR drops significantly when merge coding information is also reused from the panoramic video.

The reason of the increased PSNR might possibly be that the smaller tile sizes do not use skip-mode as often as the larger tile sizes, since they have less neighboring blocks to copy the motion information from. Instead, they are more likely to perform motion estimation to calculate new motion vectors, and might also split the CUs into PUs for which a better match can be found than for the complete CU

Figure 4.18: Illustration of tiling artefacts at QP 37. The tile borders are indicated by the white ticks.

with skip-mode. Moreover, contrary to skip-mode, a residual picture is calculated for the resulting block, which compensates for errors between the original and the motion-compensated picture and thus results in a larger PSNR value.

However, note that, although the PSNR of the tile-based method appears to be higher, inter-tile artefacts are visible at higher QP-values for the tile-based method as a blocking effect at the border of each tile (illustrated in Figure 4.18). These are not taken into account with the PSNR metric, despite lowering the subjective visual quality.

4.6.3 Discussion

In previous subsections, bit rate and PSNR were considered separately. However, in Figure 4.19, the optimal tile size (144p) and fastest personalized-view method (accelerating both CU, PU, mode, and motion decisions) are compared. Additionally, the non-accelerated personalized-view approach is shown as well.

The figure shows that the accelerated personalized-view method performs better compared to the tile-based approach in terms of compression efficiency in the lower bit rate range (under 3 Mbps for *hockey1_1* view *m4*). However, in the range of higher bit rates, both perform similar. Therefore, for static views as tested in this chapter, the tile-based method might be a better approach in terms of scalability for many users, since the tiles only need to be encoded once. Nevertheless, the non-accelerated personalized-view method is still more compression-efficient, since it does not suffer from the overhead introduced by the guided encoding, and does not have any tiling overhead either. As such, for a small amount of users,

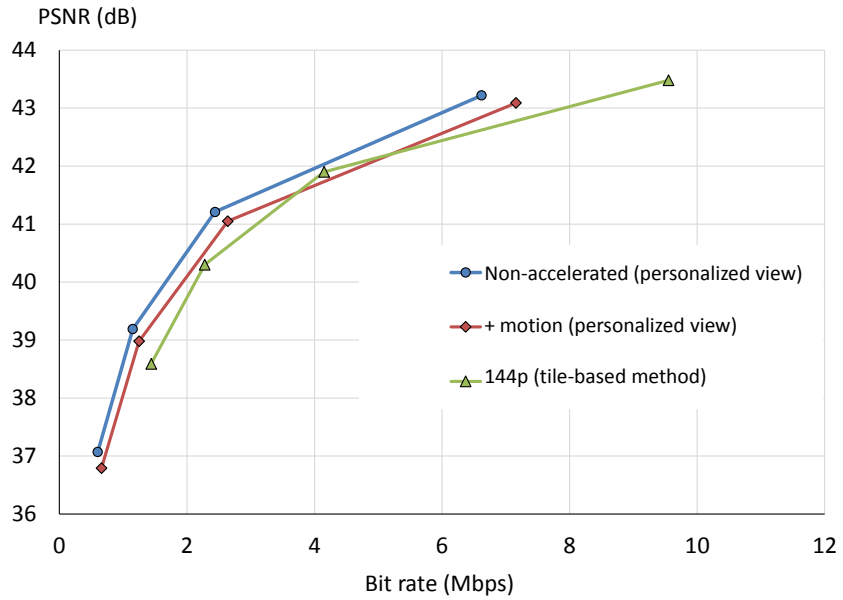


Figure 4.19: Comparison in terms of both bit rate and PSNR between the personalized-view approach and the tile-based method for hockey1_1 view m4.

and if computational power is not much of a concern, the non-accelerated personalized views or the personalized-view method that only accelerates CU and/or PU and mode decisions (and thus has a smaller BD-rate overhead compared to also accelerating motion) might be better choices.

Note that, as was mentioned in Section 4.6.1, the personalized-view method would not suffer from the structural latency or the extra bit rate overhead when considering dynamic views. Therefore, it is likely that even accelerations up to motion information will outperform the tile-based approach when considering movement of the view. However, in order to test this, motion vectors can no longer be simply copied from the full panoramic video, and should thus be modified to compensate for the movement of the view. Moreover, special care might need to be taken for motion vectors that point to non-existent blocks due to the movement. Furthermore, in order to have a fair comparison with the tile-based approach, different intra-periods besides the 0.5s as tested for static views should be tested as well, since this affects both the structural latency and bit rate overhead of the tile-based method. Finally, if the coding decisions of the guided encoding can be further refined, and in particular if merge decisions can be handled better by for example using different strategies for blocks at the boundaries of the cropped views, the performance of the accelerated personalized-view method might still increase. However, these further investigations are left as future work.

4.7 Conclusion

In this chapter, a fast personalized-view method for delivery of interactive views from immersive video content was proposed. An initial method that accelerated the encoding of the personalized views by using CU structures predicted from the entire panoramic video was applicable for any location of the RoI. However, this method was limited in complexity reduction and may require a computationally expensive retraining of the machine learning algorithm during encoding. Therefore, a second method consisting of directly copying coding information from the entire panoramic video was also investigated by forcing the RoI to be aligned with the CTU-grid of the entire video.

The second method was then compared to the tile-based method. This comparison showed that reusing coding information obtained from a panoramic video to accelerate the encoding of each personalized view with 95-98% results in a bit rate overhead of 8-20%, which is still smaller in terms of bit rate overhead compared to the tile-based method. Moreover, since personalized, guided encoders do not suffer from the same structural latency as the tile-based method if the system were to be extended to dynamic views, such encoders could especially become a viable alternative for the tile-based approach in low-latency scenarios.

References

[1] A. Mavlankar and B. Girod. *Spatial-Random-Access-Enabled Video Coding for Interactive Virtual Pan/Tilt/Zoom Functionality*. IEEE Trans. Circuits Syst. Video Technol., 21(5):577–588, May 2011.

[2] N. Quang Minh Khiem, G. Ravindra, and W. T. Ooi. *Adaptive encoding of zoomable video streams based on user access pattern*. Signal Processing: Image Communication, 27(4):360–377, 2012.

[3] V. Reddy Gaddam, H. B. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P. Halvorsen. *Tiling of panorama video for interactive virtual cameras: Overheads and potential bandwidth requirement reduction*. In Picture Coding Symposium (PCS), pages 204–209, May 2015.

[4] Y. Umezaki and S. Goto. *Image segmentation approach for realizing zoomable streaming HEVC video*. In Int. Conf. Inf. Commun. Signal Process. (ICICS), pages 1–4, Dec. 2013.

[5] P. R. Alface, J. F. Macq, and N. Verzijp. *Interactive omnidirectional video delivery: A bandwidth-effective approach*. Bell Labs Technical Journal, 16(4):135–147, Mar. 2012.

[6] N. Quang Minh Khiem, G. Ravindra, A. Carlier, and W. T. Ooi. *Supporting Zoomable Video Streams with Dynamic Region-of-interest Cropping*. In Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, MMSys '10, pages 259–270, 2010.

[7] C. Fehn, C. Weissig, I. Feldmann, M. Muller, P. Eisert, P. Kauff, and H. Bloss. *Creation of High-Resolution Video Panoramas of Sport Events*. In Proc. IEEE Int. Symposium Multimedia (ISM), pages 291–298, Dec. 2006.

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

[9] K. McCann, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description*. Technical Report JCTVC-S1002, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2014.

[10] L. Song, X. Tang, W. Zhang, X. Yang, and P. Xia. *The SJTU 4K video sequence dataset*. In Proc. IEEE Int. Workshop Quality Multimedia Experience (QoMEX), pages 34–35, July 2013.

[11] ITU-T. *Subjective video quality assessment methods for multimedia applications*. Technical Report Rec. P.910, Apr. 2008.

[12] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

[13] I.-K. Kim, W.-J. Han, J. H. Park, and X. Zheng. *CE2: Test results of asymmetric motion partition (AMP)*. Technical Report JCTVC-F379, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), July 2011.

[14] B. Li and J. Xu. *On motion estimation start point*. Technical Report JCTVC-R0105, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), July 2014.

[15] L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Performance analysis of machine learning for arbitrary downsizing of pre-encoded HEVC video*. IEEE Trans. Consum. Electron., 61(4):507–515, Nov. 2015.

5

Guided Encoding of Personalized Dynamic-Range Video

5.1 Introduction

A third type of personalization in this dissertation consists of adapting the dynamic range of the bitstream to the capabilities of the receiving device. HDR technology is an emerging technology that consists of the capturing and displaying of a higher range of luminance than before [1]. For example, an HDR-display will be able to achieve a brightness of more than 4000 nits (candela per square meter), whereas a traditional high-definition television only reaches a brightness of 400 nits. This higher range of brightness improves the potential contrast of images and videos in such a way that the image corresponds better to what would be seen through the human eye instead of a camera.

Due to limitations in display technology and because the reference luminance value of HDR has not been decided yet, television manufacturers are currently producing televisions with an output luminance between 400 and 1000 nits. However, as technology continues to evolve, this luminance will increase further. As a result, consumers will be left with a wide array of televisions with different dynamic range capabilities. This will pose a challenge to the content providers, since displaying an HDR video on a display with a lower dynamic range requires the use of tone mapping algorithms [2–6]. These algorithms map an HDR video to a low-dynamic-range (LDR). However, if the tone mapping is done by the consumer television, each television might implement a different algorithm. As a result, al-

most no consumers will view the video in the way as the content provider intended, which can negatively influence the reputation of the provider if the HDR video is not optimally mapped to LDR.

A better solution for the content provider is to create multiple versions of the HDR video with different (lower) dynamic ranges. In this way, the provider can ensure that each consumer will receive a video that the target device is capable of displaying without additional tone mapping. However, this also means that the content provider needs to perform video compression, a computationally complex operation, on multiple versions of the same video. Since encoding of HDR content is expected to be done with the HEVC standard, which has a much higher computational complexity compared to older standards, the problem of the high computational encoding cost will be even greater.

To solve this problem, this chapter proposes to reuse coding information from the HDR video to guide the encoding of the LDR versions. In order to achieve this, the correlation between coding information of HDR and LDR videos encoded at different qualities is first examined. This correlation is then used to create a model that determines the LDR video correlating the most with the HDR video from which coding information is reused.

As its main contribution to the state-of-the-art, this chapter allows simultaneous encoding of multiple dynamic range versions of a video for approximately the same computational cost as a single encoder. Additionally, it also measures the trade-off between compression efficiency and computational complexity by varying the amount of coding information of the HDR video that is reused for encoding the LDR versions of the same video. This allows content providers to make a decision between having either a higher bit rate overhead or a higher computational complexity.

The rest of this chapter is organized as follows. Section 5.2 gives a short overview of related work on HEVC encoder acceleration and simultaneous encoders. In section 5.3, the analysis of the HDR and LDR video and the creation of the model are presented. Section 5.4 then evaluates the performance of the model and the use of different types of coding information. Finally, the conclusion is found in section 5.5.

5.2 Related work

As was also mentioned in section 2.2 , since all coding information needs to be evaluated for each CU, most work that accelerates HEVC encoders focuses on early termination of CU splitting or determining the optimal CU structure of the compressed video [7–13]. Other research also accelerates the encoder by limiting the PU partitioning modes [14] and TU splits [15], sometimes in combination with early termination of CU splitting [16, 17]. Finally, some works also focus on

accelerating intra mode evaluations in intra frames [18, 19], or fast skip mode decisions [20]. However, in most of these works, keeping the compression efficiency as high as possible is favored over great reductions in encoding complexity.

The above works all focus on reducing the complexity of an encoder that encodes a single sequence. However, recent research has begun to focus on multi-rate or simultaneous encoders where coding information of one version of a video is used to speed up encoding of other versions. In the past, multi-rate encoding for older compressions standards has been considered for VP8 by reusing motion information between versions encoded at different bit rates [21, 22]. More recently, a simultaneous encoder of H.264/AVC and HEVC reuses motion vectors [23] or uses block structures from H.264/AVC to predict CU and PU decisions in order to accelerate the encoding of an HEVC version [24].

Simultaneous encoding of different versions that are encoded with HEVC have been considered as well. These focus on different video representations for adaptive streaming by either considering different versions at the same resolution [25], versions with different spatial resolutions [26], or a combination of both [27]. However, all these methods focus on accelerating CU decisions, which limits the complexity reduction they attain. As a result, the multiple versions that are generated still require a sizeable amount of processing complexity.

Contrary to the aforementioned papers, this chapter reduces the complexity as much as possible in order to have a simultaneous encoder that can generate multiple versions with negligible computational cost compared to a single encode.

5.3 Proposed method

This section first describes the simultaneous encoder architecture. Next, it analyzes the correlation between the coding information of HDR and LDR versions of the same video. Based on this information, a model is built to estimate the version which has the highest correlation.

5.3.1 Simultaneous encoder architecture

The proposed simultaneous encoder architecture is shown in Figure 5.1. Several input videos are fed to the simultaneous encoder which consists of several sub-encoders. The HDR encode is the video with the highest dynamic range of the supplied versions. Videos with lower dynamic ranges are denoted as LDR.

The HDR version is fully encoded. As such, the optimal combination of coding information such as CU, PU, and TU structures, as well as inter- and/or intra-prediction information is determined for each CTU. Whenever the information for a complete CTU is determined, (part of) this information is supplied to the LDR encoders. These encoders proceed to encode the CTU at the same location in

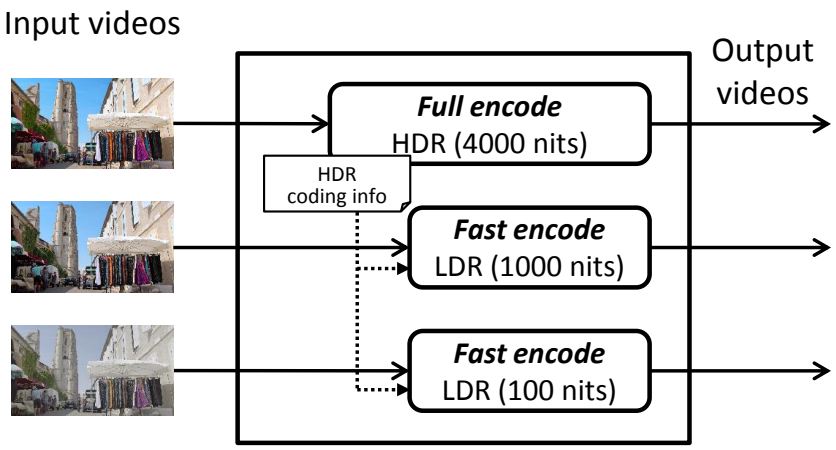


Figure 5.1: Proposed simultaneous encoder architecture. The HDR version of the video is fully encoded, whereas the encoding of the other versions is accelerated by using information from the full encode.

these different versions. The encoding of these versions is accelerated by using the supplied coding information of the HDR version.

Since the coding information from the HDR encoding is directly passed to the LDR encoders with a delay of at most one CTU, this means that the encoding of the different versions happens at the same time. As a result, no extra memory is required to store the HDR coding information. The memory requirements of the proposed simultaneous encoder are therefore the same as when running a separate encoder for each version on the same machine.

Note that the simultaneous encoder could also use an LDR encoder at a lower dynamic range to speed up the encoding of the higher dynamic ranges. However, since speeding up encoders usually results in a decrease of quality for the same bit rate, and since content providers are likely to prefer having the best possible quality for the HDR sequence, the HDR sequence is always fully encoded.

5.3.2 Analysis of HDR and LDR coding information

In order to determine the feasibility of using HDR coding information to accelerate the encoding of an LDR version of the same content, the correlation between coding information of HDR and LDR content was investigated. This analysis was performed on three video sequences provided by MPEG [28]. Only three sequences had both HDR (4000 nits) and LDR (100 nits) versions available at the time of writing. The first frame of each sequence is displayed in Figure 5.2. These sequences are the following:

Figure 5.2: First frame of each video sequence used in this chapter.

Sequence	Number of frames	Frame rate (fps)	Intra-period
Market3Clip4000r2	400	50	48
FireEater2Clip4000r1	200	25	24
BalloonFestival	240	24	24

Table 5.1: Overview of sequences and parameters

- *Market (Market3Clip4000r2)* consists of a street market with some pedestrians in the left corner of the image and clothes waving in the wind in the right corner while the camera slowly pans from left to right.
- *Fire (FireEater2Clip4000r1)* consists of a night scene with two fire jugglers and a fire breather performing a show.
- *Balloon (BalloonFestival)* consists of a scene with many people walking in the bottom part and with a skyline and balloons in the upper part.

For each of the three video sequences, both an automatically and manually color-graded version were available. Automatic color-grading means that the HDR version has been converted to an LDR version by using a tone mapping algorithm as mentioned in section 5.1. With manual color-grading on the other hand, the colors of the video are manually tuned by a professional grader to achieve the best visual quality. However, since manually color-grading a large amount of different LDR versions is costly, content providers may prefer to select an automatic color-grading algorithm to generate the LDR versions of the video. In order to simulate this scenario, this chapter will primarily focus on the automatically color-graded videos.

For each sequence, the HDR version and its corresponding automatically color-graded LDR version were encoded with HM version 16.5 [29] with QP values ranging from 18 to 42. As an encoder configuration, the 10-bit random access configuration was used with an intra-period of approximately 1 second, which is

common in adaptive streaming scenarios used by content providers. An overview of the encoding parameters is given in Table 5.1.

Since both an LDR and HDR version are still based on the same video content, there might be some correlation between coding decisions made when encoding both versions. However, since the values of the pixels between the two versions are different (e.g. with clipping of pixel values occurring in very dark or very bright regions in the LDR version), it is unlikely that the highest correlation is found when using the same coding parameters for both. Therefore, the correlation between the HDR and LDR versions of the video is investigated for different QP values.

CU information is used to determine this correlation between the different HDR and LDR versions of each sequence, since all other coding information depends on CU structures as explained in section 2.2. To determine the correlation, these CU structures can be interpreted as hierarchical data. For example, CUs of depth 0 and 1 can be considered to be more correlated compared to depth 0 and 3. As such, the Pearson product-moment correlation coefficient can be used to describe the CU correlation between two frames [30]. This coefficient is called the r-score and has a range from -1 to 1. For the positive values of an r-score, high values indicate high correlation, whereas zero implies no correlation at all.

Each frame is split up into blocks of 8×8 pixels, which equals the smallest CU size. Next, each of these blocks is assigned the depth of the CU to which it belongs. All depths of these 8×8 blocks are then added to a vector for both the HDR and LDR frame. The r-score of a frame j is then calculated as equation (5.1), with x, y, and n respectively being the LDR vector, HDR vector, and number of 8×8 blocks in a frame.

$$r_j = r_{xy,j} = \frac{\sum_{i=1}^{n}(x_{i,j} - \overline{x}_j)(y_{i,j} - \overline{y}_j)}{\sqrt{\sum_{i=1}^{n}(x_{i,j} - \overline{x}_j)^2 \sum_{i=1}^{n}(y_{i,j} - \overline{y}_j)^2}} \tag{5.1}$$

Next, the r-coefficients of each frame are combined to obtain the correlation over the entire video. Since using a normal arithmetic mean to average the correlation coefficients skews the result, a Fisher z-transform is used [31]. The CU correlation is thus calculated as equation (5.2), where m is the number of frames in the video.

$$r_{avg} = \tanh\left(\frac{\sum_{j=1}^{m} \text{arctanh}(r_j))}{m}\right) \tag{5.2}$$

An example of CU correlations is given in Table 5.2 for the *balloon* video sequence. The correlations are presented in percentage, with each column representing a QP of the LDR version (QP_{LDR}) and each row representing a QP of the HDR version (QP_{HDR}). Note that some QP values are omitted for clarity. When

$QP_{LDR} \rightarrow$ $QP_{HDR} \downarrow$	22	24	26	28	30	32	34	36	38	40
40	57	60	63	67	71	76	80	84	89	93
38	61	65	68	71	75	79	84	87	92	95
36	65	69	72	75	79	83	87	91	94	93
34	69	73	76	79	83	87	91	93	92	90
32	73	76	80	83	86	90	93	92	89	86
30	77	81	84	87	91	93	91	88	85	81
28	82	85	88	91	93	91	88	84	80	77
26	85	89	92	93	91	88	84	80	77	73
24	89	92	93	92	88	85	81	77	74	70
22	92	93	92	89	86	82	78	74	71	67
20	93	92	89	86	83	79	75	71	67	64
18	92	90	87	84	80	76	72	69	65	62

Table 5.2: Correlations (%) between an LDR and HDR version of the same video for the balloon sequence. A lighter color indicates a higher correlation.

looking at QP_{LDR} = 22, the QP_{HDR} with the highest correlation is 20, with a correlation of 93%. For each QP_{LDR}, such a QP_{HDR} with a correlation of 93% or more can be found by following a diagonal in the table. This indicates that there might be a rule to determine the most optimal QP pair of QP_{LDR} and QP_{HDR}.

In order to better determine the relationship between QP_{LDR} and QP_{HDR}, the most correlating QP_{HDR} is determined for each QP_{LDR} for the three video sequences, as shown in Figure 5.3. Based on this information, a prediction model is created by using Ridge regression with generalized cross-validation [32]. This form of regression has an adjustable parameter which is determined through the generalized cross-validation and controls the norm of the weight in the error term to control the extent of the inherent overfitting due to using a small data set. In order to obtain the model, the problem in equation (5.3) is thus solved, with $\mathbf{X}$ containing zeroes in its first column and the vector of QPs of the LDR version in its second column, $\mathbf{y}$ being the vector of the matching most correlating QP_{HDR}, and $\mathbf{w}$ containing the intercept (w_0) and coefficient (w_1) of the model to be determined.

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \alpha \|\mathbf{w}\|_2^2 \tag{5.3}$$

The resulting model estimated from the three sequences is

$$y = 0.976422x - 0.959334 \tag{5.4}$$

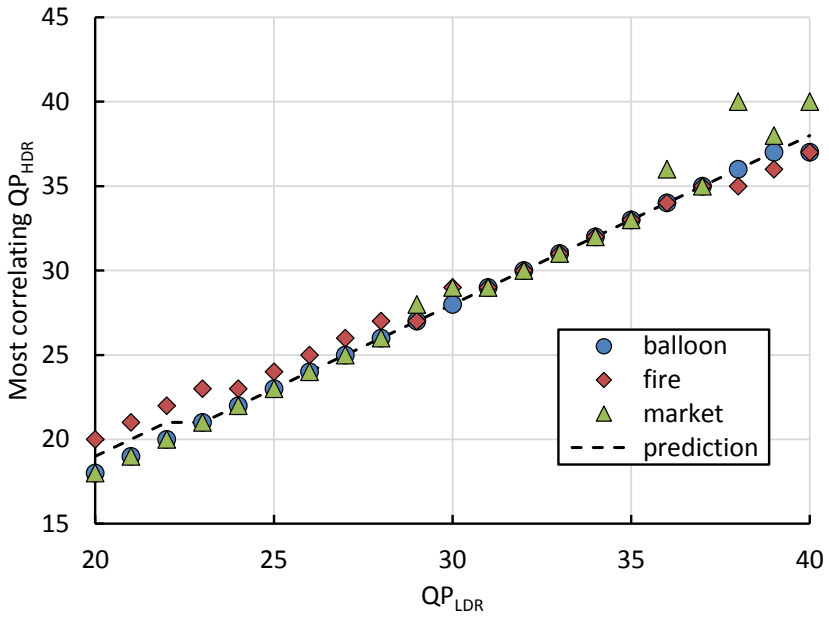


Figure 5.3: Most correlating QP of the HDR version for each QP_{LDR}. Based on the data from the three sequences, a prediction model is estimated.

When taking into account that QP values are non-continuous, the final model for determining the most correlating QP pair becomes equation (5.5). This prediction model is also shown in Figure 5.3.

$$QP_{HDR} = \lfloor QP_{LDR} \times 0.976422 - 0.459334 \rfloor \tag{5.5}$$

Due to the limited amount of video content that is currently available in both HDR and LDR versions, the model cannot be verified on a separate validation set. However, to investigate the behavior of estimating the model from a different set, leave-one-out cross-validation was used. Consequently, three additional models were created, each estimated from only two of the three available video sequences. When using these models to predict the most correlating QP pair for each of the three sequences, the predicted QP differs only by 2 at most. When looking at the highest correlated QP_{HDR} for each QP_{LDR}, some other QPs have a correlation close to the QP with the highest correlation. This is visualized in Figure 5.4 for the *balloon* sequence.

In this figure, the error bars represent a buffer margin of 2.5% around the most correlating QP_{HDR}. For example, if the most correlating QP_{HDR} has a correlation of 97% with the LDR version, the error bars cover all QPs that have a correlation of 94.5% or more. All QP values predicted by the three additional models

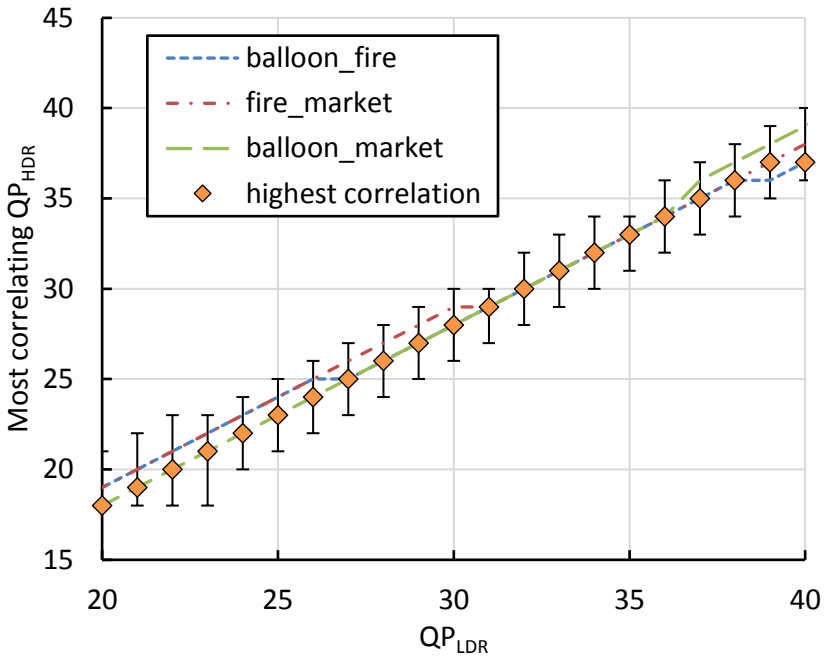


Figure 5.4: Illustration of the 2.5% buffer margin around the most correlating QP_{HDR} for each QP_{LDR} for the balloon *sequence. Models estimated from only two of the sequences always predict a most correlating QP within this buffer margin.*

fall within this range. As such, even when the model is only estimated from a subset of the sequences, the resulting model still results in a good prediction. In the remainder of this chapter, the model estimated from all three sequences is used.

5.4 Evaluation

In this section, the performance of the algorithm is evaluated when accelerating different types of encoding decisions. Next, the robustness of the model is analyzed, while also verifying the earlier assumption that a model based on the CU correlation between two sequences also performs well when copying other types of information. Finally, the proposed encoder acceleration is compared to related work.

5.4.1 Effect of coding information

To evaluate the proposed model, LDR versions of the three video sequences were encoded by copying information from the most correlating HDR version based

Name	Coding information
CU	CU split flag
+ mode + PU	Prediction mode (inter/intra)
	PU partitioning mode
+ merge	Merge index
	Merge flag
	Split flag
+ motion	Motion vector predictor index
	Motion vector difference
	Reference picture index
	Type of inter-prediction (uni-/bi-directional)
+ intra direction	Luma intra-prediction direction
	Chrome intra-prediction direction
+ TU	TU split flag

Table 5.3: Full overview of reused coding information

on equation (5.5). The same configuration as in section 5.3 was used for the encoder. The LDR versions were incrementally accelerated by copying CU information, mode and PU information, merge information, motion information, intra-prediction direction, and TU information. A full overview of this information is given in Table 5.3. For example, in the rest of this chapter, '+ merge' will always indicate that the CU split flag, prediction mode, PU partitioning mode, merge index, merge flag, and split flag from the HDR version have been reused to encode the LDR version of the video.

The performance of the algorithm was measured in terms of complexity reduction, and in terms of coding efficiency. The complexity reduction is expressed as a speed-up factor defined as equation (2.2). Additionally, the time saving in equation (2.1) is also calculated in order to allow better comparison with related work.

Coding efficiency is measured using BD-rate, which measures the average bit rate increase for the same quality compared to the non-accelerated encode [33]. Similar to previous chapters, the quality is measured in PSNR.

Figure 5.5 shows RD-curves for the LDR version of the *market* sequence. The other sequences behave in a similar way. These RD-curves show that copying coding information from the most correlated HDR version (as predicted by equation (5.5)) to accelerate an LDR version of the same video, decreases the coding efficiency of the LDR version only slightly, except when copying merge information

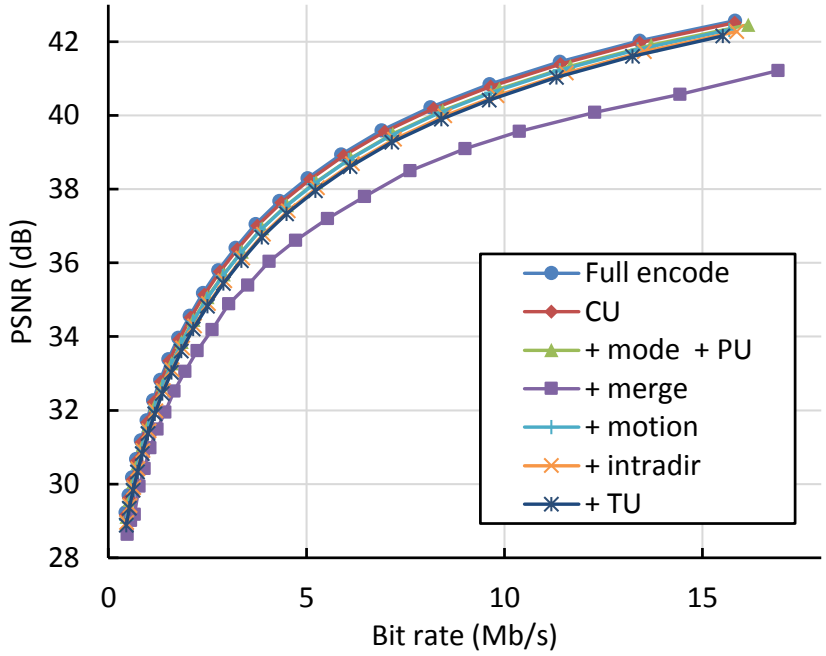


Figure 5.5: RD-curves for the LDR version of market. *Except when copying merge information from the HDR sequence without also copying motion information, the coding efficiency decreases only slightly compared to a full encode.*

without also copying motion information. This is because merge information and motion information are closely related. When a PU uses merge mode, it will copy the motion vectors from an adjacent block according to the merge index. However, when motion information is still recalculated and not copied from the HDR version, the motion vectors referred to by the merge index in the LDR version might be different from the original HDR version. As a result, wrong motion vectors will be used, and this will cause either a spike in residual information in case of merge, or the copying of a wrong block in case of skip. Furthermore, these errors will propagate as more blocks use merge mode.

A full overview of the results is given in Table 5.4. Each column shows the information that was copied from the HDR version. In each column, more information is copied. When copying only CU information, the bit rate increase is only 2.9% on average. However, when mode and PU information is copied, the BD-rate increases to 7.7% on average, since the encoder has less freedom to correct wrong CU structures by compensating with a different mode or PU structure. As was also seen in Figure 5.5, copying merge information then results in a spike in BD-rate of 38.6%. However, when also copying motion information, these values lower to 7.5%, which is a similar value as when copying CU, mode, and PU informa-

	Video	CU	+ mode + PU	+ merge	+ motion	+ intra direction	+ TU
BD-rate (%)	Market	2.4	7.4	41.6	8.0	12.8	13.7
	Fire	3.9	9.0	44.4	8.0	11.5	12.6
	Balloon	2.2	6.6	29.7	6.6	10.1	11.0
	Average	2.9	7.7	38.6	7.5	11.5	12.4
Time saving (%)	Market	74.0	95.7	99.0	99.5	99.6	99.7
	Fire	73.8	95.8	99.2	99.5	99.6	99.7
	Balloon	74.0	95.6	99.1	99.4	99.6	99.6
	Average	74.0	95.7	99.1	99.4	99.6	99.7
Speed-up factor	Market	3.8	23.4	98.5	182.8	235.5	290.4
	Fire	3.8	23.8	132.9	185.3	266.9	328.2
	Balloon	3.8	22.5	110.8	167.6	230.8	278.4
	Average	3.8	23.2	114.1	178.5	244.4	299.0
Speed-up gain	Average	3.8	6.0	4.9	1.6	1.4	1.2

Table 5.4: Results for different amounts of coding information used

tion. Copying the intra direction has a negative effect, increasing the BD-rate to 11.5%. This appears to indicate that copying wrong decisions to intra-frames has a strong negative effect on the following inter-frames because they are used as reference frames for motion estimation. Finally, when also adding TU information, the BD-rate becomes 12.4% on average.

In terms of speed-up, the encoder is already accelerated by a factor of 3.8 when copying CU information. However, when adding all information, the speed-up factor of the encoder is 299. In fact, since all coding information is fed to the encoder, the only remaining complexity is the calculation of the residual and entropy coding.

To show how much copying each type of information further accelerates the encoder compared to adding less information, the speed-up gain was also calculated. For example, copying mode and PU information further accelerates the encoder 6 times compared to copying only CU information. By using this metric, it becomes clear that copying mode and PU information on top of CU information results in the most gain. From merge information and beyond, this gain decreases. However, even when adding TU information, this gain is still 1.2, which is still significant since it means that the encoder was further sped up with 20%.

Finally, one might notice that the BD-rates when copying the intra direction and TU information are greater than 10%, which might be considered large. However, the disadvantages of these higher bit rates are compensated by achieving very high encoding speeds. As such, content providers can make a trade-off between compression efficiency and encoding speed depending on their requirements.

5.4.2 Robustness of the model

As mentioned in section 5.3, the correlation between two different dynamic range versions of the same video is determined based on the CU structure of the bitstreams due to the assumption that all other coding information is dependent on CU structures. In order to verify that a model based on CU correlations also results in the most efficient selection of a QP pair when all information is copied from the HDR version, the coding efficiency of using other HDR versions was determined by varying QP_{HDR}. In Figure 5.6, the curve '+ TU' shows the results that are obtained by using the model. The diamond-shaped points are the results obtained when encoding the version with a QP_{LDR} of 30 by copying information from different HDR versions. The most optimal points in terms of coding efficiency are obtained by copying information from QP_{HDR} 27 and 28. Since QP_{HDR} 28 is the value predicted by the model, the assumption made in section 5.3 appears to be correct. For other sequences and points, similar behavior is seen.

As another way to evaluate the robustness of the model, manually color-graded versions of the LDR version were used. A new model was estimated from these

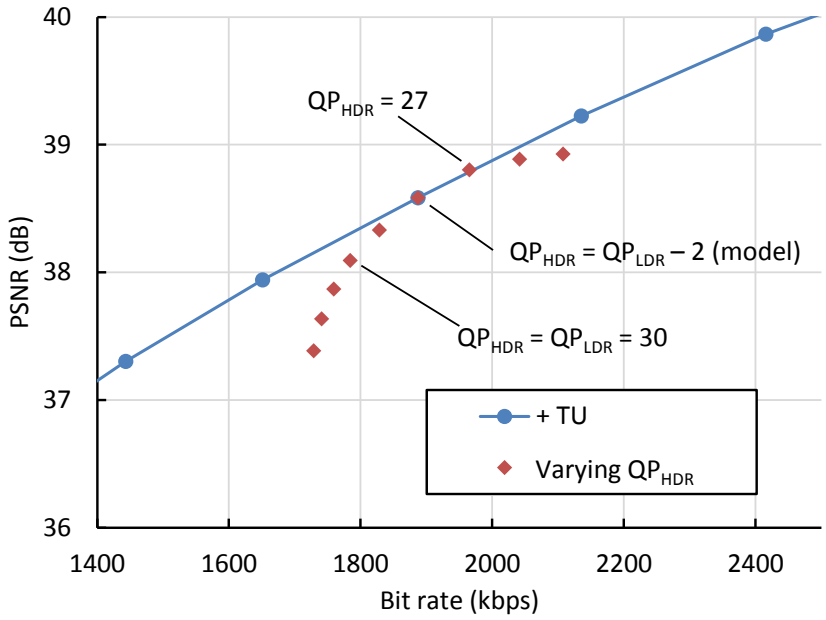


Figure 5.6: Coding efficiency when copying all coding information from an HDR version coded with varying QP_{HDR} to encode an LDR version, for the balloon *sequence. The points connected with the line show a part of the RD-curve created by encoding the LDR version by copying all coding information from the HDR version according to the model.*

Model	CU	+ motion	+ TU
Estimated from manual	3.6	11.4	18.5
Estimated from auto	3.6	11.3	18.4

Table 5.5: Average BD-rate (%) of manually color-graded sequences when using a model estimated from either the manually or automatically color-graded sequences

manually color-graded versions and the same evaluations were done as with the model for the automatic color-graded sequences. This model sometimes differed by one QP value compared to the model for automatic color-graded sequences due to the difference in color-grading techniques. Next, the latter model was used to determine the most correlating version of the manually color-graded versions. The results are shown in Table 5.5 for the cases of copying only CU information, all information up to motion, and all information. The results show that both models perform very similar, with only a difference of 0.1% BD-rate. This is due to the buffer margin of the most correlating QP_{HDR} as was also shown in Figure 5.4. As a result, the model is robust to small changes in the used color-grading algorithm.

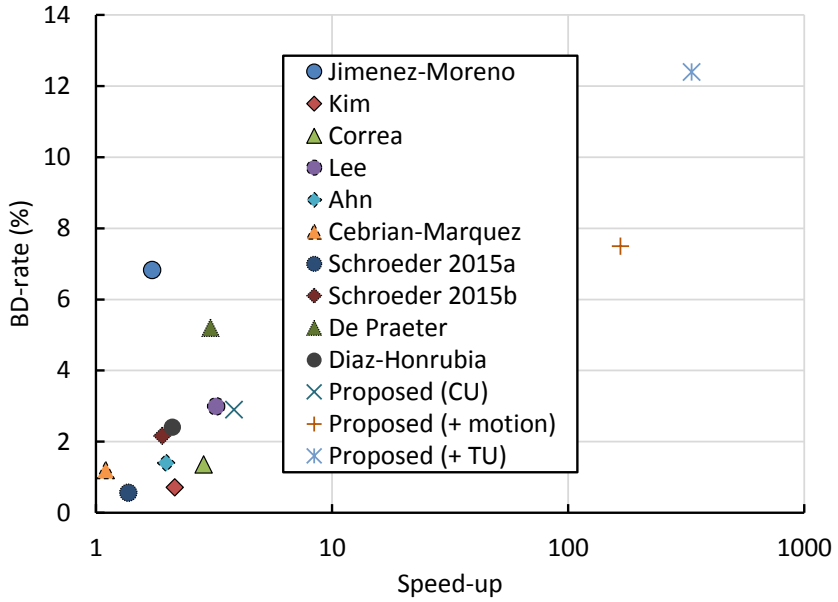


Figure 5.7: Comparison with state-of-the-art. The fastest version of the proposed algorithm achieves speed-ups that are more than 90 times greater than related work.

5.4.3 Comparison with related work

In order to compare the proposed simultaneous encoder to the state-of-the-art, it was compared to several fast-encoding and simultaneous-encoding algorithms from Jimenez-Moreno [12], Kim [13], Correa [17], Lee [8], Ahn [9], Cebrian-Marquez [23], Schroeder [25, 26], De Praeter [27], and Diaz-Honrubia [24]. The most relevant results from these works were selected. Although these works used different sequences for evaluation, it becomes clear that the proposed method achieves a far greater speed-up compared to the state-of-the-art. This is seen by comparing the fastest algorithm of the related work ('Lee' in Figure 5.7), which has a speed-up of 3.23, to the proposed method when copying all information ('Proposed (+ TU)' in Figure 5.7), which has a speed-up of 299. In this example, the proposed method is more than 90 times faster compared to the fastest related work with a BD-rate that is only 4.2 times the BD-rate of Lee's algorithm.

The reason for these high speed-ups is that the proposed methods, in particular 'Proposed (+ motion)' and 'Proposed (+ TU)', accelerate more coding decisions than the state-of-the-art fast-encoding algorithms. Even 'Proposed (+ CU)', which corresponds to line (1) in Figure 2.6, outperforms the traditional fast-encoding algorithms in terms of speed-up, since traditional fast-encoding algorithms typically do not accelerate all CU decisions (as was also remarked in Chapter 2).

5.5 Conclusion

In this chapter, a model to select the most correlating HDR version for an LDR version of the same video was presented. By copying coding information from this video, the encoding of the LDR video can be accelerated, allowing content providers to serve consumers with many different televisions with different dynamic range capabilities.

Experimental results show that only CU information should be copied if coding efficiency is the most important concern of the video provider. However, if many devices with different dynamic range capabilities need to be served and encoder complexity becomes a concern, all information can be copied because it gives a good trade-off between speed-up and bit rate overhead. In this case, the video provider can effectively encode multiple dynamic-range versions of the same video for approximately the computational cost of a single encoder by using guided encoding.

References

[1] R. Boitard, M. T. Pourazad, P. Nasiopoulos, and J. Slevinsky. *Demystifying High-Dynamic-Range Technology: A new evolution in digital media*. IEEE Consum. Electron. Mag., 4(4):72–86, Oct. 2015.

[2] J. Lee, G. Jeon, and J. Jeong. *Piecewise tone reproduction for high dynamic range imaging*. IEEE Trans. Consum. Electron., 55(2):911–918, May 2009.

[3] J. W. Lee, R. H. Park, and S. Chang. *Tone mapping using color correction function and image decomposition in high dynamic range imaging*. IEEE Trans. Consum. Electron., 56(4):2772–2780, Nov. 2010.

[4] J. W. Lee, R. H. Park, and S. Chang. *Local tone mapping using the K-means algorithm and automatic gamma setting*. IEEE Trans. Consum. Electron., 57(1):209–217, Feb. 2011.

[5] K. Kim, J. Bae, and J. Kim. *Natural hdr image tone mapping based on retinex*. IEEE Trans. Consum. Electron., 57(4):1807–1814, Nov. 2011.

[6] J. W. Lee, R. H. Park, and S. Chang. *Noise reduction and adaptive contrast enhancement for local tone mapping*. IEEE Trans. Consum. Electron., 58(2):578–586, May 2012.

[7] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz. *Complexity control of high efficiency video encoders for power-constrained devices*. IEEE Trans. Consum. Electron., 57(4):1866–1874, Nov. 2011.

[8] J. Lee, S. Kim, K. Lim, and S. Lee. *A Fast CU Size Decision Algorithm for HEVC*. IEEE Trans. Circuits Syst. Video Technol., 25(3):411–421, Mar. 2015.

[9] S. Ahn, B. Lee, and M. Kim. *A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding*. IEEE Trans. Circuits Syst. Video Technol., 25(3):422–435, Mar. 2015.

[10] L. Pham Van, J. De Praeter, G. Van Wallendael, J. De Cock, and R. Van de Walle. *Machine learning for arbitrary downsizing of pre-encoded video in HEVC*. In Proc. IEEE Int. Conf. Consum. Electron. (ICCE), pages 406–407, Jan. 2015.

[11] A. J. Diaz-Honrubia, J. L. Martinez, P. Cuenca, J. A. Gamez, and J. M. Puerta. *Adaptive Fast Quadtree Level Decision Algorithm for H.264 to HEVC Video Transcoding*. IEEE Trans. Circuits Syst. Video Technol., 26(1):154–168, Jan. 2016.

[12] A. Jimenez-Moreno, E. Martnez-Enriquez, and F. Diaz de Mara. *Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard*. IEEE Trans. Multimedia, 18(4):563–575, Apr. 2016.

[13] H. S. Kim and R. H. Park. *Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule*. IEEE Trans. Circuits Syst. Video Technol., 26(1):130–138, Jan. 2016.

[14] C. E. Rhee, K. Lee, T. S. Kim, and H. J. Lee. *A survey of fast mode decision algorithms for inter-prediction and their applications to high efficiency video coding*. IEEE Trans. Consum. Electron., 58(4):1375–1383, Nov. 2012.

[15] K. Choi and E. S. Jang. *Early TU decision method for fast video encoding in high efficiency video coding*. Electron. Lett., 48(12):689–691, June 2012.

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

[17] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz. *Fast HEVC Encoding Decisions Using Data Mining*. IEEE Trans. Circuits Syst. Video Technol., 25(4):660–673, Apr. 2015.

[18] L. Shen, Z. Zhang, and P. An. *Fast CU size decision and mode decision algorithm for HEVC intra coding*. IEEE Trans. Consum. Electron., 59(1):207–213, Feb. 2013.

[19] H. Zhang and Z. Ma. *Fast Intra Mode Decision for High Efficiency Video Coding (HEVC)*. IEEE Trans. Circuits Syst. Video Technol., 24(4):660–668, Apr. 2014.

[20] H. Lee, H. J. Shim, Y. Park, and B. Jeon. *Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality*. IEEE Trans. Broadcast., 61(3):388–397, Sept 2015.

[21] D.H. Finstad, H.K. Stensland, H. Espeland, and P. Halvorsen. *Improved Multi-Rate Video Encoding*. In Proc. IEEE Int. Symposium Multimedia (ISM), pages 293–300, Dec. 2011.

[22] H. Espeland, H. K. Stensland, D. H. Finstad, and P. Halvorsen. *Reducing processing demands for multi-rate video encoding: implementation and evaluation*. Int. J. Multimedia Data Eng. Manag., 3(2):1–19, Apr. 2012.

[23] G. Cebrian-Marquez, A. J. Diaz-Honrubia, J. De Praeter, G. Van Wallendael, J. L. Martinez, and P. Cuenca. *A motion vector re-use algorithm for*

H.264/AVC and HEVC simultaneous video encoding. In Proc. ACM International Conference on Advances in Mobile Computing and Multimedia, pages 241–245, Dec. 2015.

[24] A. J. Diaz-Honrubia, J. De Praeter, G. Van Wallendael, J. L. Martinez, P. Cuenca, J. M. Puerta, and J. A. Gamez. *CTU splitting algorithm for H.264/AVC and HEVC simultaneous encoding*. J. Supercomputing, pages 1–13, Feb. 2016.

[25] D. Schroeder, P. Rehm, and E. Steinbach. *Block structure reuse for multi-rate high efficiency video coding*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 3972–3976, Sept. 2015.

[26] D. Schroeder, A. Ilangovan, and E. Steinbach. *Multi-rate encoding for HEVC-based adaptive HTTP streaming with multiple resolutions*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 1–6, Oct. 2015.

[27] J. De Praeter, A. J. Diaz-Honrubia, N. Van Kets, G. Van Wallendael, J. De Cock, P. Lambert, and R. Van de Walle. *Fast simultaneous video encoder for adaptive streaming*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 1–6, Oct. 2015.

[28] E. François, J. Sole, J. Ström, and P. Yin. *Common test conditions for HDR/WCG video coding experiments*. Technical Report JCTVC-W1020, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Feb. 2016.

[29] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description Update 2*. Technical Report JCTVC-T1002, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Feb. 2015.

[30] L. Wasserman. *All of Nonparametric Statistics*, pages 13–25. Springer New York, 2006.

[31] D. M. Corey, W. P. Dunlap, and M. J. Burke. *Averaging correlations: Expected values and bias in combined Pearson rs and Fisher's z transformations*. J. General Psychology, 125(3):245–261, 1998.

[32] R. M. Rifkin and R. A. Lippert. *Notes on regularized least squares*. Technical Report CBCL-268, Computer Science and Artificial Intelligence Laboratory, May 2007.

[33] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

6

Video Encoder Architecture for Personalized Bit Rate Representations

6.1 Introduction

In a fourth and final scenario discussed in this dissertation, guided encoding is used in order to enable real-time video-streaming events with many participants, such as virtual classrooms and video conferences. These events require a low delay when transmitting a video to a receiver in order to provide a good quality of experience. However, maintaining such a low delay poses a considerable challenge when participants have a fluctuating connection due to e.g. an unstable wireless access point or network congestion.

A common approach to handle the fluctuating bandwidth of participants is to use adaptive streaming [1]. Using this solution, the original video is encoded at different quality levels that each target a different range of bit rates. Depending on the current bandwidth capacity of the user, a video segment of the version with an appropriate bit rate is transmitted to this user. If the bandwidth capacity of the user changes during some point in time, the user instead receives video segments from either a higher bit rate or lower bit rate version of the video.

This approach has several disadvantages. First, if the bandwidth capacity of a user becomes smaller than the bit rate of the video segment, the video will freeze until the next, lower quality, video segment is received. To reduce the delay in such a scenario, the provider could encode shorter video segments. However, if the segment length decreases, so does the compression efficiency of the video,

meaning that the user will receive a lower quality video for the same bit rate. Although the freezing of the video can also be prevented by having the receiver buffer several frames and display them with some delay, this solution is not an option in real-time video streaming due to its strict low-delay requirements.

A second disadvantage of the adaptive streaming approach is that, although attempts can be made to increase the overall fairness of the system [2], the bandwidth of an individual user might not be used to its full capacity, even in the case of a stable network connection. This occurs in cases where e.g. a low-quality and high-quality video are offered at a bit rate of respectively 1 Mbps and 10 Mbps, whereas the receiver has a bandwidth capacity of 5 Mbps. Since the capacity is too small for the high-quality version, the user will receive the low-quality version, despite having a much higher capacity. In this case, if it were to exist, a video encoded at 5 Mbps would have offered the user a better video quality by making the most out of the available bandwidth capacity.

The disadvantages of adaptive streaming in terms of delay and bandwidth utilization are overcome by using protocols for low-latency communication such as the real-time streaming protocol [3] and WebRTC. These solutions generally rely on providing a personalized bitstream over a single connection with each end-user. By adapting the bit rate of the video stream depending on the network conditions of the client, there are no delays due to segment switching, and the bandwidth of each client can be fully utilized to provide the best video quality. However, in an interactive video-streaming event with many participants, such an approach cannot be applied in practice, since providing a personalized bitstream for each receiver would require a large amount of computationally expensive video encoders.

In order to make a system for low-delay video streaming viable for events with many participants, this chapter proposes an architecture based on calculating coding information for a fixed number of bit rate representations of the video and supplying this coding information to very low-complex residual encoders. Each individual residual encoder can provide a personalized bitstream to each user over a wide range of bit rates. By increasing the amount of bit rate representations for which coding information is calculated, the compression efficiency of each generated bitstream increases at the cost of an overall increase in computational complexity of the system.

As its main contribution, this chapter enables the realization of a system for low-delay video communications by overcoming the disadvantage of having an excessive amount of computationally complex video encoders. As its second contribution, this chapter also investigates the effect of varying the amount of modules for calculation of coding information.

The remainder of this chapter describes the current state-of-the-art, followed by the proposed method in Section 6.3. Next, the method is evaluated in Section 6.4. Finally, the conclusion is presented in Section 6.5.

6.2 Related work

Since a real-time video-streaming system with many participants with a single bitstream per user is computationally complex, a possible solution would be the use of fast encoding algorithms. These algorithms attempt to reduce the computational complexity of a video encoder by limiting the number of encoding decisions evaluated by an encoder. For example, the computational complexity of encoders using the H.264/AVC video compression standard [4] has been reduced by limiting the number of block prediction modes [5–9], or by accelerating the motion estimation [10–12] or intra-prediction process [13–17]. Sometimes, different strategies are combined as well to further reduce the encoding complexity [18].

The computational complexity of the successor of H.264/AVC, HEVC [19], has been reduced in similar ways. Common approaches consist of limiting the number of block partitioning modes [20–24] or intra-prediction modes [25, 26]. Other works have also attempted to reduce the HEVC encoder complexity by accelerating motion estimation [27] or by using fast-skip decisions [28, 29]. Finally, some research further reduces the encoding complexity by limiting several types of coding information at once [30–36]. However, in all fast encoding algorithms for H.264/AVC and HEVC, the encoder still has to make a large amount of encoding decisions. As a result, although the computational complexity is reduced, the complexity reduction provided by conventional fast encoders results in a speed-up of a factor four at most, which means that four accelerated encoders have a similar computational complexity as one non-accelerated encoder. This speed-up is still insufficient for a real-time video-streaming application with many participants.

Besides creating a real-time video-streaming system by accelerating individual fast encoders, it might be possible to use a simultaneous bit rate encoder to encode multiple bit rate representations of the same video at the same time at a reduced computational cost. This idea has been proposed before in the context of adaptive video streaming for HEVC. Such an encoder performs a non-accelerated encode of a video bitstream at a certain bit rate while the encodings of other representations of the video (at different bit rates) are accelerated by limiting the coding decisions based on the coding information obtained from the non-accelerated encoding. In particular, block partitioning decisions [37, 38], as well as intra-prediction modes and motion vector decisions [39] have been used in these methods. However, as with the encoding accelerations of a conventional fast encoder, the complexity reduction is still too limited (speed-up factor of four) to make real-time live-streaming events with many users feasible.

Finally, there exists a technique for non-real-time streaming that can greatly reduce the computational complexity of an encoder [40, 41]. This method uses control streams, which are regular video streams from which the residual information is removed. Only these control streams are then stored on a server, instead

of the entire video as is the case with adaptive streaming. As a result, less storage space is required to store these control streams compared to a full bitstream. When the bitstream is requested by a client, the coding decisions stored in the control stream are used to perform fast-transcoding with a very low complexity on a high quality bitstream, in order to re-create the bitstream from which the control stream had been extracted.

Inspired by the ideas of control streams, and contrary to both traditional fast encoders and simultaneous encoders that only provide a limited computational complexity reduction in favor of retaining a higher compression efficiency, the system proposed in this chapter aims to drastically reduce the computational complexity. By doing so, the use of personalized bitstreams becomes a viable solution for low-delay live-streaming to many users.

6.3 Proposed method

A video bitstream consists of coding information (e.g. block partitioning modes and motion vector information) and the residual picture. Normally, all of this information is calculated by a traditional, non-accelerated video encoder. Consequently, if multiple users each require an individual bitstream, multiple traditional encoders are needed to create these bitstreams. Each of these bitstreams thus has its own coding information and residual picture. However, if each user receives a personalized bitstream, some users might require bitstreams in similar bit rate ranges. These bitstreams might have similar coding information, as is observed with the block structures of videos encoded with HEVC [37]. In such a case, instead of recalculating the coding information for each individual bitstream, it may be better to create a system that encodes multiple bitstreams with the same coding information.

In the following subsection, an architecture based on sharing coding information is proposed in order to provide personalized bitstreams to users in a low-delay live-streaming scenario. Next, the theoretical complexity of the proposed architecture is examined.

6.3.1 Architecture

As mentioned above, a video bitstream consists of coding information and the residual picture. It would thus be possible to use one set of coding information to generate several bitstreams at the same time. To achieve this, the proposed system shown in Figure 6.1 contains two kinds of modules: coding information calculation (CIC) modules and residual encoder (RE) modules. A CIC module calculates coding information for a certain bit rate representation of the video. This information is then shared with the RE modules. These modules each create

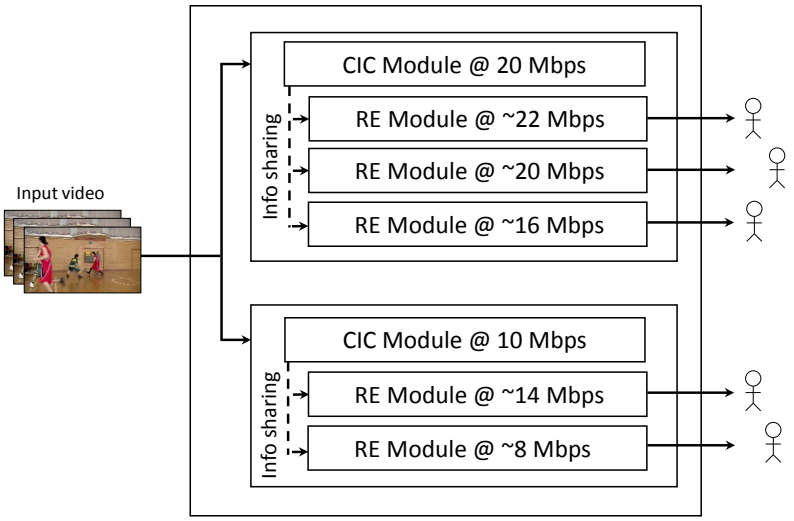


Figure 6.1: The proposed architecture when using two coding information calculation (CIC) modules and five residual encoder (RE) modules.

a standard-compliant bitstream by using the supplied coding information to calculate a residual picture. The bit rate of each video is then adapted by appropriately reducing the size in bits of the residual picture (e.g. by adjusting the quantization parameter of the frame) depending on the available bandwidth of the user connected to the RE module.

It should also be noted that the CIC module and RE modules are synchronized, meaning that in block-based encoders, the RE module encodes a block immediately after the coding information has been calculated by the CIC module. As a result, no extra delay is introduced compared to using traditional encoders, since these encoders also first have to calculate coding information for a block before encoding the residual picture. Further note that each user also needs to be synchronized with his RE module, since communication between the user and the module is necessary in order to have the module adjust to the bandwidth of the user. As a result, the entire system should be kept synchronized, which may result in some extra complexity for the implementation of the architecture. However, in case of less strict latency requirements, the need for synchronization would be smaller, and CIC module could thus temporarily store the coding decisions for later use by RE modules, meaning that only synchronization between users and RE modules is necessary.

Although a system with a single CIC module may be able to provide bitstreams at a wide range of bit rates, the coding efficiency will plummet if the bit rate of the bitstreams generated by the RE modules differs much from the bit rate representa-

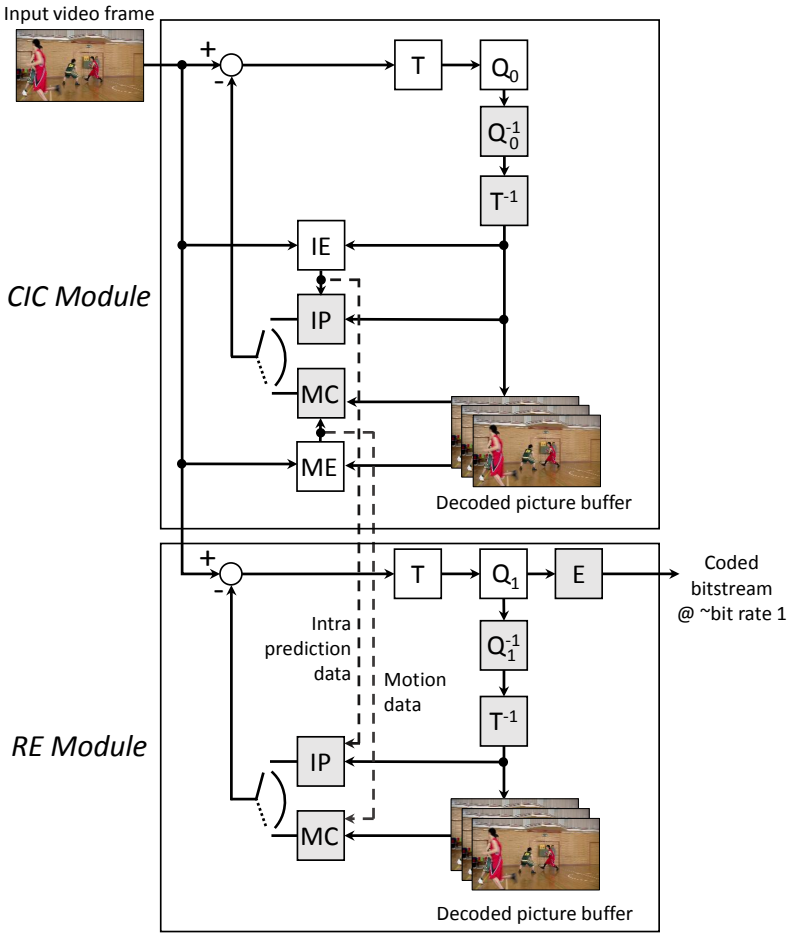


Figure 6.2: Detailed view of a CIC-RE module pair. The CIC module resembles a traditional encoder loop, whereas the RE module mostly contains components with a similar complexity as a traditional decoder. (T: transform; Q: quantization; E: entropy encode; IE: intra-picture estimation; IP: intra-picture prediction; MC: motion compensation; ME: motion estimation)

tion used by the CIC module. After all, in such a scenario, the coding information of the CIC module may no longer be a good fit, since the residual picture used within the CIC module would greatly differ from the residual picture used in the RE module. For example, a block-based video encoder will use larger block sizes at lower bit rates, since a higher quantization parameter removes detailed texture from the picture, making it easier for motion estimation to find a good match in previous pictures for large areas. At high bit rates, on the other hand, textures are better preserved, meaning that smaller blocks are used. If the large block sizes

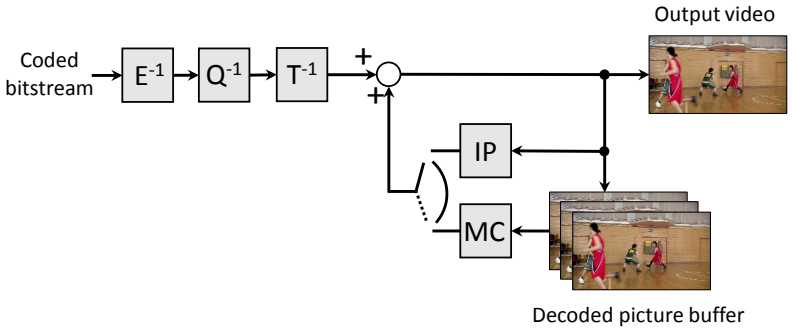


Figure 6.3: Detailed view of typical video decoder. (T^{-1}: inverse transform; Q^{-1}: dequantization; E^{-1}: entropy decode; IP: intra-picture prediction; MC: motion compensation)

of the lower bit rate are used to encode the version with a high bit rate, or vice versa, this will result in inefficient coding decisions, leading to a lower compression efficiency. In order to mitigate this effect of copying coding information from versions with a large difference in bit rate, the proposed architecture in this chapter consists of multiple CIC modules (an example with two is shown in Figure 6.1).

In the proposed architecture with multiple CIC modules, each CIC module calculates coding information for a different constant bit rate of the video. Each user is then connected to an RE module which is connected to the CIC module with the most similar bit rate. For example, if two available CIC modules generate coding information for a bit rate representation of respectively 10 Mbps and 20 Mbps, an RE module encoding a video at 14 Mbps would be connected to the CIC module of 10 Mbps. By doing this, the RE module can use the coding information that fits its bit rate representation the most, thus limiting the decrease in compression efficiency that occurs due to copying non-optimal coding information.

Note that in this architecture, RE modules should not be constrained to always being connected to the same CIC module. For example, if the internet connection of a user drops from 22 Mbps to 8 Mbps, an RE module that was previously using coding information supplied by a CIC module of 20 Mbps should switch to using coding information from a CIC module of 10 Mbps.

Figure 6.2 shows a more detailed view of the interaction between the CIC module and an RE module in the case of a block-based video encoder such as H.264/AVC, HEVC, and VP9. The CIC module resembles the encoding loop of a traditional video encoder: for each input video frame, the best coding decisions for intra-picture prediction are determined through intra-picture estimation, whereas the best coding decisions for inter-picture prediction are determined through motion estimation. Using these decisions, the residual picture is calculated, transformed and quantized with a quantization parameter Q_0. The quantized picture is

then dequantized and inversely transformed in order to be stored in a decoded picture buffer which contains the reference pictures for motion estimation and compensation. Note that contrary to a traditional encoder, there is no need to apply entropy encoding to the quantized picture since the CIC module should not generate an actual bitstream.

The encoding loop of the RE module on the other hand does not have the computationally intensive steps of intra-picture estimation and motion estimation since all coding information such as block partitioning modes, intra-coding decisions, and motion estimation decisions are copied from the CIC module. Using these coding decisions, the RE module performs intra-picture prediction and motion compensation in order to calculate the residual picture, which is then transformed and quantized with a quantization parameter Q_i for an RE module i. Note that usually $Q_i \neq Q_0$ since Q_i is used to adapt the bit rate of the generated bitstream to the available bandwidth. The resulting picture is then entropy encoded and transmitted to the user.

6.3.2 Analysis of theoretical complexity

Since the actual computational complexity of the system depends on variables such as the used video compression standard and the exact encoder implementations, this section makes a theoretical approximation of the expected computational complexity independent of the used standard and implementation.

As was explained in the previous subsection, a CIC module resembles a traditional video encoder, excluding the entropy encoding step. Therefore, the required computational complexity of a CIC module can be approximated by the computational complexity C_e of a traditional video encoder using the same video compression standard. In contrast, the complexity of an RE module is much lower, since this module does not contain the computationally complex steps of intra-picture estimation and motion estimation. When comparing the remaining components of the RE module to a traditional decoder as seen in Figure 6.3, one can see that the intra-prediction, motion compensation, inverse transformation, and quantization steps are found both in the RE module and the decoder. Additionally, the computational complexity of the entropy encoding step in the RE module is similar to the complexity of the entropy decoding step in a decoder. As such, since the computational complexity of the remaining transform and quantization steps is small, the complexity of an RE module (C_{RE}) can be approximated by the computational complexity C_d of a traditional video decoder using the same compression standard for a video encoded at the same target bit rate, multiplied by a small factor α.

The above approximation was tested for HM version 16.5 [42] by using CIC modules to calculate coding information for a low-delay configuration, for all quantization parameter values from 22 up to 37, and for all video sequences that

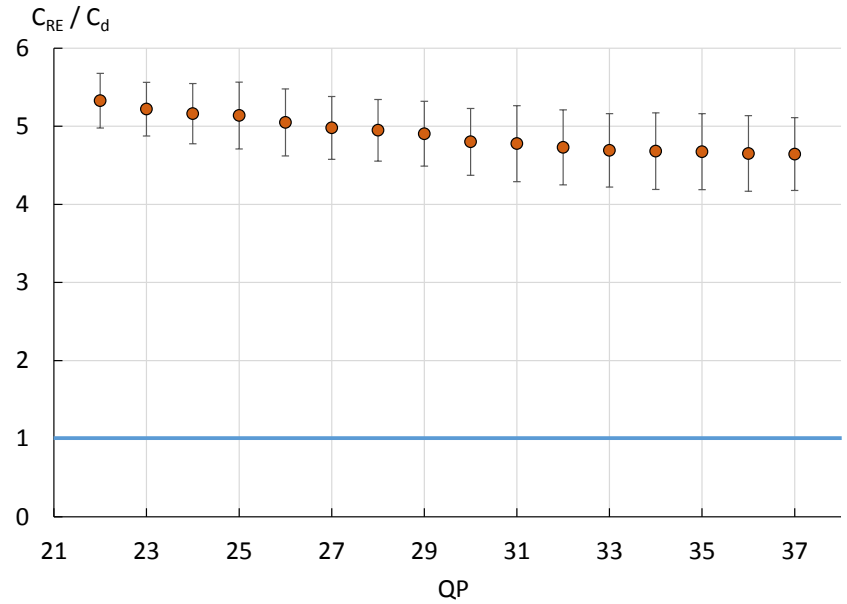


Figure 6.4: The average computational complexity of an RE module (C_{RE}) equals about five times the computational complexity of a decoder (C_d) when tested with the HEVC reference software.

are also used in section 6.4 (listed in Table 6.1). The information generated by these CIC modules was then used by RE modules to encode these sequences (thus $Q_0 = Q_i = QP$ for all tested quantization parameters). Additionally, the time needed to decode each bitstream was also measured. The result of this test is found in Figure 6.4, which shows $\alpha = \frac{C_{RE}}{C_d}$ averaged over all tested sequences, with the error bars indicating the standard deviation. This measure thus expresses the complexity of an RE module relative to the complexity of a decoder. As seen in this figure, in the case of HM 16.5, the complexity of an RE module equals about five times the complexity of a decoder. However, do note that in HEVC, the transformation step also includes partitioning each CU into TUs (see also section 2.2), and the encoder also has to determine parameters for the sample-adaptive offset filter, which might further contribute to a higher complexity of the RE modules compared to a decoder. As such, it can be assumed that for HM 16.5, $\alpha < 5$.

Using the above approximations C_e and αC_d for the computational complexity of respectively the CIC module and RE module, the total complexity of the proposed architecture for m users while providing n CIC modules can be estimated as

$$C(m, n) \approx nC_e + m\alpha C_d \,. \tag{6.1}$$

In most common video compression formats, decoding complexity is always smaller than encoding complexity. Therefore, C_e equals a multiple k of C_d, which allows rewriting the previous equation as

$$C(m,n) \approx nC_e + m\frac{\alpha}{k}C_e\,. \tag{6.2}$$

This implies that the complexity of the system increases by C_e for every CIC module that is added to the system, whereas the complexity also increases by the same amount for every k/α extra users that participate in the live-streaming event (with $k = C_e/C_d$). Since both the parameters k and α depend on the used video compression standard and encoder/decoder implementations, the relative computational cost of adding an extra CIC module compared to supporting more users will highly depend on the used video codecs.

6.4 Evaluation

The previous sections discussed a generic architecture for providing personalized bitstreams to users. However, since the actual compression efficiency of individual bitstreams depends on the used video codec, the performance of the architecture has been evaluated by applying it to the HEVC video compression standard. In particular, the simulations were carried out using HM version 16.5 [42]. In order to simulate RE modules, the encoder of this software has been modified to skip all encoding decisions except for the calculation of the residual quadtree and the residual itself by using coding information that is calculated for a certain quantization parameter by a CIC module (Q_0). Since the architecture is aimed at low-delay video, a low-delay configuration (*encoder_lowdelay_main*) is used. This configuration consists of an intra-coded frame (I-frame) followed by inter-coded frames (B-frames) that only use previous frames as reference pictures.

In the following subsections, the compression efficiency of the residual encoders is first evaluated to verify the effect of using coding information calculated for an encoding at a different quantization parameter than the one used by the RE module. Next, the impact of varying the amount of CIC modules is determined in terms of its effect on the overall compression efficiency of the system. The effects of bandwidth variations and I-frame retransmissions, as well as a scenario where an RE module switches from copying coding information from one CIC module to another are then investigated next. Finally, the performance of the RE modules is compared to the current state-of-the-art.

6.4.1 Compression efficiency of RE modules

Since the quantization parameter used by the RE module (Q_i) does not necessarily equal the quantization parameter of the corresponding CIC module (Q_0), the cod-

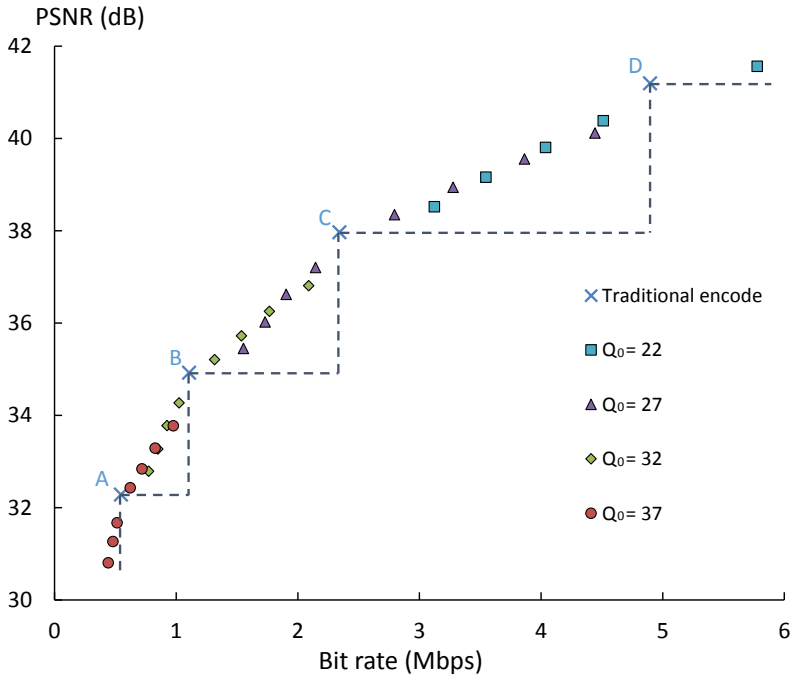


Figure 6.5: Compression efficiency of RE modules when copying coding information from CIC modules using a quantization parameter Q_0 compared to the compression efficiency of traditional encoders using the same quantization parameters. The RE modules are able to cover the range of bit rates not covered by a fixed amount of traditional encoders while providing an increasingly higher video quality.

ing information copied from the CIC module may be suboptimal. As a result, the compression efficiency of an RE module will be lower than a traditional encoder using the same quantization parameter. However, since the goal of the proposed system is to use all available bandwidth capacity of users in order to provide a higher video quality than what they would receive in a scenario with a fixed number of encoders, the architecture is useful as long as this goal is achieved.

To test whether the above goal can be achieved, all tested video sequences (see also Table 6.1) were encoded with a traditional encoder with quantization values of 22, 27, 32, and 37. These results are compared to RE modules copying coding information from CIC modules for which Q_0 equals these same values. For each CIC module with a quantization parameter Q_0, eight RE modules were tested with the quantization parameter of RE module i being Q_i. The relation between Q_0 and Q_i is

$$Q_0 = Q_i + \Delta Q_i \,, \tag{6.3}$$

with ΔQ_i ranging from -4 up to 4, excluding 0, since the compression efficiency of such an encode is the same as the compression efficiency of the traditional encoding with the same quantization parameter.

Figure 6.5 shows the results for the sequence *BasketballDrill*. In this RD-curve, the bit rate is shown on the x-axis, whereas the distortion on the y-axis is expressed as PSNR. Four RD-points of a traditional encode are shown, indicated by the letters A, B, C, and D for quantization parameters 37, 32, 27, and 22, respectively. Of these four points, the dashed lines indicate either the bit rate (vertical lines) or PSNR (horizontal lines) that these RD-points result in. All other points on the chart belong to encodings generated by RE modules by copying coding information from CIC modules with a certain Q_0.

The figure shows that the RE modules can successfully cover the range of bit rates between two subsequent traditional encodings. Moreover, the RE modules offer a higher quality than the traditional encoding when the bit rate increases, and a lower quality when the bit rate decreases. For example, this is seen by comparing the RD-points of the RE module with $Q_0 = 27$ to RD-point C of the traditional encode. The four RE modules based on this Q_0 with a $Q_i < Q_0$ all have an increasingly higher quality with an increasingly higher bit rate. This bit rate remains smaller than the next RD-point of a traditional encode (point D, with quantization value 22). Similarly, the four RE modules based on $Q_0 = 27$ with a $Q_i > Q_0$ decrease in quality while also decreasing in bit rate. However, at all times, both the bit rate and quality remain higher than the traditional encode with a quantization value of 32 (point B). This observation can also be made for all other RE modules by comparing their bit rate and PSNR to the bit rate and PSNR indicated by the dashed lines.

The occurrence of the same behavior has been verified for all other test sequences using the same values for Q_0 and Q_i. Since this observation shows that the RE modules can make the most out of the bandwidth of users by offering an appropriate video quality for bit rates that cannot be achieved by only having a limited amount of traditional encoders, it can be concluded that the architecture can successfully provide users with a personalized video stream based on the available bandwidth.

6.4.2 Effect of CIC modules on compression efficiency

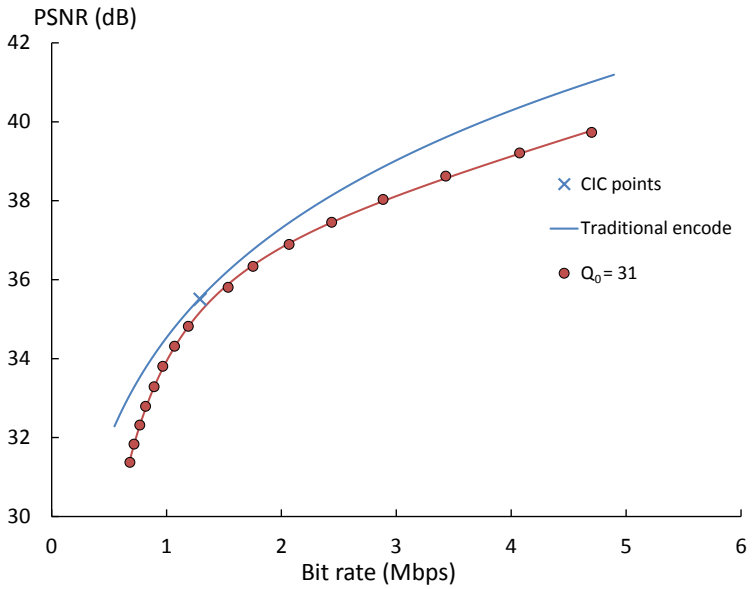
As an additional observation in Figure 6.5, it can be seen that the compression efficiency of an RE module decreases when $|\Delta Q_i|$ increases. For example, if $Q_i = 28$ the RE module could use information from either a CIC module with $Q_0 = 27$ or from a CIC module with $Q_0 = 32$. In the former case, ΔQ_i equals 1 and the resulting bit rate is about 2.1 Mbps while the PSNR is 37.2 dB. In the latter case, ΔQ_i equals -4. The resulting bit rate is also about 2.1 Mbps. However, the

PSNR is 36.8 dB, which is 0.4 dB lower compared to an RE module with a smaller $|\Delta Q_i|$. Therefore, in order to increase the compression efficiency of the complete system, the distance between the subsequent Q_0 values should be decreased.

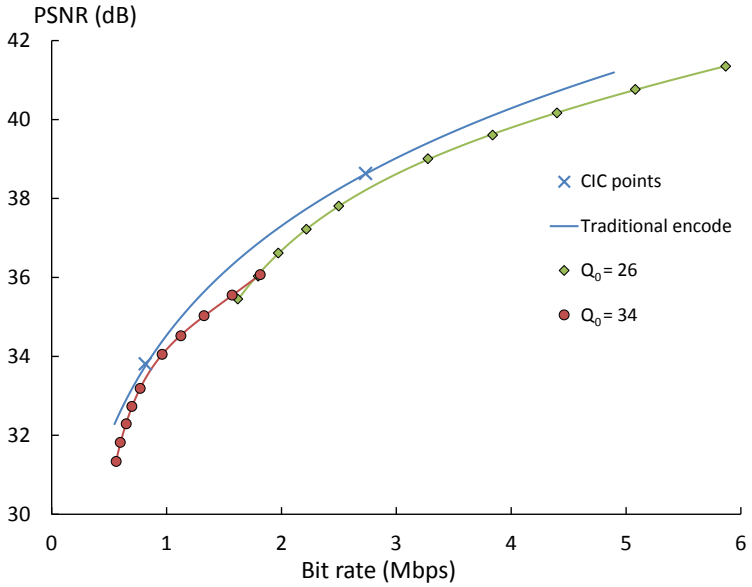
A straightforward way to decrease the distance between Q_0 values is the addition of more CIC modules to the system. However, as explained in Section 6.3, CIC modules have a similar computational complexity as a traditional encoder. Consequently, the number of these modules should be limited as much as possible. This section thus aims to investigate the effect of the number of CIC modules on the compression efficiency of the system.

Figure 6.6 shows the effect of adding more CIC modules for the sequence *BasketballDrill*. The values for Q_0 were chosen to be spread as uniformly as possible across the examined range of quantization values (22 up to 38), since this spread of Q_0 values was experimentally determined to be optimal. Each subfigure shows the interpolation of four RD-points with quantization values of 22, 27, 32, and 37 using a traditional encoder. This interpolation should be compared to the curves resulting from the interpolation of the RD-points of the RE modules that use coding information from CIC modules with different values of Q_0. Each distinct value of Q_0 has a separate curve resulting from encodings with RE modules using different values of Q_i. The CIC points, which are RD-points obtained when $Q_i = Q_0$ are also shown. This figure shows that when only a single CIC module is used, the compression efficiency of the system is much smaller than in the case of using traditional encoders. However, when the amount of CIC modules increases, this difference decreases.

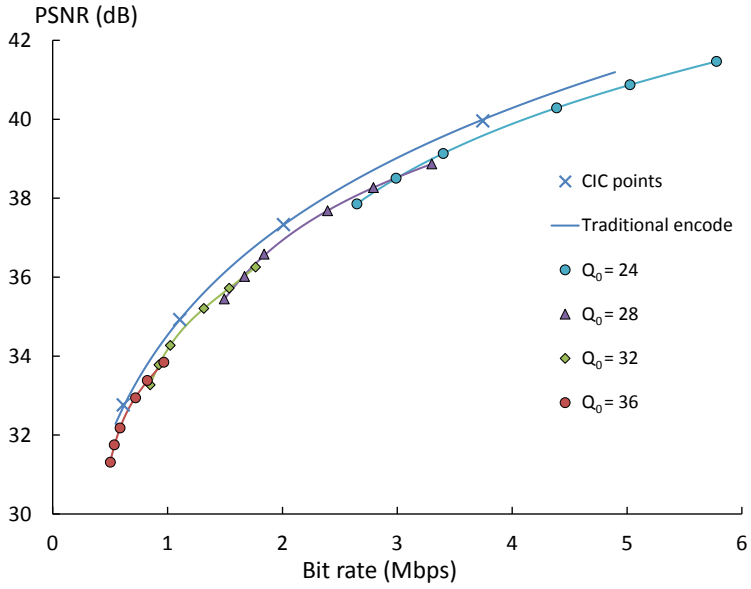
In order to better quantify the compression efficiency of the whole system, an appropriate metric is needed. Normally, when measuring the difference in compression efficiency between two encoders, this difference can be expressed as BD-rate [43]. This metric measures the relative average difference in bit rate between different encoders for the same quality and requires at least four RD-points for each of the encoders in order to create a fitting interpolation for each set of four RD-points. The average bit rate overhead is then measured as the area between the two resulting curves averaged over the range of PSNR-values covered by both curves. However, in the case of the system proposed in this chapter, the traditional BD-rate metric can only be applied in the case when one CIC module is used. Therefore, the BD-rate metric should be extended to BD-rate*. With BD-rate*, one reference encoding is created as usual by creating an interpolated curve based on the RD-points of four traditional encodings with quantization values 22, 27, 32, and 37. However, since there can be multiple CIC modules, multiple curves need to be compared to the curve of the reference encoding. Therefore, BD-rate* calculates the area between the reference encoding and the most compression-efficient curves in each PSNR-range (the curve with the lowest bit rate in the PSNR-range), as illustrated with a gray color in Figure 6.7. Similar to the normal BD-rate met-



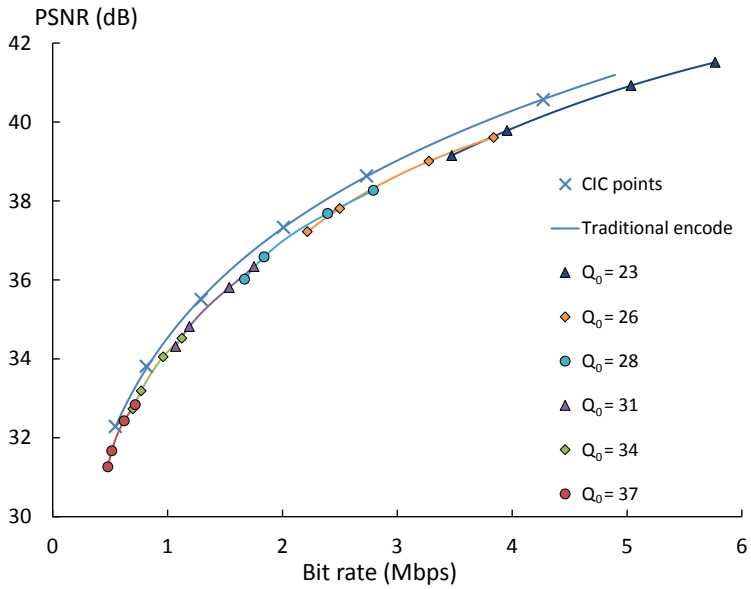
(a) 1 CIC module



(b) 2 CIC modules



(c) 4 CIC modules



(d) 6 CIC modules

Figure 6.6: Comparison between using 1, 2, 4, or 6 CIC modules. The compression efficiency of the RE modules increases as more CIC modules are used.

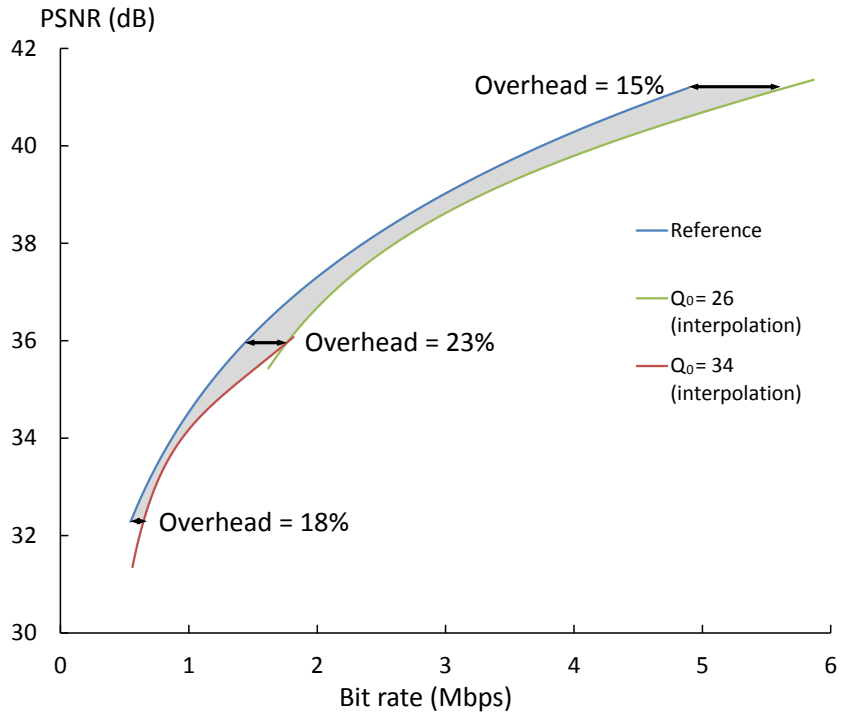


Figure 6.7: Illustration of worst and average bit rate overhead of the system. The worst bit rate overhead of the system is calculated as the highest overhead across the tested range (23% in this example). The average bit rate overhead is calculated by averaging the area between the reference measurements and the RE encoders (marked in gray) over the tested range.

ric, this area is then averaged over the entire range of PSNR-values covered by the reference encoding and the most compression-efficient curves.

Besides the average bit rate overhead, BD-rate*, the worst-case bit rate overhead is also calculated by finding the quality value for which the distance between the reference curve and the most compression-efficient curves is the largest. In order to determine the worst-case bit rate overhead, the intersections between the most compression-efficient curves as well as the lowest and highest RD-points of the reference encoding are examined. This is also illustrated in Figure 6.7 where the bit rate overhead at the lowest and highest quality of the reference encoding is respectively 18% and 15%. However, the bit rate overhead is even larger at the intersection between the two most compression-efficient curves. As such, the worst-case bit rate overhead is 23% in this example.

The results of all tested sequences in terms of average bit rate overhead are shown in Table 6.1. This table further illustrates the behavior observed in Fig-

					BD-rate* (%) when using n CIC modules									
Class	Sequence	Resolution	Frames	$n =$	1	2	3	4	5	6	7	8	9	10
A	PeopleOnStreet	2560×1600	150		24.4	15.0	13.2	12.4	12.1	11.8	11.7	11.6	11.5	11.5
	Traffic	2560×1600	150		33.3	20.9	17.9	16.8	16.2	15.9	15.7	15.6	15.5	15.5
	Average				28.8	18.0	15.5	14.6	14.1	13.9	13.7	13.6	13.5	13.5
B	BasketballDrive	1920×1080	500		24.9	14.5	12.5	11.8	11.6	11.3	11.1	11.0	11.0	11.0
	Cactus	1920×1080	500		30.5	20.4	17.5	15.8	14.6	14.4	14.1	14.1	13.8	13.7
	Kimono1	1920×1080	240		17.3	11.2	9.8	9.5	9.2	9.0	9.0	8.9	8.9	8.9
	ParkScene	1920×1080	240		29.8	17.7	15.1	14.3	13.8	13.5	13.4	13.3	13.2	13.1
	Average				25.6	16.0	13.7	12.9	12.3	12.1	11.9	11.8	11.7	11.7
C	BasketballDrill	832×480	500		22.5	14.0	12.4	11.8	11.6	11.4	11.3	11.2	11.2	11.2
	BQMall	832×480	600		25.1	15.5	13.6	13.1	12.7	12.5	12.4	12.3	12.2	12.2
	PartyScene	832×480	500		24.9	14.4	12.5	11.9	11.6	11.4	11.2	11.2	11.1	11.1
	RaceHorses	832×480	300		22.8	13.0	11.3	10.8	10.6	10.4	10.2	10.1	10.1	10.1
	Average				23.8	14.2	12.5	11.9	11.6	11.4	11.3	11.2	11.2	11.1
D	BasketballPass	416×240	500		21.3	12.6	11.2	10.8	10.5	10.3	10.3	10.2	10.1	10.1
	BlowingBubbles	416×240	500		28.2	16.3	14.2	13.4	13.0	12.8	12.6	12.5	12.5	12.5
	BQSquare	416×240	600		34.1	17.3	14.4	13.5	13.0	12.7	12.4	12.3	12.3	12.3
	RaceHorses	416×240	300		25.8	14.4	12.6	12.0	11.7	11.4	11.3	11.2	11.2	11.1
	Average				27.4	15.2	13.1	12.4	12.0	11.8	11.7	11.5	11.5	11.5
F	BasketballDrillText	832×480	500		22.0	13.8	12.2	11.8	11.4	11.3	11.1	11.1	11.1	11.0
	ChinaSpeed	1024×768	500		25.9	15.2	13.1	12.4	12.1	11.9	11.7	11.7	11.5	11.5
	SlideShow	1280×720	500		15.3	9.9	8.9	8.5	8.4	8.3	8.2	8.1	8.1	8.1
	Average				21.1	12.9	11.4	10.9	10.7	10.5	10.4	10.3	10.2	10.2
Average for all sequences					25.2	15.1	13.1	12.4	12.0	11.8	11.6	11.5	11.5	11.5

Table 6.1: Average Bit Rate Overhead (BD-rate*) of the Proposed System for all Used Test Sequences

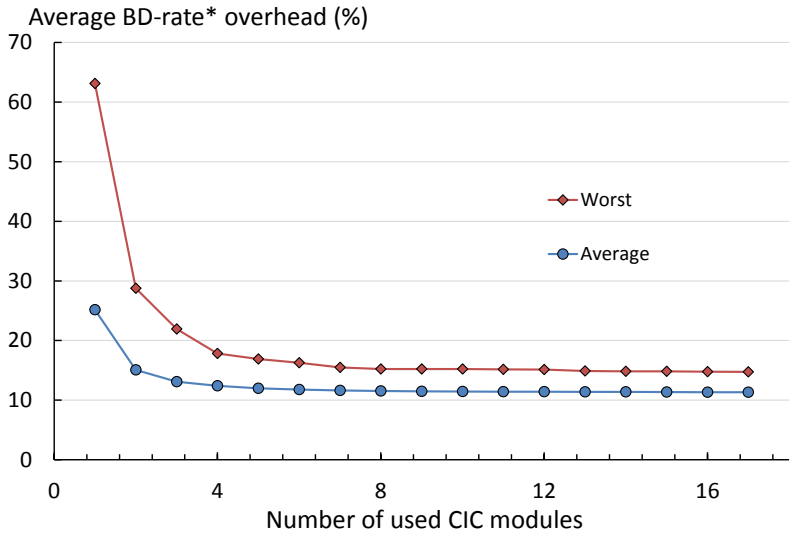


Figure 6.8: Relation between the compression efficiency and the number of used CIC modules. As the number of CIC modules increases, the amount of improvement in compression efficiency converges.

ure 6.6. For example, for the Class A sequence *PeopleOnStreet*, the average bit rate overhead is 24.4% when using only one CIC module. This overhead is almost halved to 15.0% by increasing the number of CIC modules to two. However, adding another CIC module only further decreases the overhead to 13.2%. Between six and seven CIC modules, the bit rate overhead only decreases from 11.8% to 11.7%. As the number of CIC modules is yet further increased, the bit rate overhead eventually converges. This behavior is also illustrated as an average over all sequences in Figure 6.8. As is seen in this figure, the worst-case bit rate overhead also follows a similar trend of decreasing sharply when increasing the number of CIC modules from one to two, but then stops decreasing as more CIC modules are added. However, increasing the number of these modules still contributes to the overall computational complexity. Therefore, it is advisable to limit the number of CIC modules in the proposed system to six or less.

6.4.3 Handling of small bandwidth variations and packet loss

In the previous section, the effect of the amount of CIC modules on the global compression efficiency of the system was investigated. However, this evaluation did not take possible bandwidth fluctuations and packet loss for individual RE modules into account. Therefore, in this subsection, the effects of these fluctuations on individual RE modules are analyzed.

To simulate bandwidth fluctuations, an RE module with a quantization parameter value of $Q_{i,1}$ switches to using a quantization parameter value of $Q_{i,2}$ after the sixteenth frame, with the relation between the two values being defined as

$$Q_{i,1} = Q_{i,2} + \Delta Q_{i,1-2}\,. \tag{6.4}$$

Furthermore, at all times, the RE module copies coding information from a CIC module with a quantization parameter value of $Q_0 = Q_{i,2} + \Delta Q_{i,2}$. For each test sequence from the previous subsection, four RE modules are tested with $Q_{i,2}$ values of 22, 27, 32, and 37. These modules and compared in terms of BD-rate overhead to a traditional encoding with the same four quantization parameters. Note that in order to map out the evolution over time, for each frame, the BD-rate is calculated by only taking all previous frames after the switch into account. Some results of this experiment for the *BlowingBubbles* sequence are shown in Figure 6.9. In this figure, it is seen that for negative values of $\Delta Q_{i,2}$, the first frames have a smaller BD-rate immediately after the change, whereas the opposite holds true for positive values. The reason for this behavior is that in the case of a negative $\Delta Q_{i,2}$, the PSNR of the frames before the switch are higher, meaning that the frames after the switch will have higher-quality reference frames. As more frames are encoded, this effect dissipates, as is seen by all configurations converging to a similar range of values between 9% and 10% BD-rates.

When examining the average behavior of all sequences in Table 6.2, similar behavior can be seen. This table shows the additional BD-rate overhead when the bandwidth fluctuates from $Q_{i,1}$ to $Q_{i,2}$ while copying coding information from a CIC module with $Q_0 = Q_{i,2}+\Delta Q_{i,2}$ compared to a situation where the bandwidth does not fluctuate ($Q_{i,1} = Q_{i,2}$, meaning that $\Delta Q_{i,1-2} = 0$). As with the figure in the previous paragraph, when $\Delta Q_{i,1-2} < 0$ (a switch from a higher to a lower quality), the overhead of the fluctuating scenario is smaller than in the case without fluctuation, whereas for $\Delta Q_{i,1-2} > 0$ (a switch from a lower quality to a higher quality), the opposite is true.

In a second experiment, a method to recover from packet losses due to extreme bandwidth fluctuations or other problems in the network is simulated. This method consists of having a separate CIC module that calculates coding information for the video with an *all-intra* configuration, meaning that it calculates the optimal coding information for each frame to encode it as an intra-frame. In case of packet loss, an RE module can then recover by transmitting an intra-frame encoded using the coding information from this CIC module, and then continue by copying coding information from a CIC module calculating coding information for inter-frames. The effect of inserting the coding information of such an intra-frame is shown in Figure 6.10, where the video has recovered from packet loss by coding the sixteenth frame as an intra-frame. Since this intra-frame has a higher PSNR than the inter-coded frames using the same quantization parameter, the second frame

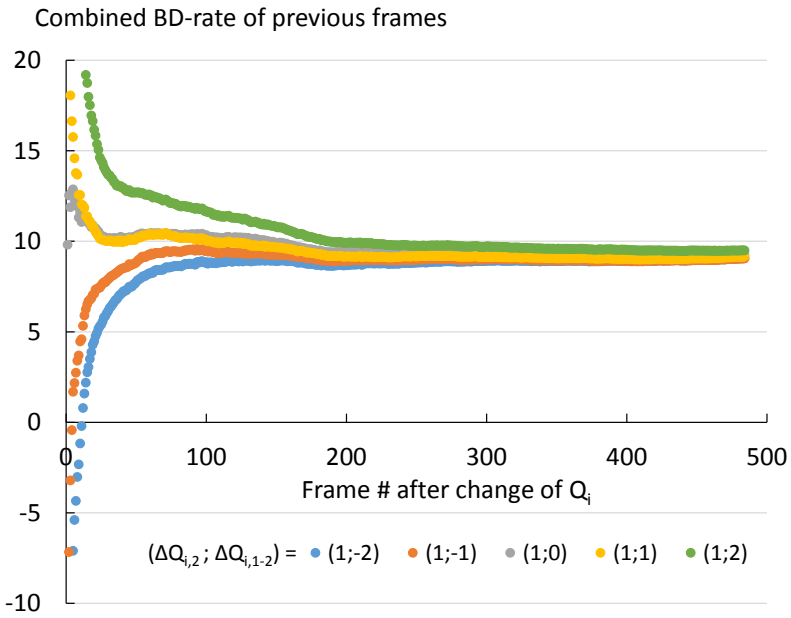


Figure 6.9: Evolution of BD-rate over time for an RE module for the BlowingBubbles *sequence, after a transition from $Q_{i,1}$ to $Q_{i,2} = Q_{i,1} - \Delta Q_{i,1-2}$, while copying coding information from a CIC module with $Q_0 = Q_{i,2} + \Delta Q_{i,2}$. For negative values of $\Delta Q_{i,2}$, the first frames have a smaller BD-rate immediately after the change, whereas the opposite is true for positive values.*

$\Delta Q_{i,2} \rightarrow$ $\Delta Q_{i,1-2} \downarrow$	-3	-2	-1	1	2	3
-5	-5.3	-5.0	-4.7	-4.6	-4.6	-4.6
-4	-4.6	-4.2	-3.9	-3.9	-3.9	-3.9
-3	-4.3	-3.4	-3.1	-3.1	-3.2	-3.2
-2	-2.7	-3.1	-2.2	-2.2	-2.2	-2.2
-1	-1.4	-1.3	-1.9	-1.1	-1.1	-1.2
0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.5	1.5	1.4	0.4	1.2	1.2
2	3.0	3.0	2.8	2.3	1.7	2.4
3	4.6	4.6	4.4	3.8	3.6	3.0
4	6.0	6.0	5.9	5.2	5.0	4.9
5	7.5	7.4	7.2	6.7	6.4	6.4

Table 6.2: Additional BD-rate overhead when the bandwidth fluctuates from $Q_{i,1}$ to $Q_{i,2}$ while copying coding information from a CIC module with $Q_0 = Q_{i,2} + \Delta Q_{i,2}$ compared to a situation where the bandwidth does not fluctuate ($Q_{i,1} = Q_{i,2}$, meaning that $\Delta Q_{i,1-2} = 0$). When $\Delta Q_{i,1-2} < 0$, the overhead of the fluctuating scenario is smaller than in the case without fluctuation, whereas for $\Delta Q_{i,1-2} > 0$, the opposite is true.

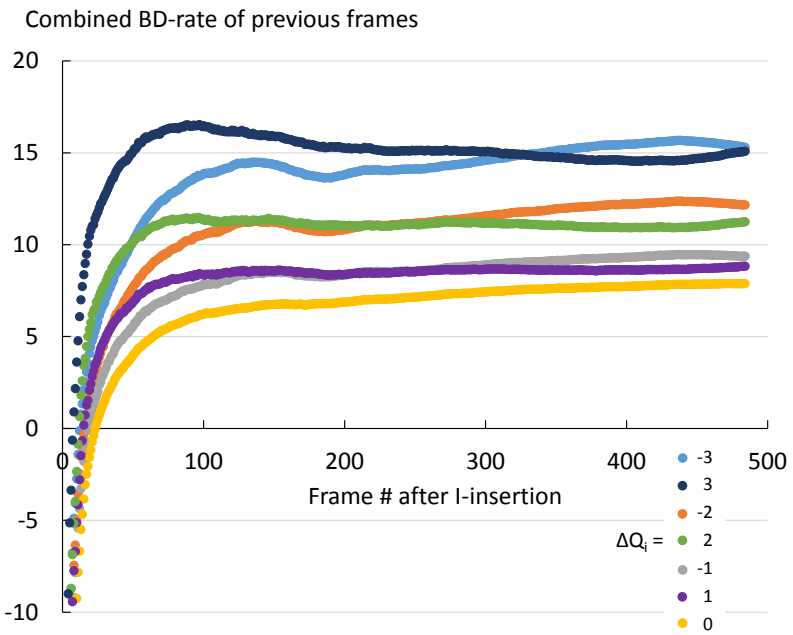


Figure 6.10: Evolution of BD-rate over time for an RE module for the BlowingBubbles *sequence, after inserting an I-frame to recover from packet loss. The first frames after the I-frame insertion have a smaller overhead compared to the rest of the frames.*

has a reference frame of higher quality than if said frame had been encoded as an inter-frame. As a result of this higher quality, the following frames appear to have a higher compression efficiency than the reference encoder[1]. Finally, by the end of the video, the effect of the intra frame has dissipated almost completely, with an extra BD-rate overhead between -0.6% and -0.7% compared to an RE module that did not experience packet loss and an intra-frame insertion (not shown in the figure).

As a conclusion, it can be said that bandwidth fluctuations have a positive effect on the bit rate overhead when going from a higher quality to a lower quality, while they have a negative effect when going from a lower quality to a higher quality. Since this negative effect is slightly stronger than the positive effect, it might be useful to transmit an intra-frame after many bandwidth fluctuations. Furthermore, this intra-frame can also be used to recover from packet losses. Finally, the exact effect of many consecutive fluctuations is left as future work.

[1] Note that only the BD-rate of the frames after the I-insertion have been taken into account, meaning that the bit rate of the intra-frame, which is typically much higher than an inter-coded version of the same frame, has not been taken into account. In practice, such a sudden burst in bit rate would be undesirable. Therefore, an intra-refresh would be used instead, meaning that only parts of the frame are coded with intra blocks in the first recovery frame, whereas other zones are encoded as intra in the following frames, until the entire picture has received an intra-refresh.

		Q_0	
	Q_i	Initial frames	Last 200 frames
Switch_set1	$\{25, 29, 33, 37\}$	$Q_i - 3$	$Q_i + 1$
Ref_set1	$\{25, 29, 33, 37\}$	$Q_i + 1$	$Q_i + 1$
Switch_set2	$\{23, 27, 31, 35\}$	$Q_i + 3$	$Q_i - 1$
Ref_set2	$\{23, 27, 31, 35\}$	$Q_i - 1$	$Q_i - 1$

Table 6.3: Configurations for testing the effect of switching the coding info source

6.4.4 Switching coding info source

As mentioned in Section 6.3, an RE module can switch the CIC module it copies coding information from. For example, this is the case if there are two CIC modules calculating coding information for a 20 Mbps and a 10 Mbps version of the video and the connection of a client linked to an RE module at 18 Mbps drops to 8 Mbps. While the connection is still 18 Mbps, the RE module will be copying coding information from the 20 Mbps CIC module. However, when the connection drops to 8 Mbps, the RE module should switch to copying coding information from the 10 Mbps CIC module instead.

In order to test the effect of switching the CIC module used as the coding info source, the system was tested using the two configuration sets shown in Table 6.3. In both configurations, five CIC modules were used with values of 22, 26, 30, 34, and 38 as Q_0. In the first configuration, four RE modules were active with Q_i values of 25, 29, 33, and 37. To simulate a switch of the coding information source, the RE modules used the CIC module for which $Q_0 = Q_i - 3$ as the source of coding information for the first part of the video. Then, for the last 200 frames of the video, the RE modules switched to using coding information provided by the CIC module for which $Q_0 = Q_i + 1$. This switch_set1 was compared to a ref_set1 for which the RE modules copied coding information from the CIC module with $Q_0 = Q_i + 1$ for the entire duration of the video. Similarly, the second configuration set first copies coding information from CIC modules with $Q_0 = Q_i + 3$ and then switches to $Q_0 = Q_i - 1$ while being compared to a reference set using a CIC module with $Q_0 = Q_i - 1$ for the entire duration of the video. In both configurations, the Q_0 after the switch is closer to Q_i than the initial Q_0, since a switch would only occur when the distance between Q_0 and Q_i becomes too large.

Note that since the last 200 frames were used, only videos with more than 200 frames (see Table 6.1) were considered. Also, temporal motion vector prediction was disabled in order to prevent dependencies between motion vectors before and after the switch from producing an invalid bitstream.

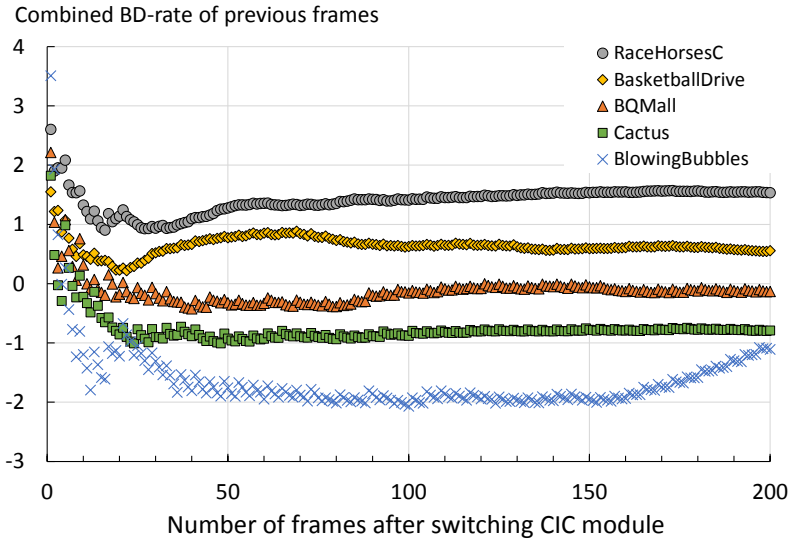


Figure 6.11: BD-rate of having an RE module switch from using coding information of a CIC module with $Q_0 = Q_i - 3$ to using coding information produced by a CIC module with $Q_0 = Q_i + 1$, compared to having an RE module use the coding information of the latter CIC module for the entire duration of the video.

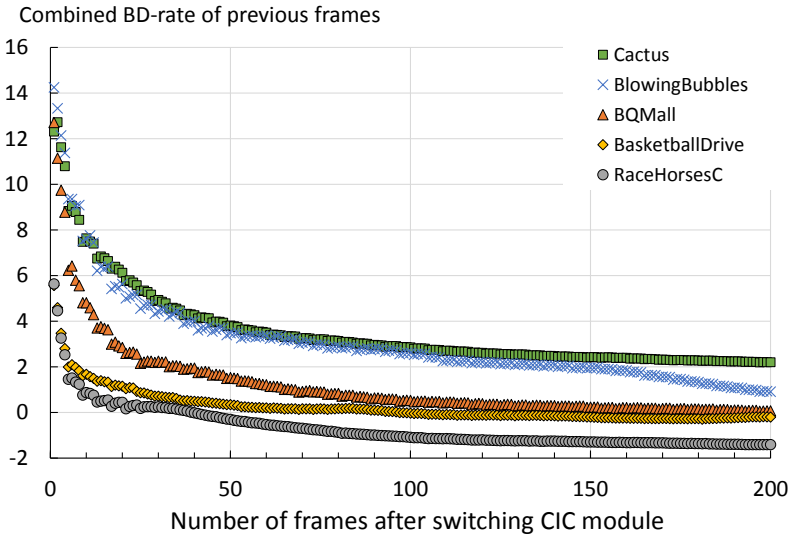


Figure 6.12: BD-rate of having an RE module switch from using coding information of a CIC module with $Q_0 = Q_i + 3$ to using coding information produced by a CIC module with $Q_0 = Q_i - 1$, compared to having an RE module use the coding information of the latter CIC module for the entire duration of the video.

The bit rate overhead for the same quality was calculated for the frames after the switch by comparing each switch_set to its corresponding ref_set. This overhead for n frames after the switch was calculated as the BD-rate between the ref_set and the switch_set by only considering the bit rate and PSNR of those n frames. A negative BD-rate indicates that the switch_set has a higher compression efficiency compared to the ref_set.

Figure 6.11 shows the effect of switching from a CIC module with a lower Q_0 to using coding information based on a higher Q_0. After about 50 frames, the overhead stabilizes for all tested video sequences. Moreover, the overhead remains between 2% and -2% for all sequences (including the ones not shown in the figure). Figure 6.12 then shows the effect of switching from a CIC module with a higher Q_0 to a lower Q_0. Again, after about 50 frames, the overhead stabilizes, with all tested videos achieving BD-rates below 4%.

Both test configurations show that switching the source of coding information mostly impacts the first frames after the switch. However, the impact becomes small after about 50 frames for most videos, showing that switching the CIC module linked to an RE module can be done with minimal impact on the compression efficiency of the system. Do note that, if compression efficiency is of great concern or if many switches are expected, it may be advisable to prohibit the use of skip-mode by CIC modules, since this mode signifies that no residual information should be encoded. As such, the RE modules will not be able to compensate for inefficient coding decisions, which will let errors propagate more easily after switching the source of coding information. In order to still be able to use possibly compression-efficient skip-mode decisions even when they are disabled for the CIC modules, the complexity of the RE modules could slightly be increased by having them check for skip-mode under certain conditions. However, further investigation of this matter is beyond the scope of this dissertation.

6.4.5 Comparison with state-of-the-art

As mentioned in Section 6.2, a possible solution for reducing the computational complexity of a real-time video-streaming system with many participants would be the use of fast encoding algorithms. However, as was also remarked in the same section, these algorithms only partially reduce the amount of coding decisions that a video encoder has to make. The RE modules in the proposed architecture, on the other hand, skip all coding decisions. In order to quantitatively compare the difference between using fast encoding algorithms and the proposed architecture with RE modules, this section offers a comparison in terms of compression efficiency and computational complexity.

Compression efficiency is measured as BD-rate, which is the average bit rate overhead of the fast encoding algorithm compared to a traditional non-accelerated

	BD-rate (%)	Time saving (%)	Speed-up
Xiong [20]	2.1	36.9	1.6
Kim [23]	0.6	49.5	2.0
Tan [24]	0.8	56.0	2.3
Shen [30]	1.0	47.1	1.9
Ahn [33]	1.0	37.7	1.6
Lee [31]	4.8	69.3	3.3
$\Delta Q_i = 1$	10.5	99.2	124.0
$\Delta Q_i = 2$	13.5	99.2	124.4
$\Delta Q_i = 3$	17.7	99.2	124.1

Table 6.4: Comparison with state-of-the-art

encoding. Computational complexity reduction is expressed as time saving (equation 2.1). Additionally, speed-up (equation 2.2) is also used as a measure in order to express the amount of fast encoders that are needed to reach the same computational complexity as a single non-accelerated encoder.

Six state-of-the-art fast encoding algorithms that reported results for a low-delay configuration were selected for the comparison [20, 23, 24, 30, 31, 33]. Since video sequences in class A, E, and F were not evaluated in all related work, only the results for class B, C, and D sequences were included in the average BD-rate and time saving presented in Table 6.4.

For the proposed architecture, three configurations were used. In each configuration, four non-accelerated encodings with quantization values of 22, 27, 32, and 37 were used as a reference. Similarly, four fast encodings were made by using the same four values as the Q_i of each RE module for each configuration. The difference between the three configurations is the $Q_0 = Q_i + \Delta Q_i$ (see also equation (6.3)) of the four CIC modules that were used in each configuration from which the coding information was copied.

As seen in Table 6.4, the proposed system results in higher bit rate overheads than the related work. However, the computational complexity of the RE modules is much lower than the fast encoding algorithms. For example, the speed-up of the RE modules is around 124, which means that the system can activate 124 RE modules before reaching the same computational complexity as using a single traditional encoder. On the other hand, the fastest algorithm of the related work by Lee [31] would already reach the same computational complexity as a non-accelerated encoder by using more than three fast encoders. As such, although the state-of-the-art performs better in terms of compression efficiency, only the proposed architecture can manage to significantly decrease the complexity for personalized bitstreams for low-delay live-streaming with a great number of participants.

6.5 Conclusion

This chapter proposed a system for low-delay live-streaming of video bitstreams with a bit rate tailored to the bandwidth capacity of individual users. This was achieved by using a combination of RE modules and CIC modules. The CIC modules calculate coding information at certain bit rates, while the RE modules copy this coding information to skip all encoding decisions except for residual encoding. The bit rate is thus adapted to the bandwidth by modifying the amount of encoded residual information.

The proposed system achieves its goal of providing each user with a bitstream at different bit rates at a very low computational complexity. Moreover, it was shown that the number of information calculation modules should be kept limited, since using more than six CIC modules with the HEVC video standard only results in negligible increases in compression efficiency. As a result, the proposed system using guided encoding can serve more than one hundred extra users for the same computational complexity as one extra user joining a similar system that does not use this set-up.

References

[1] I. Sodagar. *The MPEG-DASH Standard for Multimedia Streaming Over the Internet*. IEEE Multimedia, 18(4):62–67, Apr. 2011.

[2] J. De Praeter, H. Swimberghe, G. Renard, G. Van Wallendael, and P. Lambert. *Dynamic encoder profile optimisation for real-time video streaming applications*. Electron. Lett., 52(13):1116–1118, June 2016.

[3] R. Lanphier. *Standardizing Real-Time Streaming Protocols*. IEEE Computer, 31(7):94–96, July 1998.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. *Overview of the H.264/AVC video coding standard*. IEEE Trans. Circuits Syst. Video Technol., 13(7):560–576, July 2003.

[5] S. H. Ri, Y. Vatis, and J. Ostermann. *Fast Inter-Mode Decision in an H.264/AVC Encoder Using Mode and Lagrangian Cost Correlation*. IEEE Trans. Circuits Syst. Video Technol., 19(2):302–306, Feb. 2009.

[6] M. Paul, M. R. Frater, and J. F. Arnold. *An Efficient Mode Selection Prior to the Actual Encoding for H.264/AVC Encoder*. IEEE Trans. Multimedia, 11(4):581–588, June 2009.

[7] H. Zeng, C. Cai, and K. K. Ma. *Fast Mode Decision for H.264/AVC Based on Macroblock Motion Activity*. IEEE Trans. Circuits Syst. Video Technol., 19(4):491–499, Apr. 2009.

[8] Y. H. Sung and J. C. Wang. *Fast Mode Decision for H.264/AVC Based on Rate-Distortion Clustering*. IEEE Trans. Multimedia, 14(3):693–702, June 2012.

[9] J. Y. Lee and H. Park. *A Fast Mode Decision Method Based on Motion Cost and Intra Prediction Cost for H.264/AVC*. IEEE Trans. Circuits Syst. Video Technol., 22(3):393–402, Mar. 2012.

[10] X. Xu and Y. He. *Improvements on Fast Motion Estimation Strategy for H.264/AVC*. IEEE Trans. Circuits Syst. Video Technol., 18(3):285–293, Mar. 2008.

[11] Z. Liu, L. Li, Y. Song, S. Li, S. Goto, and T. Ikenaga. *Motion Feature and Hadamard Coefficient-Based Fast Multiple Reference Frame Motion Estimation for H.264*. IEEE Trans. Circuits Syst. Video Technol., 18(5):620–632, May 2008.

[12] Y. Ismail, J. B. McNeely, M. Shaaban, H. Mahmoud, and M. A. Bayoumi. *Fast Motion Estimation System Using Dynamic Models for H.264/AVC Video Coding*. IEEE Trans. Circuits Syst. Video Technol., 22(1):28–42, Jan. 2012.

[13] K. Bharanitharan, B. D. Liu, J. F. Yang, and W. C. Tsai. *A Low Complexity Detection of Discrete Cross Differences for Fast H.264/AVC Intra Prediction*. IEEE Trans. Multimedia, 10(7):1250–1260, Nov. 2008.

[14] B. G. Kim. *Fast Selective Intra-Mode Search Algorithm Based on Adaptive Thresholding Scheme for H.264/AVC Encoding*. IEEE Trans. Circuits Syst. Video Technol., 18(1):127–133, Jan. 2008.

[15] T. J. Kim, J. E. Hong, and J. W. Suh. *A fast intra mode skip decision algorithm based on adaptive motion vector map*. IEEE Trans. Consum. Electron., 55(1):179–184, Feb. 2009.

[16] D. Quan and Y. S. Ho. *Categorization for fast intra prediction mode decision in H.264/AVC*. IEEE Trans. Consum. Electron., 56(2):1049–1056, May 2010.

[17] C. Y. Wu and P. C. Su. *Fast Intra-Coding for H.264/AVC by Using Projection-Based Predicted Block Residuals*. IEEE Trans. Multimedia, 15(5):1083–1093, Aug. 2013.

[18] C. K. Chiang, W. H. Pan, C. Hwang, S. S. Zhuang, and S. H. Lai. *Fast H.264 Encoding Based on Statistical Learning*. IEEE Trans. Circuits Syst. Video Technol., 21(9):1304–1315, Sept. 2011.

[19] G.J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) Standard*. IEEE Trans. Circuits Syst. Video Technol., 22(12):1649–1668, Dec. 2012.

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

[21] X. Huang, Q. Zhang, X. Zhao, W. Zhang, Y. Zhang, and Y. Gan. *Fast inter-prediction mode decision algorithm for HEVC*. Signal, Image and Video Processing, pages 1–8, 2015.

[22] L. Shen, Z. Zhang, X. Zhang, P. An, and Z. Liu. *Fast TU size decision algorithm for HEVC encoders using Bayesian theorem detection*. Signal Processing: Image Communication, 32:121–128, 2015.

[23] H. S. Kim and R. H. Park. *Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule*. IEEE Trans. Circuits Syst. Video Technol., 26(1):130–138, Jan. 2016.

[24] H. L. Tan, C. C. Ko, and S. Rahardja. *Fast Coding Quad-Tree Decisions Using Prediction Residuals Statistics for High Efficiency Video Coding (HEVC)*. IEEE Trans. Broadcast., 62(1):128–133, Mar. 2016.

[25] M. R. Fini and F. Zargari. *Two stage fast mode decision algorithm for intra prediction in HEVC*. Multimedia Tools and Applications, pages 1–18, 2015.

[26] D. Ruiz, G. Fernandez-Escribano, J. L. Martinez, and P. Cuenca. *Fast intra mode decision algorithm based on texture orientation detection in HEVC*. Signal Processing: Image Communication, 44:12–28, 2016.

[27] S. H. Yang and K. S. Huang. *HEVC fast reference picture selection*. Electron. Lett., 51(25):2109–2111, Dec. 2015.

[28] H. Lee, H. J. Shim, Y. Park, and B. Jeon. *Early Skip Mode Decision for HEVC Encoder With Emphasis on Coding Quality*. IEEE Trans. Broadcast., 61(3):388–397, Sept 2015.

[29] T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert, and R. Van de Walle. *Real-time complexity constrained encoding*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 819–823, Sept. 2016.

[30] L. Shen, Z. Zhang, and Z. Liu. *Adaptive Inter-Mode Decision for HEVC Jointly Utilizing Inter-Level and Spatiotemporal Correlations*. IEEE Trans. Circuits Syst. Video Technol., 24(10):1709–1722, Oct. 2014.

[31] J. Lee, S. Kim, K. Lim, and S. Lee. *A Fast CU Size Decision Algorithm for HEVC*. IEEE Trans. Circuits Syst. Video Technol., 25(3):411–421, Mar. 2015.

[32] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz. *Fast HEVC Encoding Decisions Using Data Mining*. IEEE Trans. Circuits Syst. Video Technol., 25(4):660–673, Apr. 2015.

[33] S. Ahn, B. Lee, and M. Kim. *A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding*. IEEE Trans. Circuits Syst. Video Technol., 25(3):422–435, Mar. 2015.

[34] I. Zupancic, S. G. Blasi, E. Peixoto, and E. Izquierdo. *Inter-Prediction Optimizations for Video Coding Using Adaptive Coding Unit Visiting Order*. IEEE Trans. Multimedia, 18(9):1677–1690, Sept 2016.

[35] J. Zhang, B. Li, and H. Li. *An Efficient Fast Mode Decision Method for Inter Prediction in HEVC*. IEEE Trans. Circuits Syst. Video Technol., 26(8):1502–1515, Aug. 2016.

[36] 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 Trans. Multimedia, 18(3):364–378, Mar. 2016.

[37] J. De Praeter, A. J. Diaz-Honrubia, N. Van Kets, G. Van Wallendael, J. De Cock, P. Lambert, and R. Van de Walle. *Fast simultaneous video encoder for adaptive streaming*. In Proc. IEEE Int. Workshop Multimedia Signal Process. (MMSP), pages 1–6, Oct. 2015.

[38] D. Schroeder, P. Rehm, and E. Steinbach. *Block structure reuse for multi-rate high efficiency video coding*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 3972–3976, Sept. 2015.

[39] D. Schroeder, A. Ilangovan, M. Reisslein, and E. Steinbach. *Efficient multi-rate video encoding for HEVC-based adaptive HTTP streaming*. IEEE Trans. Circuits Syst. Video Technol., PP(99):1–1, 2016.

[40] G. Van Wallendael, J. De Cock, and R. Van de Walle. *Fast transcoding for video delivery by means of a control stream*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 733–736, Sept. 2012.

[41] T. Rusert, K. Andersson, R. Yu, and H. Nordgren. *Guided just-in-time transcoding for cloud-based video platforms*. In Proc. IEEE Int. Conf. Image Process. (ICIP), pages 1489–1493, Sept. 2016.

[42] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. Sullivan. *High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description Update 2*. Technical Report JCTVC-T1002, ITU-T Joint Collaborative Team on Video Coding (JCT-VC), Feb. 2015.

[43] G. Bjøntegaard. *Calculation of average PSNR differences between RD-curves*. Technical Report VCEG-M33, ITU-T Video Coding Experts Group (VCEG), Apr. 2001.

7
Overall Conclusion

When optimally adapting video to the device and network requirements of the receiver, each receiver requires an individual video encoder. However, doing so gradually evolves into a computationally complex nightmare as more users need to be served by this system. As shown in Chapter 2, even with state-of-the-art fast encoding algorithms, the computational complexity of HEVC encoders is only reduced by less than 75%. This means that, in the best case, four clients can be served with the same complexity as a non-accelerated encoder. Consequently, the computational complexity of the system remains too large to serve many users.

Contrary to fast encoding algorithms for a single encoder, providing a personalized video stream to each user has the distinct advantage of being able to use guided encoding by deriving coding decisions from earlier decisions made by other encoders. In this dissertation, the use of this guided encoding has thus been studied for four scenarios.

The first scenario considers the creation of a personalized video composition for each user. By rearranging videos into a different composition, the CTU-grid of the video in the composition might become misaligned with the CTU-grid of the original. Chapter 3 examines how coding information of such a spatially misaligned video can be predicted based on the coding information of the original video encoding. This is done using both a trivial method and a machine learning method. Using the trivial method, a complexity reduction of 82% is achieved with a bit rate overhead of 3% for a misalignment of 32 pixels. However, for other shifts, the performance of the machine learning method is better. As such, the proposed method succeeds in offering a fast-encoding solution for any amount of misalignment for a reasonable bit rate overhead.

Instead of combining video sequences into a composition, Chapter 4 considers a scenario in which a cropped view is extracted from a very-high-resolution video. In this chapter, it is observed that misalignment as with the previous scenario can also occur here. However, in order to allow higher complexity reductions, the cropped view is constrained to the CTU-grid of the full video. By doing so, more information from the full view can be reused, leading to complexity reductions between 96.5% and 97.5%, with bit rate overheads between 8% and 20%. Although the overhead appears to be large, it is compared to the overhead introduced by the traditional tile-based approached used in this scenario, and it is concluded that the overhead introduced by the guided encoding of personalized views is smaller than the overhead of using tiles. Moreover, since the personalized encoders do not suffer from the same structural latency as the tile-based method, the former could thus become a viable alternative for the latter.

Due to the nature of crops, not all coding information of the full view could be copied, since some coding information depended on other coding decisions from outside the cropped view. However, in a third scenario, this is not the case. In Chapter 5, a scenario is considered in which users receive a different version of the video depending on the dynamic-range capabilities of their device. In this scenario, the coding decisions of an encode of a low-dynamic-range version of a video are guided by the coding decisions taken by an encode of a high-dynamic-range version of the same video. Contrary to the cropped or shifted content in the previous scenarios, since both versions contain the same video content with only differences in pixel values, all coding information can be copied from one version to another. Moreover, by selecting the quantization parameter of the low-dynamic-range encoding according to the model described in this chapter, the correlation between both versions is maximized, which results in a smaller bit rate overhead. As such, a complexity reduction of 99.7% is achieved for a bit rate overhead of 12.4%. In other words, using guided encoding, a video provider can simultaneously encode multiple dynamic-range versions of the same video for about the same computational complexity as when encoding a single version.

Finally, after considering shifts, crops, and different dynamic ranges of the source video, a fourth scenario also considers a difference in encoding parameters. In this scenario in Chapter 6, each individual client connected to the system receives a video stream that is adapted to his current bandwidth capabilities in order to provide low-latency communications as required in real-time video-streaming events with many participants such as virtual classrooms and video conferences. In order to guide the encoding of these numerous personalized encoders, specialized modules calculate coding decisions at certain bit rates of the video. The guided encoders then only have to encode the residual picture of the video by copying all coding decisions generated by one of these modules. By creating a system with only six coding information calculation modules covering the desired bit rate

range, the system can provide individual encoders with a complexity reduction of 99.2% with the average bit rate overhead of the system being 11.8%. As a result, one hundred extra users joining the proposed system results in a smaller increase of overall computational complexity than one extra user joining a similar system that does not use guided encoding.

In conclusion, guided encoding is an effective way to greatly accelerate the encoding of personalized video streams for individual users, although the amount of coding information that can be shared efficiently depends on the scenario: spatial shifting and cropping operations require more intelligent decisions and algorithms, whereas small changes in pixel values and target bit rates more easily allow personalized encoders to completely share all information. In light of the latter scenarios, future standardization of video compression might want to take these findings into account by encoding coding decisions and the residual picture as separate parts of the bitstream in order to facilitate the use of such guided encoding.

Future work

As no research is ever truly finished, the work performed in this dissertation also serves as a foundation on which future research on guided encoding can build. Such research could focus in particular on expanding on the use case of streaming personalized views extracted from 360-degree video to users wearing VR-glasses. More specifically, the guided encoding techniques should be expanded to take dynamic views into account, in addition to the static views examined in Chapter 4. Additionally, the techniques should be applied to ultra-high resolution 360-degree video content for which content is currently becoming available in standardized test sets provided by the Joint Video Exploration Team (JVET), the successor to JCT-VC. Furthermore, the concept of CIC modules and RE modules from Chapter 6 can be combined with the streaming of 360-degree video in order to handle bandwidth fluctuations as well.

Finally, the best way to actually deploy a system for low-latency 360-degree video streaming should be investigated as well. One way to do this, inspired by the current draft requirements of the JVET related to network-distributed video coding, might be by actually sending a (near) lossless compressed version of the entire 360-degree video over a high-bandwidth backbone network together with coding information generated by CIC modules for the entire video. This coding information is then used in order to accelerate the encoding at the edge nodes where personalized views are extracted from the entire 360-degree video. These extracted views are then transmitted to the users at home over the last mile connection, which has a much lower bandwidth than the backbone network. By thus further expanding on the concepts of guided encoding as described in this dissertation, a highly immersive application such as sitting in your couch at home and being transported to a sports stadium by VR-glasses can become a reality.