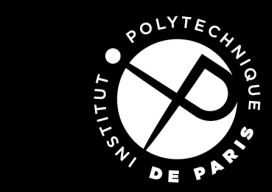
Nouvelles techniques de compression pour le codage vidéo prochaine-génération

Anthony Nasrallah



INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT043

Thèse de doctorat

Novel compression techniques for next-generation video coding

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris
École doctorale n°626 Ecole doctorale de l'Institut Polytechnique de Paris (ED IP Paris) (Sigle)
Spécialité de doctorat: Signal, Images, Automatique et Robotique

Thèse présentée et soutenue à Palaiseau, le 14 décembre 2021, par

Anthony NASRALLAH

Composition du Jury :

Daniel Ménard Professeur, INSA Rennes	Président
Aline Roumy Directrice de recherche, INRIA, Rennes	Rapporteur
Mounir Kaaniche Maître de conférences, Université Sorbonne Paris Nord	Rapporteur
Joumana Farah Professeur, Université Libanaise	Examinatrice
Mathias Wien Professeur, RWTH Aachen University	Examinateur
Marco Cagnazzo Directeur de recherche, Télécom Paris	Directeur de thèse
Thomas Guionnet Ingénieur de recherche, Ateme	Co-directeur de thèse
Attilio Fiandrotti Maître de conférence, Université de Turin	Invité
Mohsen Abdoli Ingénieur de recherche, Ateme	Invité

Titre : Nouvelles techniques de compression pour le codage vidéo prochaine-génération.
Mots clés : Codage Vidéo, VVC, Aprentissage Profond, Mode Intra, Filtres d'interpolation, optimisation d'encodeur.

Résumé : Le contenu vidéo occupe aujourd'hui environ 82% du trafic Internet mondial. Ce pourcentage important est dû à la révolution de la consommation des contenus vidéo. D'autre part, le marché exige de plus en plus de vidéos avec des résolutions et des qualités plus élevées. De ce fait, développer des algorithmes de codage encore plus efficaces que ceux existants devient une nécessité afin de limiter l'augmentation de la quantité de données vidéo circulant sur internet et assurer une meilleure qualité de service. En outre, la consommation impressionnante de contenu multimédia dans les produits électroniques a un impact écologique. Par conséquent, trouver un compromis entre la complexité des algorithmes et l'efficacité des implémentations s'impose comme un nouveau défi. Pour cela, une équipe collaborative a été créée dans le but de développer une nouvelle norme de codage vidéo, Versatile Video Coding – VVC/H.266. Bien que VVC ait pu aboutir à une réduction de plus de 40% du débit par rapport à HEVC, cela ne signifie pas du tout qu'il n'y a plus besoin d'améliorer encore l'efficacité du codage. De plus, VVC ajoute une complexité remarquable par rapport à HEVC.
Cette thèse vient répondre à ces problématiques en proposant trois nouvelles méthodes d'encodage. Les apports de cette recherche se répartissent en deux axes principaux. Le premier axe consiste à proposer et mettre en œuvre de nouveaux outils de compression dans la nouvelle norme, capables de générer des gains de codage supplémentaires. Deux méthodes ont été proposées pour ce premier axe. Le point commun entre ces deux méthodes est la dérivation des informations de prédiction du côté du décodeur. En effet, l'augmentation des choix de l'encodeur peut améliorer la précision des prédictions et donne moins de résidus d'énergie, conduisant à une réduction du débit. Néanmoins, plus de modes de prédiction impliquent plus de signalisation à envoyer dans le flux binaire pour informer le décodeur des choix qui ont été faits au niveau de l'encodeur. Les gains mentionnés ci-dessus sont donc largement compensés par la signalisation ajoutée. Si l'information de prédiction est dérivée au nivau du décodeur, ce dernier n'est plus passif, mais devient actif, c'est le concept de décodeur intelligent. Ainsi, il sera inutile de signaler l'information, d'où un gain en signalisation. Chacune des deux méthodes propose une technique intelligente différente pour prédire l'information au niveau du décodeur. La première technique construit un histogramme de gradients pour déduire différents modes de prédiction intra pouvant ensuite être combinés, pour obtenir le mode de prédiction intra final pour un bloc donné. Cette propriété de fusion permet de prédire plus précisément les zones avec des textures complexes, ce qui, dans les schémas de codage conventionnels, nécessiterait plutôt un partitionnement et/ou une transmission plus fine des résidus à haute énergie. La deuxième technique consiste à donner à VVC la possibilité de basculer entre différents filtres d'interpolation pour la prédiction inter. La déduction du filtre optimal sélectionné par l'encodeur est réalisée grâce à des réseaux de neurones convolutifs.
Le deuxième axe, contrairement au premier, ne cherche pas à ajouter une contribution à l'algorithme de base de VVC. Cet axe vise plutôt à permettre une utilisation optimisée de l'algorithme déjà existant. L'objectif ultime est de trouver le meilleur compromis possible entre l'efficacité de compression fournie et la complexité imposée par les outils VVC. Ainsi, un système d'optimisation est conçu pour déterminer une technique efficace d'adaptation de l'activation des outils au contenu. La détermination de ces outils peut être effectuée soit en utilisant des réseaux de neurones artificiels, soit sans aucune technique d'intelligence artificielle.

Institut Polytechnique de Paris
91120 Palaiseau, France

Title: Novel compression techniques for next-generation video coding.

Keywords: Video Coding, VVC, Deep Learning, Intra Mode, Interpolation Filters, Encoder optimization.

Abstract: Video content now occupies about 82% of global internet traffic. This large percentage is due to the revolution in video content consumption. On the other hand, the market is increasingly demanding videos with higher resolutions and qualities. This causes a significant increase in the amount of data to be transmitted. Hence the need to develop video coding algorithms even more efficient than existing ones to limit the increase in the rate of data transmission and ensure a better quality of service. In addition, the impressive consumption of multimedia content in electronic products has an ecological impact. Therefore, finding a compromise between the complexity of algorithms and the efficiency of implementations is a new challenge. As a result, a collaborative team was created with the aim of developing a new video coding standard, Versatile Video Coding – VVC/H.266. Although VVC was able to achieve a more than 40% reduction in throughput compared to HEVC, this does not mean at all that there is no longer a need to further improve coding efficiency. In addition, VVC adds remarkable complexity compared to HEVC.

This thesis responds to these problems by proposing three new encoding methods. The contributions of this research are divided into two main axes. The first axis is to propose and implement new compression tools in the new standard, capable of generating additional coding gains. Two methods have been proposed for this first axis. These two methods rely on the derivation of prediction information at the decoder side. This is because increasing encoder choices can improve the accuracy of predictions and yield less energy residue, leading to a reduction in bit rate. Nevertheless, more prediction modes involve more signaling to be sent into the binary stream to inform the decoder of the choices that have been made at the encoder. The gains mentioned above are therefore more than offset by the added signaling. If the prediction information has been derived from the decoder, the latter is no longer passive, but becomes active hence the concept of intelligent decoder. Thus, it will be useless to signal the information, hence a gain in signalization. Each of the two methods offers a different intelligent technique than the other to predict information at the decoder level. The first technique constructs a histogram of gradients to deduce different intra-prediction modes that can then be combined by means of prediction fusion, to obtain the final intra-prediction for a given block. This fusion property makes it possible to more accurately predict areas with complex textures, which, in conventional coding schemes, would rather require partitioning and/or finer transmission of high-energy residues. The second technique gives VVC the ability to switch between different interpolation filters of the inter prediction. The deduction of the optimal filter selected by the encoder is achieved through convolutional neural networks.

The second axis, unlike the first, does not seek to add a contribution to the VVC algorithm. This axis rather aims to build an optimized use of the already existing algorithm. The ultimate goal is to find the best possible compromise between the compression efficiency delivered and the complexity imposed by VVC tools. Thus, an optimization system is designed to determine an effective technique for activating the new coding tools. The determination of these tools can be done either using artificial neural networks or without any artificial intelligence technique.

Institut Polytechnique de Paris
91120 Palaiseau, France

Acknowledgment

I would like to express my gratitude to my supervisors, **Marco Cagnazzo** and **Attilio Fiandrotti**, who accompanied me through this adventure. Beside their precious scientific input, their continuous support gave me the confidence I needed to develop the researcher in me, while also leaving me ample space for developing my own ideas.

I would also like to express my appreciation for **Thomas Guionnet**, research engineer at ATEME. His pragmatism and his constructive criticism during his supervision on my work have contributed to the success of this thesis. I would also like to thank **Mohsen Abdoli** and **Thibaud Biatek** from the Research and Innovation team at ATEME for sharing their experience and their knowledge, for their advice and their constant follow-up of the work progress. Without forgetting to thank my manager **Mickael Raulet** for ensuring the environment and the necessary conditions for the success of this thesis.

I also wish to thank **Aline Roumy** and **Mounir Kaaniche** for reviewing my Ph.D. thesis, as well as for their time and for being part of my Ph.D. jury. **Joumana Farah**, **Mathias Wien** and **Daniel Ménard**, accepting our invitation to participate in the jury was greatly appreciated. I hope that you will enjoy reading this manuscript.

My deepest appreciation goes to Imène for reading my manuscript.

Completing a thesis between the academic and industrial environment allowed me to acquire this balance between technical skills and analytical skills. I have also learned many things that helped me broaden my horizons and motivation. For this, I am deeply grateful to my colleagues in ATEME and Télécom Paris. In particular, I would like to thank the permanent staff in Télécom Paris: **Florence Besnard** and **Delphine Laude** for their help in the administrative procedures. At ATEME side, I wish to acknowledge the collegues from Research and Innovation team, and from Video Processing team. Especially, **Mathieu**, **Sassan**, **Marwa**, **Sebastien**, **Josselin**, **AbdelMajid** and **Anne-Lyse**, who contributed in stimulating suggestions and encouragement.

Special thoughts for my co-author in some of my publications, patents and contributions, **Elie Mora**. Despite the brief period we spent together in this journey, he left a mark on the person I am today. Obtaining a Ph.D. at the frontier of video coding and artificial intelligence was a long-term ambition that evolved over the years. For this, I am grateful to a special professor, **Béatrice Pesquet-Popescu** who let me discover a new domain in the best way ever.

Special thanks go to my friends, whether in France or in Lebanon, for their support, their motivation, their exchanges, their laughter and their confidence. Their pride yields my perseverance. The Ph.D journey is also a human experience, and it would never have been completed without them.

I am also indebted to my aunt Madeleine, who hosted me during the last years. Last but not least, I would like to thank my family starting with my parents Yvonne and Joe, and my brother Yves. Their unreserved love and support has always been a source of motivation, so I thank them from the bottom of my heart.

Many thanks to you all, and happy reading!

Contents

Abstract (French) i

Abstract ii

Acknowledgment iii

Acronyms ix

List of Figures xiii

List of Tables xviii

1 Résumé (French) 3
1 Introduction . . . 3
2 Contributions . . . 4
2.1 Premier axe : Proposer de nouveaux outils de codage . . . 4
2.1.1 **Première contribution d'outil de compression : Approche conventionnelle** . . . 5
2.1.2 **Deuxième contribution d'outil de compression : Approche basée sur l'apprentissage profond** . . . 5
2.2 Deuxième axe : Optimisation des outils existants . . . 6
3 Structure du manuscrit . . . 6
3.1 Partie I . . . 6
3.2 Partie II . . . 6
3.3 Partie III . . . 7
4 Résultats et conclusions . . . 7
4.1 Première contribution : DIMD . . . 7
4.1.1 Principe de fonctionnement . . . 7
4.1.2 Résultats . . . 8
4.1.3 Conclusions . . . 9
4.2 Deuxième contribution : C-SSIF . . . 9
4.2.1 Principe de fonctionnement . . . 9
4.2.2 Résultats . . . 10
4.2.3 Conclusions . . . 10
4.3 Troisième contribution : Optimized Content-Adaptive Activation of VVC tools . . . 11
4.3.1 Principe de fonctionnement . . . 11
4.3.2 Validations expérimentales . . . 13
4.3.3 Conclusions . . . 14

2 Introduction **15**
2.1 Context . . . 15
2.2 Contributions . . . 16
2.2.1 First axis: proposing new coding tools . . . 16
2.2.1.1 First compression tool contribution: Conventional Approach . . . 17
2.2.1.2 Second compression tool contribution: Deep Learning-based Approach . . . 17
2.2.2 Second Axis: Optimizing existing tools . . . 18
2.3 Structure of the manuscript . . . 18

3 Background **21**
3.1 General Overview on Video Coding Standards . . . 21
3.1.1 Block Partitioning . . . 22
3.1.2 Intra-picture prediction . . . 22
3.1.3 Motion-compensated or inter-picture prediction . . . 23
3.1.4 Transformation . . . 23
3.1.5 Quantization . . . 24
3.1.6 Entropy Coding . . . 24
3.1.7 In-loop filtering . . . 25
3.2 Versatile Video Coding (VVC) . . . 25
3.2.1 Standardization and Developpment . . . 26
3.2.1.1 Exploration Phase (2015–2017) . . . 26
3.2.1.2 Standardization Phase (2018–2020) . . . 27
3.2.2 Coding Tools . . . 27
3.2.2.1 Block Partitioning . . . 27
3.2.2.2 Inter-Picture Prediction . . . 30
3.2.2.3 Intra-Picture Prediction . . . 32
3.2.2.4 Transforms and Quantization . . . 33
3.2.2.5 Entropy Coding . . . 33
3.2.2.6 In-Loop Filtering . . . 34
3.2.2.7 Screen Content Coding Tools . . . 35
3.2.2.8 360° Video Coding Tools . . . 36
3.2.3 Systems and Transport Interfaces . . . 37
3.2.3.1 Random Access Support . . . 37
3.2.3.2 Adaptation Parameter Set (APS) . . . 38
3.2.3.3 Picture Header . . . 38
3.2.3.4 Reference Picture Management . . . 38
3.2.3.5 High-Level Picture Partitioning . . . 39
3.2.3.6 Picture Resolution Changes With Inter-Picture Prediction . . . 43
3.2.3.7 Scalability Support . . . 44
3.2.4 Performance . . . 45
3.2.4.1 Objective evaluation . . . 45
3.2.4.2 Subjective evaluation . . . 47
3.3 Basics of Machine Learning (ML) . . . 49
3.3.1 Supervised Learning . . . 49
3.3.2 Unsupervised Learning . . . 51

3.3.3 Deep Learning . . . 53

4 State of the Art **57**
4.1 Decoder-side intra mode derivation techniques . . . 57
4.2 Interpolation Filtering . . . 58
4.2.1 Non-CNN based switching interpolation filters . . . 59
4.2.2 Related Deep Learning works . . . 60
4.3 Encoder optimization . . . 60
4.3.1 Related works . . . 60
4.3.2 Presentation of VVenc . . . 62
4.3.2.1 Main features . . . 63
4.3.2.2 Results . . . 63

5 Decoder-side Intra Mode Derivation (DIMD) **65**
5.1 The problem . . . 65
5.2 Motivation . . . 65
5.2.1 Experimental settings . . . 65
5.2.2 The motivation of the work . . . 65
5.3 Overview of the method . . . 67
5.4 Description of the Implicit Intra Mode Derivation . . . 68
5.5 Prediction Fusion . . . 69
5.5.1 Fusion modes . . . 71
5.5.2 Selected fusion configuration . . . 72
5.6 Experimental Results . . . 73
5.6.1 Implementation . . . 73
5.6.2 Statistics . . . 73
5.6.3 Performance . . . 74
5.6.3.1 Coding efficiency . . . 74
5.6.3.2 Complexity . . . 74

6 CNN-based Switchable Sub-pel Interpolation Filters (C-SSIF) **79**
6.1 The problem . . . 79
6.2 Motivation . . . 79
6.3 Overview of the method . . . 80
6.4 Sub-pel Interpolation filters Design . . . 80
6.4.1 Filter design . . . 80
6.4.1.1 Kaiser Window . . . 80
6.4.1.2 Filter coefficients generation . . . 81
6.4.2 Rationale behind the choices made . . . 82
6.4.3 Different implementations of SSIF . . . 82
6.5 Preliminary filter evaluation . . . 84
6.6 CNN-based Interpolation Filters Prediction . . . 85
6.6.1 General Design Considerations . . . 86
6.6.2 Neural network topology . . . 87
6.6.3 Training set generation . . . 88

6.6.4 Training procedure . . . 89
6.7 Experimental Results . . . 89
6.7.1 Experimental setup . . . 89
6.7.2 Architecture evaluation . . . 89
6.7.3 Performance evaluation . . . 91
6.7.4 Statistical analysis . . . 91
6.7.4.1 The Motion . . . 93
6.7.4.2 The AutoSAD . . . 95
6.7.4.3 The QP value . . . 95
6.7.4.4 The block sizes . . . 95
6.7.4.5 Interpretation of the conducted analysis . . . 95
6.8 Future works . . . 97

7 Optimization of the new Adopted VVC Tools 101
7.1 VVC Tools Description . . . 101
7.1.1 Intra Prediction . . . 102
7.1.1.1 Cross-Component Linear Model prediction (CCLM) . . . 102
7.1.1.2 Position Dependent intra Prediction Combination (PDPC) . . . 103
7.1.1.3 Multiple Reference Line (MRL) intra prediction . . . 104
7.1.1.4 Intra Sub-Partitions (ISP) . . . 105
7.1.1.5 Matrix weighted Intra Prediction (MIP) . . . 108
7.1.2 Inter Prediction . . . 109
7.1.2.1 Affine motion compensated prediction . . . 111
7.1.2.2 Prediction refinement with optical flow for affine mode (PROF) . . . 114
7.1.2.3 Sub-block-based Temporal Motion Vector Prediction (SbTMVP) . . . 117
7.1.2.4 Adaptive Motion Vector Resolution (AMVR) . . . 118
7.1.2.5 Bi-prediction with CU-level Weight (BCW) . . . 120
7.1.2.6 Merge mode with MVD (MMVD) . . . 122
7.1.2.7 Symmetric MVD coding (SMVD) . . . 123
7.1.2.8 Combined Inter and Intra Prediction (CIIP) . . . 126
7.1.2.9 Bi-Directional Optical Flow (BDOF) . . . 126
7.1.2.10 Decoder side Motion Vector Refinement (DMVR) . . . 129
7.1.2.11 Triangle Partition Mode for inter prediction (TPM) . . . 133
7.1.2.12 Motion field storage . . . 135
7.1.2.13 Miscellaneous inter prediction aspects . . . 135
7.1.3 Transform and Quantization . . . 135
7.1.3.1 Sub-Block Transform (SBT) . . . 135
7.1.3.2 Multiple Transform Selection (MTS) for core transform . . . 136
7.1.3.3 Low-Frequency Non-Separable Transform (LFNST) . . . 139
7.1.3.4 Dependent quantization . . . 142
7.1.3.5 Joint Coding of Chroma Residuals (JCCR) . . . 144
7.1.4 In-Loop Filters . . . 145
7.1.4.1 Adaptive Loop Filter (ALF) . . . 145
7.1.4.2 Luma Mapping with Chroma Scaling (LMCS) . . . 146
7.2 Optimized Content-Adaptive Activation of VVC tools . . . 152

7.2.1 The problem . . . 152
7.2.2 Motivation . . . 154
7.2.3 Overview of the method . . . 155
7.2.4 Preliminary framework . . . 155
7.2.4.1 The used metric: Efficiency Ratio . . . 155
7.2.4.2 Tools ordering towards an optimized encoder . . . 156
7.2.4.3 Experimental validation . . . 163
7.2.5 Generalized framework . . . 167
7.2.5.1 Simplified generalized framework . . . 172
7.2.5.2 Future work . . . 174

8 Conclusion **177**
8.1 Thesis objectives . . . 177
8.2 Summary of the contributions . . . 177
8.2.1 First contribution: DIMD . . . 177
8.2.1.1 Recap of the method . . . 177
8.2.1.2 Results . . . 178
8.2.1.3 Conclusions . . . 178
8.2.2 Second contribution: C-SSIF . . . 178
8.2.2.1 Summary of the method . . . 178
8.2.2.2 Results . . . 179
8.2.2.3 Conclusions . . . 179
8.2.3 Third contribution: Optimized Content-Adaptive Activation of VVC tools . . . 180
8.2.3.1 Recap of the method . . . 180
8.2.3.2 Experimental validations . . . 180
8.2.3.3 Conclusions . . . 180

9 Publications **183**
9.1 Conferences papers . . . 183
9.2 Journal paper . . . 183
9.3 Webinars . . . 183
9.4 Contributions to the MPEG/JVET standard: H.266/VVC . . . 183
9.5 Patents . . . 184

References **184**

List of Abbreviations

ABT	**A**symmetric **B**inary **T**ree
ACT	**A**daptive **C**olor **T**ransform
AHG	**A**d-**H**oc **G**roup
AI	**A**ll **I**ntra
AI	**A**ritificial **I**ntelligence
AIF	non separable **A**daptive **I**nterpolation **F**ilter
ALF	**A**daptive **L**oop **F**ilter
ANN	**A**rtificial **N**eural **N**etworks
APS	**A**daptation **P**arameter **S**et
AV2	**A**OMedia **V**ideo **2**
AVC	**A**dvanced **V**ideo **C**oding
BCW	**B**i-prediction with **C**U-level **W**eight
BD	**B**jontegaard **D**elta
BDOF	**B**i-**D**irectional **O**ptical **F**low
BDPCM	**B**lock-level **D**ifferential **P**ulse **C**ode **M**odulation
BDR	**B**jontegaard-**D**elta **R**ate
BD-rate	**B**jontegaard **D**elta bit rate
BIO	**BI**-Directional **O**ptical Flow
BLA	**B**roken **L**ink **A**ccess
BM	**B**ilateral-**M**atching
bS	**b**order Filter **S**trength
BT	**B**inary **T**ree
CABAC	**C**ontext-**A**daptive **B**inary **A**rithmetic **C**oding
CAVLC	**C**ontext **A**daptive **VLC**
CB	**C**oding **B**lock
CC-ALF	**C**ross **C**omponent **ALF**
CCLM	**C**ross-**C**omponent **L**inear **M**odel
CfP	**C**all **f**or **P**roposal
CGs	**C**oefficient **G**roup**s**
CIIP	**C**ombined **I**nter and **I**ntra **P**rediction
CMP	**C**ube **M**ap **P**rojection
CNNs	**C**onvolutional **N**eural **N**etwork**s**
CP	**C**ontrol **P**oints
CPMV	**C**ontrol **P**oints **MV**
CRA	**C**lean **R**andom **A**ccess
C-SSIF	**C**NN-based **S**witchable **S**ub-pel **I**nterpolation **F**ilters
CST	**C**hroma **S**eparate **T**ree
CTC	**C**ommon **T**est **C**onditions
CTU	**C**oding **T**ree **U**nit
CU	**C**oding **U**nit
CVS	**C**oded **V**ideo **S**equence
DCT	**D**iscrete **C**osine **T**ransform
DCT-IF	**DCT**-based **I**nterpolation **F**ilter
DIMD	**D**ecoder-side **I**ntra **M**ode **D**erivation
DL	**D**eep **L**earning
DMVR	**D**ecoder **M**otion **V**ector **R**efinement
DPB	**D**ecoded **P**icture **B**uffer
DQ	**D**ependent **Q**uantization
dQP	**d**elta **QP**
DT	**D**ecoder run-**T**ime
EAIF	**E**nhanced **A**daptive **I**nterpolation **F**ilter
ERP	**E**quirectangular **P**rojection **F**ormat
ET	**E**ncoder run-**T**ime

EVC	**E**ssential **V**ideo **C**oding
FCN	**F**ully **C**onnected **N**etwork
FIFO	**F**irst-**I**n-**F**irst-**O**ut
FIR	**F**inite **I**mpulse **R**esponse
FRUC	**F**rame **R**ate **U**p **C**onversion
GDR	**G**radual
Decoding	**R**efresh
GOP	**G**roup **O**f **P**ictures
GPM	**G**eometric **P**artitioning **M**ode
HDR	**H**igh **D**ynamic **R**ange
HM	**H**EVC **M**odel
HMVP	**H**istory-based **MV P**rediction
HoG	**H**istogram **o**f **G**radients
IBC	**I**ntra-picture **B**lock **C**opy
IDR	**I**nstantaneous **D**ecoder **R**efresh
IF	**I**nterpolation **F**ilter
IP	**I**nternet **P**rotocol
IRAP	**I**ntra **R**andom **A**ccess **P**oint
ISP	**I**ntra **S**ub-**P**artitions
JCCR	**J**oint **C**oding of **C**hroma **R**esiduals
JEM	**J**oint **E**xploitation **M**odel
JVET	**J**oint **V**ideo **E**xperts **T**eam
KCS	**K**aiser **C**onvoluted with **S**inc
KCS-FT	**KCS-F**lat**T**op
KCS-G	**KCS-G**auss
KLT	**K**arhunen-**L**oève **T**ransform
k-NN	**k-N**earest **N**eighbors
KTA	**K**ey **T**echnical **A**rea
LDB	**L**ow **D**elay **B**
LFNST	**L**ow-**F**requency **N**on-**S**eparable **T**ransform
LMCS	**L**uma **M**apping with **C**hroma **S**caling
LUT	**L**ook-**U**p **T**ables
Luma	**L**uminance
MC	**M**otion **C**ompensation
MDLM	**M**ulti-**D**irectional **LM**
ME	**M**otion **E**stimation
MIP	**M**atrix-based **I**ntra-picture **P**rediction
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
MMVD	**M**erge **M**ode with **MVD**
MOS	**M**ean **O**pinion **S**core
MPEG	**M**oving **P**icture **E**xperts **G**roup
MRL	**M**ultiple **R**eference **L**ine
MTS	**M**ultiple **T**ransform **S**election
MTT	**M**ulti-**T**ype **T**ree
MV	**M**otion **V**ector
MVD	**M**otion **V**ector **D**ifferences
NAL	**N**etwork **A**bstraction **L**ayer
PB	**P**rediction **B**lock
PDPC	**P**osition **D**ependent intra **P**rediction **C**ombination
PH	**P**icture **H**eader
PPS	**P**icture **P**arameter **S**et
PROF	**P**red **R**efinement with **O**ptical **F**low
PSNR	**P**eak **S**ignal-to-**N**oise **R**atio
PTL	**P**rofile **T**ier **L**evel
PU	**P**rediction **U**nit
QP	**Q**uantization **P**arameter
QPA	**Q**uantization **P**arameter **A**djustments
QT	**Q**uad-**T**ree
RA	**R**andom **A**ccess
RDO	**R**ate **D**istortion **O**ptimization
RDPCM	**R**esidual **D**ifferencial **P**ulse **C**ode **M**odulation
RF	**R**andom **F**orest
ROI	**R**egion-**O**f-**I**nterest

RPR	**R**eference **P**icture **R**esampling
RST	**R**educed **S**eparable **T**ransform
SAD	**S**um of **A**bsolute **D**ifferences
SAIF	**S**eparable **A**daptive **I**nterpolation **F**ilter
SAO	**S**ample **A**daptive **O**ffset
SBT	**S**ub-**B**lock **T**ransform
SDH	**S**ign **D**ata **H**iding
SDR	**S**tandard **D**ynamic **R**ange
SIFO	**S**witched **I**nterpolation **F**ilter with **O**ffset
SMVD	**S**ymmetric **M**otion **V**ector **D**ifference
SPS	**S**equence **P**arameter **S**et
SSD	**S**um-of-**S**quared **D**ifferences
SSIF	**S**ub-Pel **S**witchable **I**nterpolation **F**ilters
SVM	**S**upport **V**ector **M**achine
TM	**T**emplate **M**atching
TMVP	**T**emporal **M**otion **V**ector **P**rediction
QP	**Q**uantization **P**arameter
TPM	**T**riangle **P**artition **M**ode
TS	**T**ransform **S**kip
TSRC	**T**ransform **S**kip **R**esidual **C**oding
VCEG	**V**ideo **C**oding **E**xperts **G**roup
VLC	**V**ariable-**L**ength **C**oding
VOD	**V**ideo-**O**n-**D**emand
VPDUs	**V**irtual **P**ipeline **D**ata **U**nits
VPS	**V**ideo **P**arameter **S**et
VTM	**V**VC **T**est **M**odel
VVC	**V**ersatile **V**ideo **C**oding
VVenC	**V**ersatile **V**ideo **enC**oder
WP	**W**eighted **P**rediction
XPSNR	Weighted e**X**tended **PSNR**

List of Figures

1.1 Première étape : processus d'entraînement. 11
1.2 Comparaison des performances des trois tests. 12
1.3 Comparaison des performances des trois tests pour chaque séquence. 13

3.1 Block diagram of a hybrid video encoder, including the modeling of the decoder within the encoder [1]. 22
3.2 Motion Compensation [2]. 23
3.3 Example of Quad-Tree with nested multitype tree coding block structure. 29
3.4 Comparison between partitionings of AVC, HEVC and VVC. 29
3.5 Disallowed ternary splitting and binary splitting in VVC when the luma coding block width or height is 128 to enable 64×64 VPDU operation. 30
3.6 Wide-angular intra prediction [3]. 32
3.7 Picture with 18×12 luma CTUs that are partitioned into 24 tiles and nine rectangular slices. 39
3.8 Picture partitioned into four tiles and four rectangular slices (note that the top-right tile is split into two rectangular slices). 40
3.9 Picture with 18×12 luma CTUs that are partitioned into 12 tiles and three raster-scan slices. 41
3.10 Picture partitioned into 18 tiles, 24 slices, and 24 subpictures. 41
3.11 Sub-picture-based viewport-dependent 360° video delivery scheme. 42
3.12 Sub-picture-based viewport-dependent 360° video delivery scheme making use of inter-layer prediction. 43
3.13 Rate-distortion plots of VVC, HEVC, and AVC for the CatRobot1 video test sequence (random access configuration) [1]. 46
3.14 Average (arithmetic) MOS and (geometric mean) bit rates of VVC (VTM and VVenC encoders) and HEVC (HM encoder) pooled over the five UHD SDR sequences used in the verification test [1]. 48
3.15 A simple Linear regression example with one feature/variable. 50
3.16 Sigmoid curve having a bound between 0 and 1. 50
3.17 A simple k -NN model for different values of k. 51
3.18 The K-means and the cluster centroids. 52
3.19 Uniform quantization of 2-dimensional Data. 52
3.20 Vector quantization of 2-dimensional Data. 53
3.21 Artificial Neural Network with four hidden layers. 54
3.22 A CNN with 3 convolutional, 2 subsampling layers. 54

4.1 Introducing 4-pel precision to improve accuracy of MCP. 58
4.2 Runtime and BD rate VVenC 1.0.0, x265 3.4, aomenc 3.0 and VTM 12.0 compared to HM 16.22. All codecs but VTM and HM are run multithreaded with 8 threads [4]. 64

5.1 The proposed design of the VVC decoder, integrated with DIMD. The parsing phase is modified to incorporate the use of the DIMD flag. In case of DIMD blocks, a different reconstruction path is taken which includes the intra mode derivation and multiple prediction fusion steps. 69
5.2 Angular IPMs of VVC (i.e., IPM #2 to IPM #66): a) Covering the range of 180 degrees in four quarters (i.e., Q_1 to Q_4); b) Uneven angle distribution in the second quarter (i.e., the range between IPM #18 to IPM #34). 70
5.3 A 32×32 block example from BasketballDrive. The major texture directions can be visually detected. As a consequence, the calculated HoG from each block shows two IPMs that clearly correspond to the dominant directions of the texture in the image. 70
5.4 Correlation between content texture and implicitly derived angular IPMs using DIMD. . . 71
5.5 Spatial distribution of DIMD blocks in different sequences of Tango (3840×2160) and RitualDance (1920×1080). 74

6.1 Impulse response of sinc function in time domain for different configurations. 81
6.2 Frequency responses of the DCT-IF filter, the Gauss and FlatTop filters in [5] and our proposed KCS-Gauss and KCS-FlatTop counterparts for half-pel (left) and quarter-pel (right) precision. 84
6.3 Distribution of the DCT-IF, KCS-Gauss and KCS-FlatTop filters selection as a function of the block size. 86
6.4 The convolutional architecture used to predict the optimal interpolation filter: it includes three convolutional layers and size fully connected layers for a total of about 150k learnable parameters. 87
6.5 Source of the potential gains distributed on the different block sizes. 88
6.6 Interpolation filter distribution as a function of the QP. 90
6.7 Illustration of the bad and the right predictions of the interpolation filters on cactus sequence. 92
6.8 Source of the potential gains distributed on the different block sizes on BQ-terrace. 92
6.9 An example of high motion in a video sequence BasketballDrive. 93
6.10 The relation between the average of the motion vector amplitude and the BD-rate gains obtained. 94
6.11 a) Partitioned BasketballDrive frame colored with the first gradient map in d) according to the distortion value. b) Partitioned BasketballDrive frame colored with the second gradient map in d) according to the *AutoSAD* value. c) Original BasketballDrive frame. d) Gradient color maps. 94
6.12 Distribution of filter selection according to QP value. 96
6.13 The block sizes play an important role in the selection of the filters put into competition. 96
6.14 Measure of the correlation between different criteria and the filter choice. 97
6.15 The different options introduced to the standard interpolation process with and without separation between horizontal and vertical interpolations. 98
6.16 Percentage of selection for each of the 9 possible filter combinations. 99

7.1 Locations of the samples used for the derivation of α and β [6]. 102
7.2 Definition of samples used by PDPC applied to diagonal and adjacent angular intra modes. 104
7.3 Example of four reference lines neighboring to a prediction block [7]. 105
7.4 Sub-partition depending on the block size [8]. 108

7.5 Matrix weighted intra prediction process [9]. . . . 109
7.6 control point based affine motion models for 2 and 3 contorl points respectively [8]. . . . 111
7.7 Affine MVF per sub-block [8]. . . . 112
7.8 Locations of inherited affine motion predictors [10]. . . . 113
7.9 Control point motion vector inheritance [8]. . . . 113
7.10 Locations of Candidates position for constructed affine merge mode [8]. . . . 113
7.11 Illustration of motion vector usage for proposed combined method [11]. . . . 115
7.12 Sub-block MV VSB and pixel $\Delta v(i,j)$ (red arrow) [8]. . . . 116
7.13 Spatial neighboring blocks used by SbTVMP [8]. . . . 118
7.14 Deriving sub-CU motion field by applying a motion shift from spatial neighbor and scaling the motion information from the corresponding collocated sub-CUs [8]. . . . 119
7.15 MMVD Search Point [8]. . . . 123
7.16 Illustration for symmetrical MVD mode [8]. . . . 125
7.17 Top and left neighboring blocks used in CIIP weight derivation [8]. . . . 126
7.18 Extended CU region used in BDOF [12]. . . . 129
7.19 Decoding side motion vector refinement [13]. . . . 130
7.20 Triangle partition based inter prediction [14]. . . . 133
7.21 Uni-prediction MV selection for TPM. . . . 134
7.22 Weights used in the blending process [14]. . . . 134
7.23 SBT position, type and transform type [15]. . . . 136
7.24 Low-Frequency Non-Separable Transform process [16]. . . . 140
7.25 Illustration of the two scalar quantizers used in the proposed approach of dependent quantization [17]. . . . 142
7.26 State transition and quantizer selection for the proposed dependent quantization [17]. . . . 143
7.27 ALF filter shapes (chroma: 5×5 diamond, luma: 7×7 diamond) [8]. . . . 146
7.28 Luma mapping with chroma scaling architecture [18]. . . . 147
7.29 Graphs showing the performance obtained once the LMCS tool is disabled on the three configurations in VTM7. . . . 153
7.30 Efficiency ratios of the tools on tango in a curve. . . . 159
7.31 Convex curve obtained by the optimized order of the tools on tango sequence. . . . 160
7.32 Deactivating the tools on tango sequence one after the other based on the order of Table 7.32. . . . 160
7.33 Our method responding to two different use-cases. . . . 161
7.34 First use case with a constraint limited to a minimum of -30% of BD-R reductions compared to HEVC. . . . 161
7.35 Second use case with a constraint limited to a maximum of 500% (5 times) of run-time complexity compared to HEVC. . . . 162
7.36 First step: Cheating mode or training process. . . . 163
7.37 Comparison of the three tests performance. . . . 166
7.38 Comparison of the three tests performance for each sequence. . . . 166
7.39 Variability of SMVD performance across different GOPs from variant sequences. . . . 167
7.40 An example of simplified generalized framework. . . . 168
7.41 A second example of inference or generalized framework. . . . 169
7.42 Third example of inferring. . . . 169

7.43 Impact of the content nature on the efficiency of SMVD, on the 4 GOPs of 'AKartingIRIS_3840x2176_24fps_10bit_420' sequence. 170
7.44 Variability of AMVR performance across different GOPs from variant sequences. 171
7.45 Variability of JCCR performance across different GOPs from variant sequences. 171
7.46 Variability of LMCS performance across different GOPs from variant sequences. 172
7.47 YUV graphs from 'AJockeyHarmonics' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this seuquence. 174
7.48 YUV graphs from 'BToyCalendarHarmonics' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this seuquence. 175
7.49 YUV graphs from 'ABangkokMarketVideo' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this sequence. 176
7.50 A zoom in focused on the two extremes of the non-legal range for 'ABangkokMarketVideo' sequence. 176

List of Tables

1.1 Performances du DIMD proposé avec la méthode de fusion, dans les configurations All Intra (AI), Random Access (RA) et Low-Delay B (LB). L'ancre de référence de cette expérience est VTM-5. . . . 8
1.2 Performances de codage de C-SSIF pour QP 22-27-32-37. . . . 10
1.3 Tests VVC sur trois configurations d'outils différentes (sur du contenu UHD). . . . 12

3.1 Overview of the tools adopted in HEVC and VVC. . . . 28
3.2 YUV BD-Rate Savings of VVC (VTM-9.0) Over AVC and HEVC. . . . 46
3.3 MOS and PSNR-YUV BD-Rate Savings of VVC (VTM-10.0) Over HEVC (HM-16.22) and of an Optimized VVC Encoder (VVenC-0.1) Over VTM. . . . 48

5.1 Different classified sequences from CTC. . . . 66
5.2 Proportion of bits dedicated to intra luma mode signaling. . . . 66
5.3 Proportions based on QP values. . . . 66
5.4 Selected configurations for prediction fusion. . . . 72
5.5 Coding performance of the proposed DIMD with fusion method, in All-Intra (AI), Random Access (RA) and Low-Delay B (LB). The reference anchor of this experiment is VTM-5. . . . 75
5.6 Selection rate of the proposed DIMD with fusion, in block sizes and with two Quantization Parameters (QP). . . . 76
5.7 Share of each DIMD module from total run-time overhead of encoding and decoding each DIMD block. . . . 77

6.1 Summary table of the parameters tuned in the formula (6.3) used finally to generate the different filters. . . . 83
6.2 Luma interpolation filter coefficients for DCT-IF in VVC and the proposed alternative filter. 83
6.3 BD-rate results compared the VTM11 anchor in case of one or two alternative sub-pel interpolation filters at half plus quarter pel precision. . . . 85
6.4 Performance of the different Neural Networks classifying the three filters. . . . 90
6.5 Coding performance for CTC QP 22-27-32-37. . . . 91
6.6 Performance of the different Neural Networks for predicting DCT-IF or KCS-G. . . . 97

7.1 Simulation results of disactivating CCLM VTM tool. (VTM 7 anchor) . . . 103
7.2 Test results of VTM MRL tool "off" on various JVET CTC sequences in different configurations. . . . 106
7.3 Entropy coding coefficient group size. . . . 107
7.4 Transform selection depends on intra mode. . . . 107
7.5 Test results of VTM ISP tool "off" on various JVET CTC sequences in different configurations. . . . 109

7.6 Test results of VTM MIP tool "off" on various JVET CTC sequences in different configurations. . . . 110
7.7 Test results of VTM AFFINE tool "off" on various JVET CTC sequences in different configurations. . . . 115
7.8 Test results of VTM SbTMVP tool "off" on various JVET CTC sequences in different configurations. . . . 119
7.9 Test results of VTM AMVR tool "off" on various JVET CTC sequences in different configurations. . . . 121
7.10 Test results of VTM BCW tool "off" on various JVET CTC sequences in different configurations. . . . 122
7.11 The relation of distance index and pre-defined offset . . . 123
7.12 Sign of MV offset specified by direction index. . . . 123
7.13 Results of VTM MMVD tool "off" on various JVET CTC sequences in different configurations. . . . 124
7.14 Results of VTM SMVD tool "off" on various JVET CTC sequences in RA configuration. 125
7.15 Results of VTM CIIP tool "off" on various JVET CTC sequences in different configurations. 127
7.16 Results of VTM BDOF tool "off" on various JVET CTC sequences in different configurations. . . . 130
7.17 Results of VTM DMVR tool "off" test on various JVET CTC sequences in RA configurations. . . . 132
7.18 Results of VTM TPM tool "off" test on various JVET CTC sequences in different configurations. . . . 135
7.19 Results of VTM SBT tool "off" test on various JVET CTC sequences in different configurations. . . . 137
7.20 Results of VTM LFNST tool "off" test on various JVET CTC sequences in different configurations. . . . 137
7.21 Transform basis functions of DCT-II/VIII and DST-VII for N-point input. . . . 138
7.22 Transform and signalling mapping table. . . . 139
7.23 Performance results of VTM MTS tool "off" test on various JVET CTC sequences in different configurations. . . . 140
7.24 Transform selection table. . . . 141
7.25 Performance results of VTM DQ tool "off" test on various JVET CTC sequences in different configurations. . . . 144
7.26 Reconstruction of chroma residuals. The value CSign is a sign value (+1 or –1), which is specified in the slice header, resJointC[.][.] is the transmitted residual. . . . 145
7.27 Performance results of VTM JCCR tool "off" test on various JVET CTC sequences in different configurations. . . . 145
7.28 Performance results of VTM ALF tool "off" test on various JVET CTC sequences in different configurations. . . . 147
7.29 Performance results of VTM LMCS tool "off" test on various JVET CTC sequences in different configurations. . . . 153
7.30 Test results of VTM tools "off" test on various VTM versions in Random Access configuration. 154
7.31 Correlation between the 4 methods of tools evaluation. . . . 156
7.32 Descending order of the tools Efficiency Ratios on Tango. . . . 158
7.33 Tools activation on 4 different sequences. . . . 164

7.34 VVC tests on three different set-up of tools (on UHD content). 165
7.35 Sequences descriptions. 173

Chapter 1

Résumé (French)

1 Introduction

Les deux dernières décennies ont été témoins de développements passionnants dans les applications de l'électronique grand public. Les applications multimédias, en particulier celles en charge de l'encodage, de la transmission, du stockage et du décodage vidéo, jouent un rôle essentiel dans ce cadre. Selon une étude récente de Cisco, le contenu vidéo occupe aujourd'hui environ 82% du trafic Internet mondial [19]. Ce pourcentage important est dû à la révolution des contenus vidéo au cours des dernières années avec l'émergence des services de vidéo à la demande (VOD) (par exemple, YouTube, Netflix, Amazon prime, Hulu, etc.), de la web-TV, des sites de partage de vidéos, des réseaux sociaux (par exemple, Facebook, Instagram, Snapchat, Tik-Tok, etc.), ou du service streaming des vidéos en direct pour les particuliers (Facebook live, Twitch, etc.). Tous les services mentionnés ci-dessus augmentent rapidement le trafic vidéo sur Internet, les besoins de stockage et, surtout, son impact écologique. Par exemple, le streaming vidéo produit actuellement environ 1% des émissions mondiales de gaz à effet de serre, ce qui équivaut aux émissions de l'Espagne [20]. Les émissions de CO2 provenant du streaming vidéo devraient dépasser les émissions mondiales de CO2 dues aux véhicules d'ici 2025 [20].

Cette consommation impressionnante de contenu multimédia dans les produits électroniques grand public (ex. téléphones mobiles, téléviseurs intelligents, consoles vidéo, vidéo immersive et à 360 degrés, appareils de réalité augmentée et virtuelle) est causée par une augmentation significative de la quantité de données à transmettre. Cela a créé un besoin de développer des algorithmes de codage vidéo encore plus efficaces que ceux existants pour limiter l'augmentation du taux de transmission de données et assurer une meilleure qualité de service.

De nos jours, le marché exige des vidéos avec des résolutions plus élevées (UHD) avec une précision de pixel plus élevée (HDR ou précision 10 bits) et des fréquences d'images plus élevées (jusqu'à 120 images par seconde). Toutes ces fonctionnalités doivent être intégrées dans des appareils à faibles ressources et à batteries limitées. Par conséquent, trouver un équilibre entre la complexité des algorithmes et l'efficacité des implémentations devient un défi dans le développement de nouveaux appareils électroniques grand public.

Pour cette raison, la Joint Video Experts Team (JVET), un comité de collaboration formé par les organismes de normalisation Moving Picture Experts Group (MPEG) et Video Coding Experts Group (VCEG), a été créé dans le but de développer une nouvelle norme de codage vidéo, aujourd'hui connue sous le nom de Versatile Video Coding – VVC/H.266.

Suite à cela, un modèle de test pour la nouvelle norme, VVC, a été introduit. La première version du modèle VVC était entièrement basée sur le modèle HEVC (HM), avec quelques outils VVC supplémentaires - principalement liés au partitionnement de blocs. Ce modèle a ensuite été amélioré au

cours d'un processus de normalisation de 3 ans au fur et à mesure que de nouvelles contributions ont été adoptées.

Par rapport à son prédécesseur, HEVC [21], les principaux objectifs de la normalisation VVC sont de prendre en charge une plus large gamme de formats vidéo ainsi que d'atteindre 50% d'économie de bande passante à la même qualité subjective [13, 22, 23, 24]. Le développement de la norme VVC a commencé à la fin de l'année 2015 et s'est terminé en juillet 2020. Il convient de mentionner qu'un chemin similaire a été emprunté en 2010, lorsque la normalisation HEVC visait à obtenir un gain d'efficacité de compression de 50% par rapport au codage vidéo avancé (AVC) [25] pour une qualité visuelle similaire [26].

Actuellement, VVC [1] est la norme vidéo la plus moderne, et donc celle qui décrit l'état actuel de la technique. Bien que VVC ait adopté un schéma hybride basé blocs similaire à celui de HEVC, cette norme offre des améliorations significatives par rapport à ses prédécesseurs. Comme pour HEVC, cet objectif d'économie de bande passante est censé être atteint en adoptant des centaines de propositions, d'outils et de fonctionnalités répartis sur les principaux modules du schéma de codage hybride conventionnel de prédiction/transformation. En accumulant ces quantités d'améliorations, VVC peut obtenir une réduction de plus de 40% du débit [27] par rapport à HEVC.

2 Contributions

Dans le cadre de la norme VVC, les apports de ce travail de recherche se répartissent en deux axes principaux :

1. Proposer de nouveaux outils de compression.
2. Optimiser l'utilisation des outils déjà existants.

Il est important de noter que ces deux axes ne sont pas séparés mais partagent en fait un pilier commun, qui est la veille technologique constante sur tous les outils adoptés dans la norme. Pour le premier axe, l'analyse des outils existants permet d'établir quelles sont les pistes les plus prometteuses, dignes d'un effort d'amélioration supplémentaire. Alors que pour le deuxième axe, cette analyse aide à l'appréhension et à l'évaluation des outils existants, dans le but d'optimiser leur utilisation.

2.1 Premier axe : Proposer de nouveaux outils de codage

Le premier axe consiste à proposer et mettre en œuvre de nouveaux outils de compression dans la nouvelle norme, capables de générer des gains de codage supplémentaires à VVC. La difficulté réside dans le fait que VVC offre déjà une efficacité de compression élevée par rapport à HEVC, qui a déjà montré aussi une bonne performance de compression. Cela rend le problème de plus en plus difficile.

Au cours de cette thèse, deux approches ont été proposées pour ce premier axe, visant à augmenter l'efficacité de codage de VVC.

1. La première approche suit le concept traditionnel de compression vidéo. La méthode est ici conforme aux contraintes du monde réel en termes de complexité, d'utilisation de la mémoire et de praticité. Cette approche est plus orientée vers les codecs vidéo industriels et la standardisation.
2. La deuxième approche dépasse les contraintes et les règles existantes des méthodes conventionnelles. Les méthodes de cette catégorie sont plus susceptibles d'être utilisées pour une approche de codage sans contrainte où l'objectif est d'obtenir des gains de codage significatifs, quelle que soit la complexité introduite.

2.1.1 Première contribution d'outil de compression : Approche conventionnelle

Comme le début de la thèse tombe dans la même période que le processus de normalisation, la première proposition a pris la forme de contributions à la normalisation pour VVC. Dans ce contexte, de fortes contraintes sont imposées aux contributions, en termes, par exemple, de complexité au niveau du décodeur, de dépendances au niveau de l'analyse, de bande passante mémoire ou de possibilité de parallélisation des calculs. En général, le respect de ce type de limitations décompose le potentiel de la proposition, mais en retour, multiplie les chances d'adoption dans la norme.

L'une des méthodes envisagées est la dérivation des informations de prédiction du côté du décodeur. En effet, l'augmentation des modes de prédiction peut améliorer la précision des prédictions et donne moins de résidus d'énergie, conduisant à une réduction du débit. Néanmoins, plus de modes de prédiction impliquent plus de signalisation à envoyer dans le flux binaire pour informer le décodeur des choix qui ont été faits à l'encodeur. Les gains mentionnés ci-dessus sont donc largement compensés par la signalisation ajoutée. Si l'information de prédiction est dérivée du décodeur (le décodeur n'est plus passif, mais devient actif d'où le concept de décodeur intelligent), il sera inutile de la signaler, d'où un gain en signalisation. Notez toutefois que les calculs du décodeur sont basés uniquement sur les pixels reconstruits. Ainsi, les informations de prédiction dérivées peuvent être moins bonnes que celles estimées à l'encodeur avec les pixels sources réels. Par conséquent, selon les cas, le compromis peut être plus ou moins intéressant.

Bien que les techniques de ce type aient longtemps été étudiées [28, 29, 30, 31], elles ont fini par être retirées de la normalisation en raison de l'augmentation générée de la complexité du décodage (puissance des processeurs et des puces matérielles). En effet, plusieurs outils ont été proposés dans un modèle de test exploratoire. Ces outils consistent à soit dériver, soit affiner les informations de prédiction (généralement les vecteurs de mouvement), au niveau du décodeur à l'aide de la correspondance de modèles. Les exemples incluent la conversion FRUC (Frame Rate Up Conversion), le flux optique bidirectionnel (BIO) et le raffinement du vecteur de mouvement du décodeur (DMVR) [32]. Pour la réponse à CfP, plusieurs solutions, dont des outils de décodage intelligents, ont également été proposées [33, 34]. Il est donc très probable que de tels outils se trouvent dans la norme finale.

2.1.2 Deuxième contribution d'outil de compression : Approche basée sur l'apprentissage profond

Le fait que le processus de normalisation ait atteint sa fin au cours de la thèse, ne signifie pas du tout qu'il n'y a plus de place pour proposer des outils pour améliorer encore l'efficacité du codage. Étant donné que cette deuxième contribution est sortie du champ d'application de la normalisation, moins de contraintes ont été imposées, ce qui a donné plus de liberté à la technique proposée. Surtout qu'avec les progrès technologiques, les processeurs et les GPU deviennent moins chers et/ou plus rapides, permettant la production d'algorithmes plus grands et plus efficaces.

Certaines difficultés surgissent lorsque l'on tente d'étendre le concept d'apprentissage automatique, ou Deep Learning (DL), au cas du codage vidéo où les approches traditionnelles sont extrêmement efficaces. Cela rend le défi d'introduire le DL dans ce domaine, considérablement complexe. Aujourd'hui, les contributions du DL à la compression vidéo peuvent impliquer l'optimisation des outils existants (par exemple, l'allocation de débit), l'introduction de nouveaux outils (par exemple, de nouvelles méthodes de prédiction) ou une nouvelle architecture. L'un des objectifs de cette thèse est d'explorer les contributions possibles du DL au codage vidéo.

Parmi les outils dits normatifs, les nouveaux prédicteurs (prédiction spatiale ou temporelle) basés sur le DL semblent être l'approche la plus prometteuse. Par exemple, l'analyse du coût des informations de

prédiction peut bénéficier des approches DL, dans le but de prédire certains choix d'encodeur au niveau du décodeur. De ce fait, ces choix n'auraient plus besoin d'être signalés.

2.2 Deuxième axe : Optimisation des outils existants

Le deuxième axe, contrairement au premier, ne cherche pas à ajouter une contribution à l'algorithme de base de VVC. Néanmoins, cet axe vise à construire une utilisation optimisée de l'algorithme VVC déjà existant. Pour cela, une étude approfondie nécessaire est menée pour évaluer et caractériser les nouveaux outils de codage proposés pour la VVC. L'analyse tient compte des résultats de codage objectifs des différents outils et de leur complexité.

Tous les outils intégrés dans VVC ont amélioré l'efficacité de compression, mais cela au prix d'une augmentation complexité. Par conséquent, cette contribution vise à détecter les outils qui sont efficaces pour toutes les séquences vidéo, et les autres outils qui peuvent être efficaces pour une séquence ou un type spécifique de contenu, et pour un type spécifique de cas d'utilisation. Cette décision peut être appliquée à chaque séquence vidéo ou partie d'une séquence en fonction du contenu et des caractéristiques locales de la séquence.

L'objectif ultime est de trouver le meilleur compromis possible entre l'efficacité de compression fournie et la complexité imposée par les outils. Ainsi, un système d'optimisation est conçu pour déterminer une technique efficace d'activation des nouveaux outils de codage. La détermination des outils et des algorithmes mis en place peut être effectuée soit en utilisant des réseaux de neurones artificiels, soit sans aucune technique d'Intelligence Artificielle (IA).

3 Structure du manuscrit

3.1 Partie I

Le deuxième chapitre de cette thèse commence par un contexte détaillé allant d'une présentation générique de la représentation et du codage vidéo à une description plus concrète de la dernière norme de codage vidéo : VVC. L'objectif principal de cette thèse étant de proposer de nouvelles techniques (intelligence artificielle par exemple) pour améliorer l'efficacité de la compression tout en contrôlant la complexité supplémentaire, un bref résumé des méthodes d'apprentissage automatique et d'apprentissage profond est également fourni en toile de fond.

Ensuite, dans le chapitre 3, un large état de l'art, lié aux contributions à cette thèse, est discuté. Dans ce chapitre, plusieurs travaux proposés pour les prédictions intra ou inter sont mentionnés. En outre, une vaste étude des techniques proposées basées sur l'IA dans le codage vidéo est rapportée. Enfin, nous donnons un aperçu des résultats précédents en relation avec les méthodes d'optimisation des encodeurs.

3.2 Partie II

Dans la deuxième partie, les contributions du premier axe sont détaillées comme suit :

- Le chapitre 4 détaille la première contribution en mode intra. Premièrement, nous exposons le problème que notre proposition est en mesure de résoudre. Ensuite, la motivation derrière le travail est présentée avant de donner un aperçu de la méthode. Dans le but d'augmenter l'efficacité du codage, la nouvelle méthode présentée repose sur deux composantes principales : l'analyse de

gradient générant les modes intra et la fusion de ces modes dans le but d'améliorer la précision des modes directionnels de prédiction. Les gains de codage significatifs sont ensuite donnés et analysés.

- Le chapitre 5 présente la deuxième contribution du premier axe. Ce travail cible le mode d'inter prédiction, plus précisément le processus d'interpolation. À l'image de la structure de la première contribution, ce chapitre commence par examiner le problème abordé, suivi de la motivation et des avantages d'une telle proposition. Deux piliers principaux constituent cette contribution qui consiste à proposer une nouvelle façon d'interpoler. En tant que premier pilier, un processus est développé pour concevoir des filtres d'interpolation alternatifs avec une précision arbitraire. Nous montrons expérimentalement que des gains peuvent être trouvés par plusieurs filtres alternatifs à une fréquence de coupure inférieure à celle du filtre standard de VVC. En tant que deuxième pilier, nous mettons en œuvre un schéma où le décodeur déduit le filtre optimal pour une unité de codage sans avoir besoin de signalisation. À savoir, nous entraînons un réseau neuronal convolutif pour prédire quel filtre minimise la distorsion pour une unité de codage donnée en regardant son prédicteur compensé par le mouvement. Les résultats de notre méthode sont rapportés ensuite avant de conclure la contribution en donnant quelques perspectives.

3.3 Partie III

La troisième partie est consacrée à la contribution du deuxième axe.

Le chapitre 6 commence par une description détaillée d'une liste d'outils VVC commutables adoptés lors de la normalisation. Après avoir décrit le problème causé par ce nombre croissant d'outils adoptés, la motivation de personnaliser les outils en fonction du contenu vidéo est justifiée. Après avoir présenté un aperçu de la méthode, nous donnons une explication détaillée des deux principaux processus contribuant à la réalisation de l'objectif de cette proposition. Le premier processus est construit afin de définir la méthode d'optimisation à suivre. Ce mécanisme donne la possibilité de classer les outils de manière adaptative au contenu d'une façon qui garantit le compromis souhaité. Enfin, une fois cette méthode validée sur un ensemble de séquences en prouvant des réductions de temps d'encodage considérables pour des pertes de compression négligeables, le deuxième mécanisme est de généraliser le premier pour être applicable sur n'importe quelle séquence. Cette dernière étape peut être effectuée que ce soit en utilisant une technique d'IA ou non.

Nous clôturons ce manuscrit avec un résumé des méthodes proposées et de leurs résultats associés, ainsi que quelques perspectives pour les futurs travaux dans ce domaine.

4 Résultats et conclusions

4.1 Première contribution : DIMD

4.1.1 Principe de fonctionnement

En bref, notre méthode proposée réduit le taux d'encodage d'une vidéo (à la même qualité) en ajoutant un nouveau mode de codage intra. L'opération mentionnée ici consiste à traiter le DIMD de la même manière que les autres modes de codage et dans le cas où ce mode présente le coût d'encodage le moins cher, il sera choisi parmi les modes et son choix sera signalé au décodeur. En mode DIMD, il n'est pas nécessaire d'envoyer le mode de prédiction Intra. Nous proposons une technique plus intelligente en construisant un histogramme de gradients sur une zone causale entourant le bloc actuel, et en dérivant normativement

Classe	Séquence	All Intra			Random Access			Low Delay B		
		BDR	**ET**	**DT**	**BDR**	**ET**	**DT**	**BDR**	**ET**	**DT**
A1	Tango2	-0.97%	108%	104%	-0.51%	102%	102%	-0.18%	103%	101%
	FoodMarket	-0.97%	108%	104%	-0.37%	102%	102%	-0.29%	102%	100%
	Campfire	-0.59%	109%	104%	-0.44%	102%	102%	-0.25%	103%	101%
	Moyenne	**-0.84%**	**108%**	**104%**	**-0.44%**	**102%**	**102%**	**-0.24%**	**103%**	**101%**
A2	CatRobot	-0.55%	109%	103%	-0.26%	102%	102%	-0.17%	102%	101%
	DaylightRoad2	-0.43%	106%	104%	-0.20%	102%	102%	-0.26%	102%	101%
	ParkRunnig3	-0.47%	108%	104%	-0.17%	102%	102%	-0.32%	102%	101%
	Moyenne	**-0.48%**	**108%**	**104%**	**-0.21%**	**102%**	**102%**	**-0.25%**	**102%**	**101%**
B	MarketPlace	-0.47%	107%	104%	-0.21%	101%	101%	-0.24%	102%	101%
	RitualDance	-0.85%	109%	104%	-0.43%	101%	102%	-0.29%	101%	102%
	Cactus	-0.51%	108%	104%	-0.34%	102%	102%	-0.28%	102%	101%
	BasketballDrive	-0.68%	109%	105%	-0.46%	102%	102%	-0.25%	102%	100%
	BQTerrace	-0.25%	108%	104%	-0.14%	102%	102%	-0.13%	102%	101%
	Moyenne	**-0.55%**	**108%**	**104%**	**-0.31%**	**102%**	**102%**	**-0.24%**	**102%**	**101%**
C	BasketballDrill	-0.32%	109%	104%	-0.34%	102%	101%	-0.25%	101%	101%
	BQMall	-0.73%	108%	104%	-0.38%	102%	102%	-0.18%	102%	102%
	PartyScene	-0.61%	109%	105%	-0.36%	102%	102%	-0.23%	102%	101%
	RaceHorses	-0.70%	108%	105%	-0.37%	102%	101%	-0.27%	102%	101%
	Moyenne	**-0.59%**	**109%**	**105%**	**-0.36%**	**102%**	**102%**	**-0.23%**	**102%**	**101%**
D	BasketballPass	-0.67%	109%	104%	-0.36%	102%	102%	-0.04%	102%	101%
	BQSquare	-0.42%	110%	104%	-0.18%	102%	101%	-0.29%	102%	101%
	BlowingBubbles	-0.89%	110%	104%	-0.25%	103%	102%	-0.29%	103%	102%
	RaceHorses	-0.68%	111%	104%	-0.28%	102%	101%	-0.14%	103%	102%
	Moyenne	**-0.67%**	**110%**	**104%**	**-0.27%**	**102%**	**102%**	**-0.19%**	**103%**	**102%**
E	FourPeople	-0.74%	108%	104%	-0.41%	102%	102%	0.04%	102%	101%
	Johnny	-0.79%	108%	104%	-0.48%	103%	102%	-0.11%	103%	101%
	KristenAndSara	-0.72%	109%	104%	-0.47%	102%	103%	-0.45%	102%	101%
	Moyenne	**-0.75%**	**108%**	**104%**	**-0.45%**	**102%**	**102%**	**-0.17%**	**102%**	**101%**
F	BasketballDrillText	-0.28%	111%	104%	-0.30%	102%	102%	-0.27%	102%	101%
	ArenaOfValor	-0.80%	111%	104%	-0.39%	102%	102%	-0.28%	102%	101%
	SlideEditing	-0.16%	110%	104%	-0.17%	102%	102%	-0.23%	102%	102%
	SlideShow	-0.13%	110%	104%	-0.24%	103%	103%	-0.21%	102%	101%
	Moyenne	**-0.34%**	**111%**	**104%**	**-0.28%**	**102%**	**102%**	**-0.25%**	**102%**	**101%**
Total		**-0.6%**	**109%**	**104%**	**-0.31%**	**102%**	**102%**	**-0.23%**	**102%**	**101%**

TABLE 1.1: Performances du DIMD proposé avec la méthode de fusion, dans les configurations All Intra (AI), Random Access (RA) et Low-Delay B (LB). L'ancre de référence de cette expérience est VTM-5.

le meilleur mode intra pour le bloc courant à partir de cet histogramme. Un autre avantage de cette proposition est la possibilité d'avoir plus d'une direction d'intra-prédiction grâce à cette même analyse de texture de faible complexité. Ces différents modes intra-prédiction déduits peuvent ensuite être combinés au moyen de la fusion de prédiction, pour obtenir l'intra-prédiction finale pour un bloc donné. Cette propriété de fusion permet à DIMD de prédire plus précisément les zones avec des textures complexes, ce qui, dans les schémas de codage conventionnels, nécessiterait plutôt un partitionnement et/ou une transmission plus fine des résidus à haute énergie.

4.1.2 Résultats

Les résultats de DIMD dans VTM-5 montrent un compromis significatif entre l'efficacité du codage et la complexité du codec. Plus précisément, DIMD peut fournir jusqu'à -0,6% de gain d'efficacité de codage BDR avec un temps d'exécution de l'encodeur (ET) et du décodeur (DT) de 109% et 104%, respectivement.

4.1.3 Conclusions

Cette technique présente quelques inconvénients : d'abord la dérivation du mode, comme elle doit être effectuée au niveau décodeur, est basée sur des pixels reconstruits et non sur les pixels d'origine, ce qui entraîne une perte de précision. En outre, l'opération ajoute de la complexité au processus de décodage, ce qui peut causer des problèmes pour les implémentations matérielles. Cependant, tant que la complexité ajoutée au décodeur a été maîtrisée comme présenté dans la Table 1.1, et que le gain en termes de réduction du signal compense la perte de précision dans la prédiction, notre technique a de grandes chances d'être adoptée dans une prochaine norme de compression vidéo.

Pour conclure, ce projet ne se limite pas aux travaux décrits. Plusieurs propositions pour améliorer les performances de notre méthode sont prévues : tout d'abord, au lieu d'utiliser le filtre Sobel 3×3 pour effectuer l'analyse de gradient, nous pouvons le remplacer par un filtre plus simple (filtres 3x2 ou même 2x2) capable de fournir le même résultat. Une autre amélioration future que nous prévoyons de faire, afin de réduire davantage la complexité, est d'éviter le calcul supplémentaire des coûts de RD imposé par DIMD. Une façon de le faire est de prendre les décisions de l'encodeur en fonction de la distorsion au lieu d'ajouter une vérification RD. Cette solution a prouvé son potentiel dans la deuxième contribution du premier axe (C-SSIF). Ces améliorations, inspirées du fait que DIMD n'a finalement pas été adopté dans la norme en raison du niveau de complexité, peuvent avoir un impact important sur le temps d'exécution.

Enfin, il convient de noter que l'algorithme proposé est innovant car il n'a pas fait l'objet d'autres travaux publiés. Cela m'a offert la chance de rédiger mes premiers brevets déclarant une invention dans le domaine du codage vidéo. De plus, après un examen attentif des résultats montrant des gains de codage intéressants tout en maintenant un faible degré de complexité (en particulier au niveau du décodeur), ce travail a offert à ATEME l'occasion de présenter ce travail comme contributions lors des réunions JVET, car il avait le potentiel d'être inclus dans la norme VVC. DIMD a reçu beaucoup de commentaires positifs mais n'a finalement pas été adopté. Cependant, les résultats obtenus ont fait l'objet de deux publications de conférences internationales (DCC-IEEE et ICIP-IEEE).

4.2 Deuxième contribution : C-SSIF

4.2.1 Principe de fonctionnement

La deuxième contribution de la première piste décrit une autre technique proposée pour améliorer l'efficacité de codage de VVC. La méthode consiste à donner à VVC la possibilité de basculer entre différents filtres d'interpolation avec un moyen intelligent d'éviter la signalisation de l'encodeur vers le côté décodeur, appelé CNN-based Switchable Sub-pel Interpolation Filters (C-SSIF) qui veut dire filtres commutables d'interpolation sous-pixel basés sur CNN. Comme pour DIMD, C-SSIF a également été implémenté dans le VTM, mais peut être adapté à n'importe quelle norme future.

C-SSIF peut être divisé en 2 parties. Dans la première partie, nous proposons une famille de filtres obtenus en convoluant une fenêtre Kaiser avec un Sinc (KCS). Ces filtres, avec des caractéristiques passe-bas plus fortes que DCT-IF, peuvent être bénéfiques pour atténuer les composants de bruit à haute fréquence dans certains contextes. La deuxième partie est un schéma où le décodeur déduit le filtre optimal sélectionné par l'encodeur pour chaque unité de codage, sans avoir besoin de signaler les coefficients de filtre. Pour ce faire, nous avons entraîné des réseaux de neurones convolutifs à prédire quel filtre minimise le taux de distorsion attendu pour chaque unité de codage en ne regardant que la référence compensée par le mouvement dans le tampon de décodage. L'encodeur et le décodeur partagent un tel réseau, de sorte que la prédiction du filtre au décodeur correspond à celle de l'encodeur.

Résolution	Séquence	Y	U	V
4K (3840x2160p)	Tango2	-0,68%	-0,21%	-0,59%
	FoodMarket	-0,77%	-0,64%	-1,09%
	Campfire	-0,40%	-0,45%	-0,27%
	CatRobot	-0,59%	-0,20%	-0,63%
	DaylightRoad2	-0,37%	-0,33%	-0,02%
	ParkRunnig3	-0,62%	-0,39%	-0,25%
	TrafficFlow	-0,12%	-0,27%	-0,35%
	Drums	-0,41%	-0,02%	-0,32%
Moyenne		**-0,50%**	**-0,31%**	**-0,44%**
Full HD (1920x1080p)	MarketPlace	-0,17%	-0,52%	-0,44%
	RitualDance	-0,56%	0,21%	-0,22%
	Cactus	-0,41%	-0,20%	-0,53%
	BasketballDrive	-0,69%	0,76%	-0,44%
	BQTerrace	-0,09%	-0,08%	0,50%
	Kimono	-1,06%	-1,08%	-0,40%
Moyenne		**-0,50%**	**-0,15%**	**-0,26%**
WVGA (832x480p)	BasketballDrill	-0,52%	0,03%	-0,94%
	BQMall	-0,24%	-0,41%	0,07%
	PartyScene	-0,11%	-0,20%	-0,38%
	RaceHorses	-0,56%	0,48%	0,16%
Average		**-0,36%**	**-0,03%**	**-0,27%**
Total		**-0,47%**	**-0,20%**	**-0,34%**

Table 1.2: Performances de codage de C-SSIF pour QP 22-27-32-37.

4.2.2 Résultats

La méthode proposée permet d'obtenir en moyenne une réduction du taux de débit-distorsion (ou BD) de -0,47%, -0,20% et -0,34% pour la gamme QP de la CTC : 22-27-32-37 (Table 1.2). L'objectif principal de ce travail est de montrer l'intérêt d'adapter le filtre d'interpolation au contenu, à une granularité fine, par exemple au niveau du bloc. Enfin, les résultats expérimentaux présentés ont prouvé l'efficacité du passage du concept d'interpolation avec un filtre fixe à l'alternance dynamique entre trois ou plusieurs filtres sans signalisation en utilisant des réseaux de neurones artificiels pour prédire le meilleur filtre.

4.2.3 Conclusions

Cette technique a aussi ses problèmes. Par exemple, le filtre optimal peut ne pas être toujours correctement prédit en raison d'erreurs de prédiction faites par les réseaux de neurones. Cela peut entraîner certaines pertes des gains potentiels. Un autre problème est le niveau de complexité imposé par les CNN.

Pour conclure, plusieurs propositions et travaux futurs sont suggérés afin d'investir et de bénéficier du potentiel de cette nouvelle approche, comme l'extension de trois filtres d'interpolation à plus et la séparation entre les interpolations verticales et horizontales. Étant donné que dans le modèle de référence VVC, un filtre d'interpolation DCT fixe est proposé, les interpolations horizontales et verticales ont été effectuées à l'aide de ce même filtre. Cependant, dans notre proposition, le processus d'interpolation peut bénéficier de la présence de trois filtres différents ou plus. La séparation crée différentes combinaisons de filtres, de sorte que les interpolations horizontales et verticales peuvent être exécutées avec différents filtres si nécessaire. Par conséquent, le processus d'interpolation gagne en flexibilité et s'adapte davantage à chacune des deux directions. Par conséquent, l'inter prédiction devient plus efficace car il y aura moins d'informations résiduelles et moins de données à encoder et à transmettre. Les résultats expérimentaux montrent que les gains potentiels de taux de BD peuvent augmenter considérablement en raison de l'acte de séparation de l'interpolation.

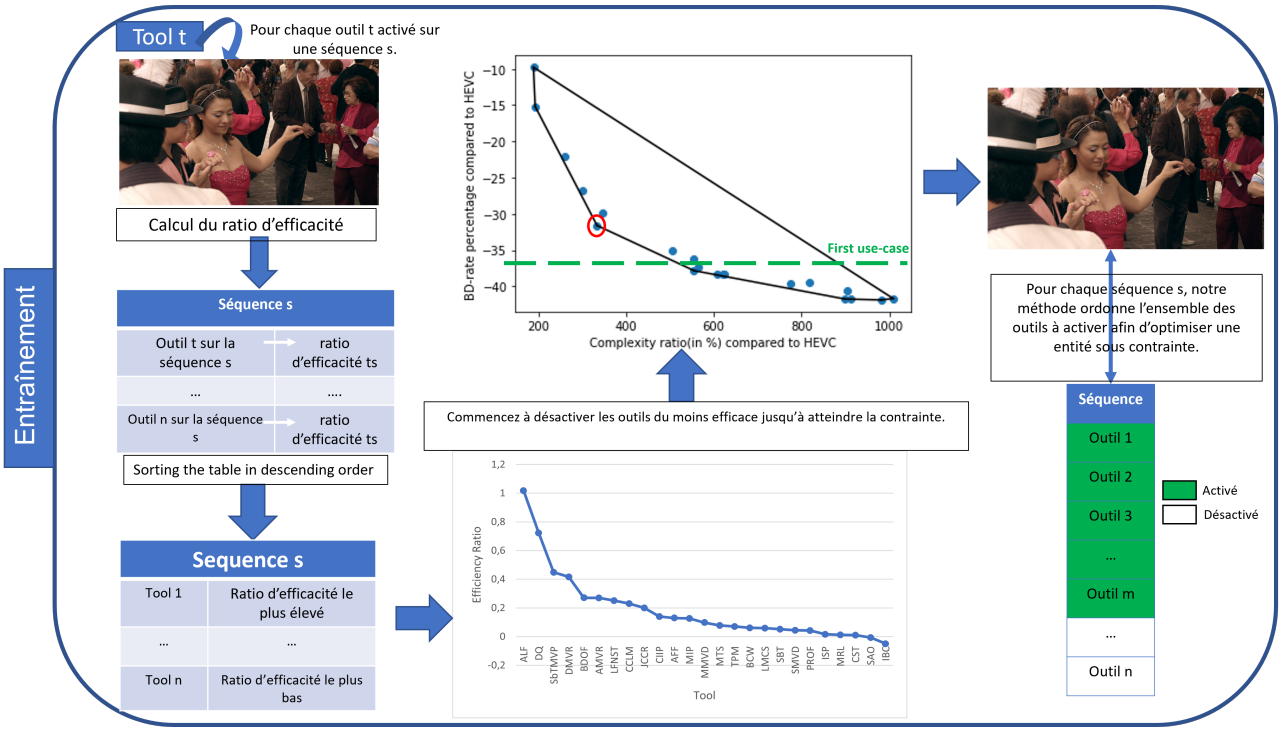


FIGURE 1.1: Première étape : processus d'entraînement.

Il convient de mentionner que ce deuxième travail n'a pas non plus fait l'objet d'autres travaux publiés. Cela m'a offert la possibilité de remplir un autre brevet déclarant une invention dans les domaines du codage vidéo et de l'apprentissage profond (Deep Learning). De plus, un papier qui détaille les techniques proposées est en cours de finalisation pour une soumission dans une revue scientifique.

4.3 Troisième contribution : Optimized Content-Adaptive Activation of VVC tools

4.3.1 Principe de fonctionnement

VVC atteint un niveau significatif d'économie de bande passante en accumulant plusieurs améliorations et en adoptant des centaines de propositions. Cependant, ce nombre élevé d'outils et d'algorithmes adoptés a entraîné une complexité supplémentaire significative. Pour cette raison, il est nécessaire de trouver le compromis optimal entre complexité et efficacité de compression en déterminant l'activation optimisée des outils adoptés en fonction des besoins.

Tout d'abord, les performances de chaque outil sont étudiées sur un nombre considérable de séquences pour comprendre le cas d'utilisation pratique de chacun des outils VVC. Compte tenu du problème considéré, la performance d'un outil est censée être le compromis entre l'efficacité de compression fournie et la complexité imposée par l'outil de codage. Pour cela, la métrique de mesure de l'efficacité d'un outil donné est le rapport entre les gains de débit-distorsion de cet outil et sa complexité. Ensuite, les outils sont triés par ordre décroissant de ratio d'efficacité. Vu le but de notre méthode qui consiste à déterminer les outils qui doivent être désactivés sur un contenu donné afin de garantir un fonctionnement optimal de VVC, cet ordre des outils nous amène à déterminer l'ordre de désactivation des outils sur chaque séquence considérée, par ordre croissant du rapport d'efficacité (du moins efficace au plus efficace). Ainsi, une courbe

	Over HEVC					
Test	**VVC tous les outils activés**		**VVC tous les outils désactivé**		**compromis obtenu**	
Sequences	BDR	ET	BDR	ET	BDR	ET
Tango	-41,96%	850%	-9,77%	188%	-32,00%	338%
FoodMarket	-38,29%	758%	-16,08%	159%	-32,74%	413%
Campfire	-38,76%	1470%	-14,10%	351%	-34,56%	659%
CatRobot	-46,58%	991%	-10,15%	180%	-38,04%	479%

TABLE 1.3: Tests VVC sur trois configurations d'outils différentes (sur du contenu UHD).

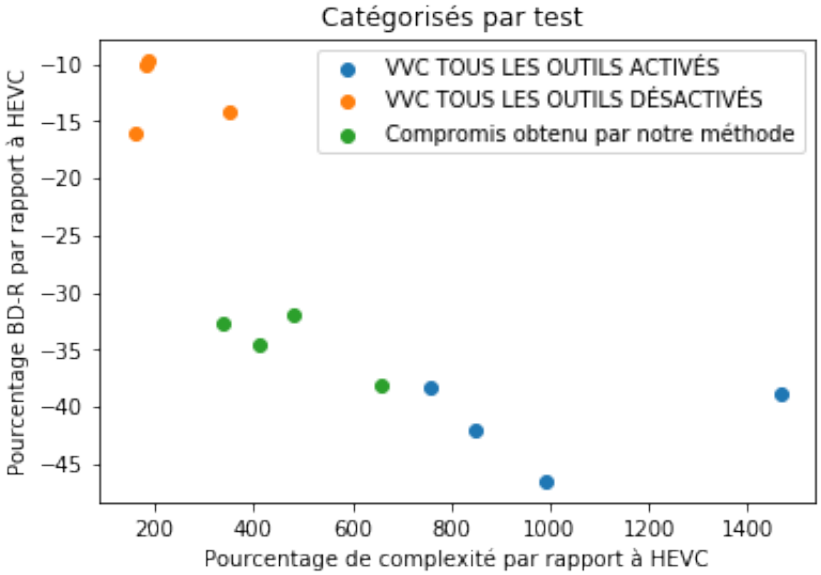


FIGURE 1.2: Comparaison des performances des trois tests.

convexe est générée en suivant cet ordre. Selon le cas d'usage dans lequel on applique notre méthode, une contrainte (de complexité ou de gains de codage) est déterminée. Le point optimal désiré est le dernier point de la courbe convexe avant d'atteindre la limite de la contrainte. Ce point optimal correspond au dernier outil à désactiver. Dans la Figure 1.1, le cas d'utilisation considéré est un seuil d'efficacité de compression. Par conséquent, pour toute séquence donnée sous une contrainte définie, cette première étape détermine l'ordre de désactivation des outils afin d'obtenir une performance optimisée de VVC. Ce processus est répété sur un tas de séquences, pour entraîner notre méthode, visant à la généraliser.

Les étapes décrites et illustrées dans la Figure 1.1 constituent le premier cadre de notre méthode. Ce cadre est construit afin de définir la méthode d'optimisation à suivre. En d'autres termes, ce processus a pour but d'entrainer notre méthode. Une fois cette méthode validée sur un ensemble de séquences, le cadre est généralisé pour être applicable sur n'importe quelle nouvelle séquence. Cette deuxième étape peut être effectuée que ce soit en utilisant une technique d'IA, ou non. En suivant ces étapes, notre méthode forme un prédicteur automatisé capable d'optimiser l'activation de n'importe quel outil sur n'importe quel contenu.

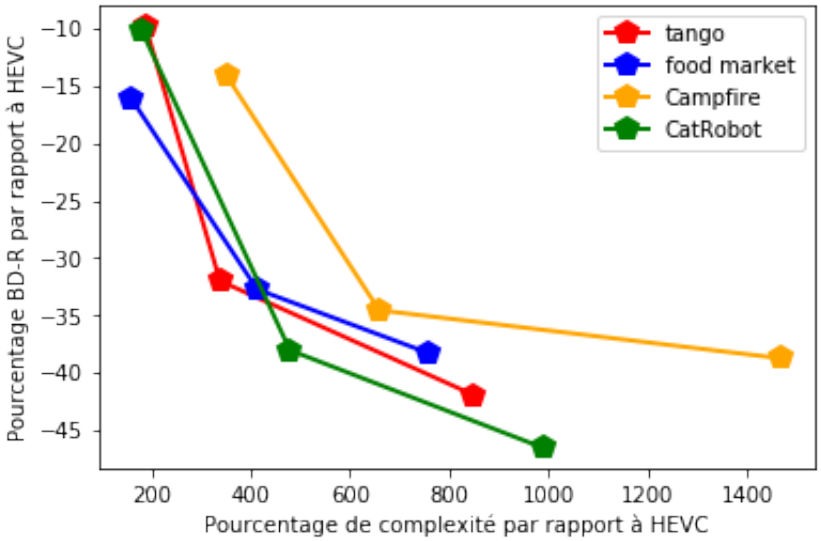


FIGURE 1.3: Comparaison des performances des trois tests pour chaque séquence.

4.3.2 Validations expérimentales

Nous présentons dans la Table 1.3, trois colonnes représentant chaque expérience lancée avec le modèle de test VVC 8 (VTM8.0) sur des séquences de classe A1/A2 (UHD). Pour la première expérience, les performances de codage VVC sont mesurées avec tous les outils de codage activés (première colonne du tableau). La deuxième colonne représente quant à elle le résultat de la désactivation de tous les outils commutables à l'exception de ceux liés au partitionnement. Cette expérience offre la complexité la plus faible possible, au prix d'un gain en débit beaucoup plus faible. Nous procédons à une troisième expérience, présentée dans la troisième colonne, visant à trouver le meilleur compromis complexité/débit. Dans cette dernière expérience, nous voulons conserver la proportion la plus élevée possible des gains de débit maximaux fournis par le VVC complet (tous les outils activés) pour une réduction remarquable du temps requis pour l'encodage. La Figure 1.2 et la Figure 1.3 illustrent la performance fournie par notre méthode d'optimisation de deux manières différentes. Dans la Figure 1.2, les points de performance sont regroupés par tests de la Table 1.3. Par exemple, la catégorie de compromis (en vert) correspond à la troisième colonne du tableau susmentionné. Ce test fait des compromis entre une efficacité de compression non loin de celle maximale obtenue par VVC complet et une complexité raisonnable non loin de celle obtenue par l'encodeur VVC le moins complexe, avec tous les outils désactivés. La Figure 1.3 montre le compromis obtenu pour chaque séquence.

Dans ce travail, nous montrons qu'avec un sous-ensemble d'outils, VVC est capable d'assurer des gains suffisants avec moins de complexité ajoutée. Finalement, notre méthode fournit un encodeur deux fois moins complexe que le VVC complet, pour une économie de débit seulement 1.2 fois inférieure. Le compromis désactive certains outils apportant des gains mineurs par rapport à la complexité introduite. Étant donné que VVC et Essential Video Coding (EVC) bénéficient d'un schéma et d'outils de partitionnement similaires, cette technique peut être appliquée sur EVC ou une autre norme future.

4.3.3 Conclusions

Ensuite, la deuxième étape consiste à prédire pour une séquence donnée, cet ordre de désactivation pour atteindre les performances optimales. Au lieu de calculer pour chaque outil, son rapport d'efficacité, notre classificateur détermine pour toute séquence sous une certaine contrainte, l'ensemble des outils à désactiver offrant une performance optimisée.

Afin de prédire cette activation optimale de l'outil VVC considéré au niveau de la GOP, un travail actuel est lancé dans le but d'implémenter un réseau de neurones convolutif (CNN). Nous supposons que le réseau prend toujours en entrée les pixels sources (pas encore codés) des séquences considérées. Un indicateur signalant l'activation de l'outil en question est envoyé de l'encodeur au décodeur pour assurer la conformité entre les deux côtés. Dans le but d'entrainer notre réseau, 200 séquences Full HD et 4K du jeu de données BVI-DVC [35] sont choisies pour entrainer le réseau neuronal à prédire le comportement de l'algorithme RDO de chaque outil VVC. La conception de la topologie du réseau est dictée par certaines contraintes. L'une de ces contraintes est la complexité de calcul du réseau ainsi que son empreinte mémoire, de sorte qu'il peut être utilisé en temps réel également sur des décodeurs de faible puissance. Un exemple d'un tel réseau est le Mobilenet [36]. Comme son nom l'indique, le modèle MobileNet est conçu pour être utilisé dans les applications mobiles. Ces CNN sont de petits modèles à faible latence et à faible consommation paramétrés pour répondre aux contraintes de ressources d'une variété de cas d'utilisation. Ils peuvent être développés pour la classification, la détection, l'intégration et la segmentation.

L'inconvénient d'une telle technique est la précision de prédiction du bon ensemble d'outils. Cela peut entraîner une mauvaise performance représentée, par exemple, par une perte de compression perceptible avec une réduction mineure de la complexité. Cependant, le travail est focalisé actuellement sur cette tâche d'améliorer la précision de prédiction par un prédicteur performant. Notre méthode aura ainsi toutes les chances d'être adoptée dans n'importe quelle prochaine norme de compression vidéo.

Enfin, il convient de noter que ce travail n'a, à notre connaissance, fait l'objet d'aucun autre travail publié. Cela m'a donné la chance de conclure ma thèse par le dépôt d'un cinquième brevet d'invention.

Chapter 2

Introduction

2.1 Context

The last two decades have witnessed exciting developments in consumer electronics applications. Multimedia applications, especially those in charge of video encoding, transmission, storage and decoding, play a critical role in this framework. According to a recent Cisco research, video content occupies nowadays around 82 percent of global Internet traffic [19]. This significant percentage is due to the revolution of the video contents in the last years with the emergence of Video-On-Demand (VOD) services (e.g. YouTube, Netflix, Amazon prime, Hulu, etc.) web-TV, video-sharing sites, social networks (e.g. Facebook, Instagram, Snapchat, Tik-Tok, etc.), or live video streaming service for individuals and so on. All of the above-mentioned services are rapidly increasing the Internet video traffic, storage requirements, and most importantly, its ecological aspect. For example, video streaming is currently producing around 1% of global greenhouse gas emissions, which is equivalent to the emissions of Spain [20]. The CO2 emission from video streaming is expected to surpass worldwide CO2 emission due to vehicles by 2025 [20].

This impressive consumption of multimedia content in consumer electronic products (e.g. mobile phones, smart TVs, video consoles, immersive and 360-degree video, and augmented and virtual reality devices) is caused by a significant increase in the amount of data to pass. This created a need to develop more efficient video coding algorithms even more effective than existing ones to limit the increase of data transmission rate and ensure a better quality of service.

Nowadays, the market is demanding videos with higher resolutions (i.e., greater than 4K) with higher pixel precision (HDR or 10-bit precision) and higher frame rates (up to 120 frames per second). All of these features must be integrated in devices with low resources and limited batteries. Therefore, finding a balance between complexity of the algorithms and efficiency in the implementations is a challenge in the development of new consumer electronic devices.

For this reason, the Joint Video Experts Team (JVET), a collaborative committee formed by the standardization bodies Moving Picture Experts Group (MPEG) and Video Coding Experts Group (VCEG), has been created with an aim to develop a new video coding standard, today known as Versatile Video Coding – VVC/H.266.

An exploratory test model, Joint Exploration Model (JEM), has therefore been put in place, and the new coding tools that have been proposed, allowed for already a reduction of bit-rate between 25% and 30% compared to HEVC [32]. This motivated JVET to launch a formal standardization activity in order to release finally the new standard. A call for contributions [37] (Call for Proposal - CfP) was thus launched, and the many responses to this call were evaluated at the 122nd MPEG meeting in San Diego (April 2018). Following this, a test model for the new standard, dubbed Versatile Video Coding (VVC), was introduced. The earliest version of the VVC model was entirely based on the HEVC Model (HM),

with a few additional VVC tools - mostly related to block partitioning. This model was then improved in the course of a 3-year standardization process as new contributions were adopted.

Compared to its predecessor, HEVC [21], the main objectives of the VVC standardization are supporting a wider range of video formats as well as reaching 50% bandwidth saving at the same subjective quality [13, 22, 23, 24]. The development of VVC standard began in late 2015 and ended in July 2020. It is worthy of mention that a similar path was taken back in 2010', when the HEVC standardization aimed at obtaining 50% compression efficiency gain with respect to Advanced Video Coding (AVC) [25] at 50% for similar visual quality [26].

Currently, VVC [1] is the most modern video standard, and thereby the one that describes the current state of the art. Although VVC has adopted a similar hybrid block-based scheme as in HEVC, it offers significant improvements over its predecessors. Like HEVC, this target bandwidth saving has been supposed to be achieved by adopting hundreds of proposals, tools and features distributed over the main modules of the conventional hybrid prediction/transform coding scheme. By accumulating this amounts of improvements, VVC was able to achieve more than 40% [27] bit-rate reduction compared to HEVC.

2.2 Contributions

In the context of the VVC standard, the contributions of this research fall into two main axes:

1. Proposing new compression tools into VVC.

2. Optimizing the usage of already existing tools in VVC.

It is important to note that these two axes are not separated but actually share a common pillar, which is the constant technology watch on all the tools being adopted in the standard. For the first axis, the analysis of existing tools makes it possible to establish what the most promising tracks are, worthy further improvement effort. While for the second axis, this analysis helps the apprehension, assessment and evaluation of the existing tools, with the goal of optimizing their usage.

2.2.1 First axis: proposing new coding tools

The first axis consists in proposing and implementing novel compression tools into the new standard, capable of yielding interesting additional coding gains to VVC. The difficulty lies in the fact that VVC already offers high compression efficiency compared to HEVC, which has already shown good compression performances as well.

During this thesis, two approaches were proposed for this axis, aiming at increasing the coding efficiency of VVC.

1. The first approach follows the traditional concept in video compression. The method here complies to real-world constraints in terms of complexity, memory usage, and practicality. This approach is more oriented towards industrial video codecs and standardization.

2. The second approach, more or less, violates the existing constraints and rules of the conventional methods. The methods in this category are more likely for unconstrained coding approach where the aim is to achieve significant coding gains regardless of the introduced complexity.

2.2.1.1 First compression tool contribution: Conventional Approach

As the beginning of the thesis falls in the same period of the standardization process, the first proposal takes the form of standardization contributions for VVC. In this context, strong constraints are imposed on the contributions, in terms of, for example, complexity at the decoder, dependencies at the parsing level, memory bandwidth or possibility of parallelization of the calculations. In general, respecting these kinds of limitations breaks down the potential of the proposal, but in return, multiplies the chances of adoption in the standard.

One of the envisaged methods is the derivation of prediction information on the decoder side. Indeed, the increase of the prediction modes can improve the precision of the predictions and gives less energy residues, leading to a reduction of flow. Nevertheless, more prediction modes involve more signaling to be sent in the bitstream to inform the decoder of the choices that have been made to the encoder. The provided gains are therefore largely offset by the added signaling. If the prediction information is derived from the decoder (the decoder is no longer passive, but becomes active hence the concept of smart decoder), it is useless to report it, hence a gain in signaling. Note however that decoder calculations are based on reconstructed pixels only. Thus, the derived prediction information may be less good than estimated at the encoder with the true source pixels. Consequently, depending on the case, the compromise can be more or less interesting.

Although techniques of this type have long been studied [28, 29, 30, 31], they have ended up by being removed from normalization because of the generated increase in the decoding complexity (power of processors and hardware chips). Indeed, several tools in the JEM have been proposed in which the prediction information (typically the motion vectors) are either derived or refined at the decoder with the use of template matching. Examples include Frame Rate Up Conversion (FRUC), BI-Directional Optical Flow (BIO), and Decoder Motion Vector Refinement (DMVR) [32]. For the answer to CfP, several solutions, including intelligent decoding tools, have also been proposed [33, 34]. It is therefore very likely that such tools are found in the final standard.

2.2.1.2 Second compression tool contribution: Deep Learning-based Approach

The fact that the standardization process reached its end does not mean at all that there is no more room for proposing tools to improve further the coding efficiency. Indeed, this second contribution has been proposed out of the standardization scope; therefore, less constraints (especially those related to the complexity) have been imposed, allowing more freedom to the designed approach. Moreover, CPUs and GPUs are becoming cheaper and/or faster with the technological advancements, leading to producing bigger and more efficient algorithms.

Some difficulties arise when one tries to extend the concept of Deep Learning (DL) to the case of video coding where traditional approaches are extremely efficient. This makes the challenge of introducing DL to this domain, significantly complex. Today DL's contributions to video compression may involve the optimization of existing tools (e.g. bit rate allocation), the introduction of new tools (e.g. new prediction methods) or a new architecture. One of the objectives of this thesis is to explore the possible contributions of DL to video coding.

Among the so-called normative tools, new predictors (spatial or temporal) based on the DL seem to be the most promising approach. For example, the analysis of the prediction information cost may benefit from DL approaches, with the intention of predicting certain encoder choices at decoder level. By doing that, these choices would no longer need to be signaled.

2.2.2 Second Axis: Optimizing existing tools

The second axis, contrarily to the first one, does not seek a contribution to be added to the core algorithm of VVC. Nevertheless, this axis aims to build an optimized usage of the already existing VVC algorithm. To do so, a necessary deep study is conducted to evaluate and characterize new coding tools proposed for VVC. The analysis considers the objective coding results of the different tools, and their complexity.

All the tools integrated in VVC have improved compression efficiency but come at the cost of increased complexity. Therefore, this contribution aims to detect the tools that are efficient for all the video sequences, and the other tools that may be efficient for a specific sequence(s) or type(s) of content(s), and for a specific type of use case. This decision can be applied on each video sequence or part of a sequence based on the content and the local characteristics of the sequence.

The ultimate objective is to find the best possible compromise between the provided compression efficiency and the complexity imposed by the tools. Thereby, an optimization framework is designed to determine an efficient technique of activating the new coding tools. The determination of the tools and algorithms set-up can be carried out either by employing artificial neural networks or without any Artificial Intelligence (AI) technique.

2.3 Structure of the manuscript

The rest of the manuscript is organized as follows. This introduction is followed by a detailed background ranging from a generic presentation of video representation and coding to a more concrete description of the last video coding standard: VVC. In **Chapter 3**. we provide a summary of all the materials needed to understand the contributions of the thesis, namely, machine and deep learning methods.

Then, in **Chapter 4** a wide state-of-the-art, related to the contributions of this thesis, is discussed. Therein, several works in the intra or inter predictions are mentioned. In addition, a wide investigation of proposed AI-based techniques in video coding is reported. Finally, we give an overview of previous findings in relation with encoder optimization methods.

Equipped with this fundamental knowledge, we are ready to proceed the manuscript by describing the proposed contributions in this manuscript. We start by focusing on the contributions of the first axis.

In particular, **Chapter 5** details the first contribution that enhances the intra prediction mode. First, we expose the problem that our proposal is solving. Then, the motivation behind the work is presented before giving an overview of the method. Afterward, the presented novel method is detailed with a focus on its two main components: the gradient analysis generating the intra modes and the fusion of these modes. The significant coding gains are then given and analyzed.

Chapter 6 presents the second contribution of the first axis. This work targets inter prediction mode, more precisely the interpolation process. Similarly to the structure of the first contribution, this chapter begins by discussing the addressed problem followed by the motivation and the advantages of such proposal. Basically, a novel way of interpolating is proposed. Two main pillars, constituting this contribution, are explained in the chapter. As first pillar, a framework is developed for designing alternative interpolation filters. As a second pillar, we implement a scheme where the decoder infers the optimal filter. The results of our method are reported next before concluding the contribution by giving some of its future perspectives.

Then, we focus on the contribution of the second axis. **Chapter 7** starts with a detailed description of a list of switchable VVC tools adopted during the standardization. After describing the problem caused by this increasing number of adopted tools, the motivation, of customizing the tools depending on the

video contents, is justified. After presenting an overview of the method, we give a detailed explanation of the two principle frameworks contributing in reaching the ultimate objective of this proposal. The preliminary framework is built in order to define the optimization method to be followed. Once this method is validated on a set of sequences, the framework is generalized to be applicable on any sequence.

Finally, we conclude this manuscript in **Chapter 8** with a summary of the proposed methods, their results, as well as some perspectives for future works.

Chapter 3

Background

3.1 General Overview on Video Coding Standards

Modern video coding standards have been designed to efficiently transmit and store digital video with a range of data rate, picture quality, latency, random accessibility, complexity, and other criteria. It is critical to provide support for the following applications.

- Real-time conversational services, such as video telephony, video conferencing, screen sharing, and cloud gaming (an application recently pushed to the forefront by the COVID-19 pandemic). Low delay/latency and moderate complexity are important requirements for these scenarios.
- Live broadcast, for example, TV over satellite, cable, and terrestrial transmission channels where the focus is on picture quality, constant or moderately varying channel bit rate, moderate delay, and frequent random access points for channel tuning-in and channel switching.
- Video on demand, for example, video streaming over Internet Protocol (IP). Here, what matter most is the picture quality, bit rate, and adaptation to transmission channels.
- Capture, streaming, and storage by digital cameras, for example, as used in smartphones, drones, actions, security cameras, and professional camera systems.

End-to-end video compression technology involves, an encoder to compress the video from the source into a bit stream and a decoder to decompress this bit stream for consumption at the destination. The term "codec" refers to the combination of an encoder and a decoder.

All video coding standards since H.261 (1988) [38] have been based on the so-called hybrid video coding scheme presented in Figure 3.1. The word "hybrid" refers to a combination of two methods for reducing video signal redundancy, namely prediction and transform coding with quantization of the prediction residual.

Although prediction and transforms reduce redundancy in the video signal by decorrelation, quantization decreases the data of the transform coefficient representation by reducing their precision. Ideally, by removing only imperceptible details, it serves to reduce irrelevance in the data. The two most recent standards, HEVC and VVC, likewise employ this hybrid video coding design concept. The reader is recommended to [25] for a more detailed overview of the previous standards, which span from H.120 [39] through AVC and include H.261, MPEG-1 Video [40], H.262/MPEG-2 Video [41], H.263 [42], and MPEG-4 Visual [43].

A modern hybrid video coder can be characterized by the referred building blocks in Figure 3.1 and described in the following.

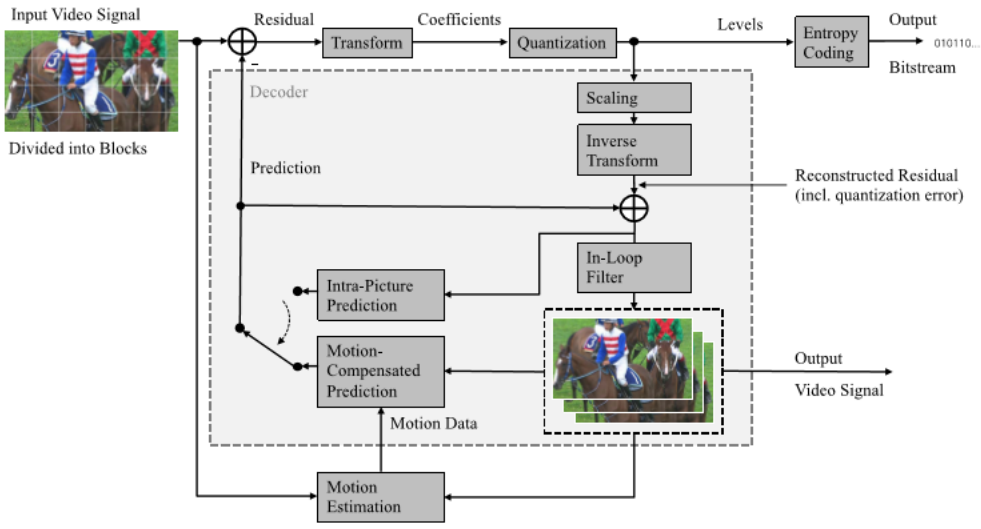


FIGURE 3.1: Block diagram of a hybrid video encoder, including the modeling of the decoder within the encoder [1].

3.1.1 Block Partitioning

For the prediction and transform operations, block partitioning is applied to divide the image into smaller blocks. The first hybrid video coding standards used a fixed block size, with luminance (luma) prediction areas using 16×16 samples and transformations using 8×8. Partitioning became a key element of the design focus starting from H.263, and especially with AVC. Block partitioning has developed through the generations to become more flexible by introducing more and varied block sizes and shapes to allow for adaptability to local region data. In the prediction stage, this allows an encoder to trade between high prediction accuracy (using small blocks) for a low data rate for the side or prediction information to be signaled (using big blocks).

In fact, testing various combinations and deciding which to select is the main origin of the encoder complexity. Accordingly, as the number of options for splitting an image into blocks is increasing, the complexity grows in comparison to encoders with fixed size or restricted partitioning set. Recent standards, on the other hand, have been able to give a significant degree of flexibility due to fast partitioning algorithms and advances in computing power. AVC, HEVC, and VVC all use tree-based partitioning structures with different depth levels and blocks as leaf nodes, with VVC further allowing non rectangular partitions to be used.

3.1.2 Intra-picture prediction

Intra prediction takes advantage of the spatial redundancy that occurs inside a picture by deriving a block's prediction from previously coded/decoded, spatially nearby reference samples. With AVC, this type of spatial sample domain prediction has been introduced, whereas earlier standards were relying on a simplified transform-domain prediction. Three types of prediction modes are used in AVC: "DC", "planar", and "angular", all of which use neighboring samples of previously decoded blocks to the left and/or above the block to be predicted. The first, known as DC mode, averages neighboring reference samples and uses that value as a prediction for the entire block, for each sample. The planar mode, on

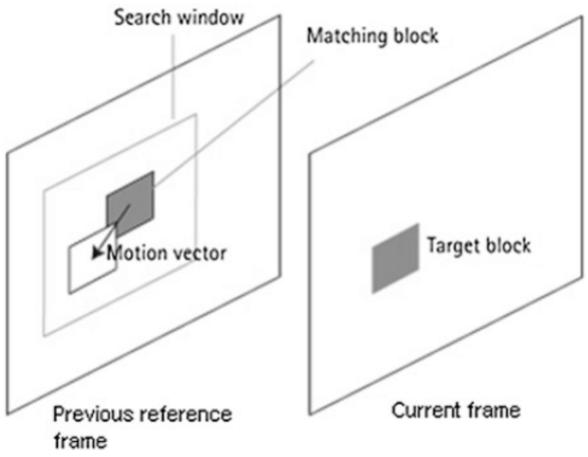


FIGURE 3.2: Motion Compensation [2].

the other hand, represents the samples to be predicted as a plane by using position-dependent linear combinations of the reference samples. The third option, the angular modes, interpolates the reference samples along a specific direction/angle. The vertical angular mode, for example, just duplicates the above reference samples along each column. For example, HEVC has increased the number of angles from 8 to 33. Further, VVC does not only increase the number of modes but also adds new approaches, such as a Matrix-based Intra-picture Prediction (MIP) that uses machine learning [44].

3.1.3 Motion-compensated or inter-picture prediction

Motion Compensation (MC) exploits the temporal redundancy between frames to improve the encoding efficiency. Block-based motion compensation is a key concept, in which the picture is divided into blocks, and for each block, an encoder searches reference pictures to find a best matching block as shown in Figure 3.2. The best matching block is called the prediction of the corresponding block and the difference between the original and the prediction signal is coded by various means, such as transform coding, and transmitted to a decoder. The relative position of the prediction with respect to the original block is called a Motion Vector (MV) and it is transmitted to the decoder along with the residual signal. This concept of translational motion compensation has been later generalized by using fractional-sample MV accuracy with interpolation (with half-sample precision in MPEG-1 and MPEG-2 videos and quarter-sample precision from MPEG-4 Visual onward), averaging two predictions from one temporally preceding and one temporally succeeding picture (bidirectional prediction in MPEG-1 and MPEG-2 videos), or from several reference pictures with arbitrary temporal positions (in standards since AVC). Furthermore, the use of multiple reference pictures from various temporal positions allows for hierarchical prediction structures inside a Group Of Pictures (GOP), which increases coding efficiency even further. VVC, the most recent standard, goes even farther than the translational motion model by approximating affine motion and employing a different motion estimation technique for motion refinement at the decoder.

3.1.4 Transformation

Transformation decorrelates a signal by converting it from the spatial domain to a transformed domain (usually a frequency domain) using an appropriate transform basis. The prediction residual (regardless of

whether it originates from inter- or intra-picture prediction), that is, the difference between the prediction and the original input video signal, is transformed in hybrid video coding standards, as shown in Figure 3.1. In the transform domain, the relevant information is generally concentrated in a limited number of coefficients. To reconstruct the residual samples at the decoder, an inverse transform must be applied. The Karhunen–Loève transform (KLT) is considered as an optimal decorrelator but depends on correlation properties of the input signal that are typically unknown at the decoder. KLT is an example of a transform basis. Another example is the Discrete Cosine Transform (DCT), which has been used for hybrid video compression since H.261 and is also utilized in the well-known JPEG picture compression [45]. For strongly correlated auto-regressive sources, the DCT decorrelates approximately as well as the KLT and is easier to compute. In later standards, such as H.263 version 3 and AVC, integer-based reduced complexity transforms (commonly referred to as DCTs) are exploited, despite the fact that a genuine DCT uses trigonometric basis functions involving irrational numbers and enables extra factorizations.

3.1.5 Quantization

Quantization is the process of reducing the accuracy of an input value or a group of input values in order to reduce the amount of data required to represent the values. Quantization is generally performed to individual transformed residual samples, that is, to transform coefficients, resulting in integer coefficient levels. This technique is done to the encoder, as shown in the figure of the hybrid video scheme. Inverse quantization, known also as scaling, is the equivalent technique, which recovers the original value range without restoring the precision. In the block diagram for hybrid video coding, Quantization is the most responsable part causing distortion because of the precision loss. Quantization and scaling can be seen as a rounding operation with a step size that controls precision. The step size is generated from a so-called Quantization Parameter (QP) that regulates the fidelity and bit rate in current video coding standards. A greater step size (larger QP) decreases the bit rate but degrades the quality, resulting in video images with blocking artifacts and blurred details for example. Typically, each sample is quantized separately, consisting thus a process known as scalar quantization. On the other hand, vector quantization processes a group of samples at the same time, for example, by mapping a block onto a vector from a codebook. Prior to HEVC, all contemporary video coding standards used solely scalar quantization, at least at the decoder side. Knowing that the quantization of a sample is dependent on the states of previous samples, HEVC includes a trick known as sign data hiding, which can be viewed as a form of vector quantization. As for VVC, it introduces Dependent Quantization (DQ), which can be interpreted as a kind of sliding-block vector quantization. From the decoder perspective, advanced approaches for optimized encoding using previous standards can also be seen as vector quantization while appearing to be scalar quantization.

3.1.6 Entropy Coding

Entropy Coding assigns codewords to a discrete-valued set of source symbols by taking into account their statistical properties, that is, relative frequency. In order to approach the entropy, all recent video coding standards employ Variable-Length Coding (VLC) tables, which allocate shorter code words to symbols that appear more frequently. The Huffman coding methodology was used to generate code word tables in previous standards (with minor adjustments). The vast bulk of data, including control data, motion data, and coefficient levels, is generally encoded and decoded using VLC. By employing a Context Adaptive VLC (CAVLC) method, AVC greatly enhanced the VLC technique for coefficient level coding. A context is defined by the value (or a combination of values) of prior symbols, which may be used to switch to a

VLC table created specifically for that context. AVC was the first video coding standard to use Context-Adaptive Binary Arithmetic Coding (CABAC) as a second, more efficient entropy coding approach. CABAC continues to employ VLC tables to map symbols such as coefficient levels, to binary strings (code words). However, the binary strings are not directly written to the bit stream, but, instead, each bit in the binary string is further coded using binary arithmetic coding with context-adaptive probability models. CABAC has become the sole entropy coding method in HEVC and VVC standards because of its great efficiency.

3.1.7 In-loop filtering

In-loop filtering is a filtering procedure that is applied to the reconstructed picture, which is made up of the reconstructed residual signal (which includes quantization error) and the prediction. After in-loop filtering, the reconstructed picture can be stored and used as a reference for inter-picture prediction of subsequent pictures. This impact on other pictures inside the hybrid video coding prediction loop gave rise to the term "in-loop filtering". The purpose of filtering is mainly the reduction of the visual artifacts and reconstruction errors. H.263 version 2 was the first standard to employ a deblocking in-loop filter, which was included to AVC version 1 as a fundamental feature. This filter was designed to be adaptive to quantization fidelity, allowing it to attenuate blocking artifacts caused by the quantization of block-based prediction residuals while maintaining sharp edges in the picture content. In HEVC, after deblocking, a second nonlinear in-loop filtering stage called sample adaptive offset filtering is added to reduce ringing and banding artifacts. An Adaptive Loop Filter (ALF) was added as a third filter in the VVC standard, where the filter coefficients are generally determined by minimizing the reconstruction error using a Wiener filter optimization approach. Furthermore, at the in-loop processing stage of VVC, another tool named as Luma Mapping with Chroma Scaling (LMCS) might be used before the others.

The next section describes, in more detail, the recent improvements to prior hybrid video coding schemes for the most modern video coding standard, VVC.

3.2 Versatile Video Coding (VVC)

This section describes in detail the most recent standard 'VVC', formally approved as ITU-T H.266 and ISO/IEC 23090-3. The VSEI standard [46], which specifies the VUI and part of the SEI messages used with VVC bit streams and is based on ITU-T H.274 and ISO/IEC 23002-7, was created and approved at the same time [46]. For HEVC and AVC, these characteristics are specified explicitly within the same video coding standard that specifies the coding tools. Apart from achieving significant bit-rate reductions over its HEVC and AVC predecessors, for camera-content video sequences, VVC was developed to provide and enhance functionalities and coding efficiency for a broader variety of existing and upcoming applications, such as:

- **Video beyond the standard and high definitions** is substantially improved by more and bigger block structures (see Section 3.2.2.1.1) for higher resolutions, as well as a luma adaptive deblocking filter designed for HDR video characteristics (see Section 3.2.2.1.6). In addition, profiles supporting chrominance formats beyond 4:2:0, such as 4:2:2 and 4:4:4, were defined in the first version of VVC.
- **Computer-generated or screen content** motivated the inclusion of techniques derived from the HEVC SCC extensions, such as IBC, Block-level Differential Pulse Code Modulation (BDPCM),

ACT, palette mode coding, and full-sample adaptive MV precision, as well as an alternative residual coding for transform skip modes (see Section 3.2.2.1.7).

- **Ultralow-delay streaming** is facilitated by built-in Gradual Decoding Refresh (GDR) handling that can avoid bit rate peaks caused by intra-picture coded pictures and virtual boundaries for better GDR compatibility (see Section 3.2.3.1).
- **Adaptive streaming** with resolution changes benefits from Reference Picture Resampling (RPR) (see Section 3.2.3.6), which provides switching resolutions inside a CVS by resampling reference pictures to the picture resolution of the current picture for the purpose of inter-picture prediction.
- **360° video for immersive and augmented reality applications** is efficiently coded by motion-compensated prediction that wraps around picture boundaries, eliminating in-loop filtering across virtual boundaries (see Section 3.2.2.1.8), and subpictures with boundary padding (see Section 3.2.3.5).
- **Multilayer coding** is already supported in VVC's first version, thanks to a complexity-constrained, single-layer-friendly method that allows for temporal, spatial, and quality scalabilities, as well as multiview coding (see Section 3.2.3.7)

Section 3.2.1 discusses the first steps toward developing a new standardization project with compression efficiency beyond HEVC, as well as a brief overview of the VVC standard development. Then, in Section 3.2.2, the innovative coding techniques in VVC that contribute to overall bit-rate savings are explained. Finally, in Section 3.2.3, advances and novelties in the systems and transport interfaces are presented.

3.2.1 Standardization and Developpment

VVC's development may be divided into two phases, as shown in the following. The first phase, which began in 2015, was mainly focused on investigating the potential and the possibility for enhanced coding efficiency without taking into account practical complexity restrictions. The exploratory phase proved that technology with sufficient compression capabilities beyond existed, prompting the kick off of the official standardization phase (the second phase), which will span from 2018 to 2020. This phase aimed to retain, even improve, compression efficiency while taking into account all aspects of implementation and complexity, as well as meeting a broader variety of application scope.

3.2.1.1 Exploration Phase (2015–2017)

The need for even more efficient compression than the current HEVC standard motivated ITU-T VCEG and ISO/IEC MPEG to join forces again in October 2015 for exploring new coding technology in a new team called the Joint Video Exploration Team (JVET). By the end of 2017, the JVET had developed the Joint Exploration Model (JEM) software codebase [47], which exhibited up to a 30% bit-rate savings compared to HEVC, based on VCEG Key Technical Area (KTA) software.

The coding efficiency improvements achieved in this exploration effort were deemed sufficient evidence to issue a formal Joint Call for Proposals (CfP) for new video coding technology in October 2017. It was agreed that, once the drafting of a formal standard began, the joint team would be renamed to reflect its change of mission, becoming the Joint Video Experts Team, keeping the same JVET abbreviation.

3.2.1.2 Standardization Phase (2018–2020)

The CfP received proposals from 32 organizations for the coding of three types of video content: Standard Dynamic Range (SDR), high dynamic range (HDR), and 360° video [48]. In most test scenarios, all contributions were superior to HEVC in terms of subjective quality, and some submissions were superior to the technology previously explored in the JEM framework in a relevant number of situations, according to an independent subjective review made in April 2018. VVC development began in April 2018 with the first draft of the specification document and test model software, after an examination of the best-performing ideas among all submissions. Following the CfP's where a significant number of coding tools were submitted, it was decided to start with a "clean slate" approach.

This first draft only included advanced Quad-Tree with Multi-Type Tree (QT+MTT) block partitioning, which was identified as a common element in almost all proposals, especially that its implementation would have a major impact on the design of all other block-based coding tools. On top of that, additional coding tools from the CfP responses, as well as new ones, were thoroughly investigated in "core experiments" in terms of coding efficiency and implementation complexity. These tools were added to the VVC architecture in instances when a reasonable compromise between coding efficiency and complexity was observed.

3.2.2 Coding Tools

VVC employs a block-based hybrid video coding architecture similar to those of its predecessors. Despite the fact that the same architecture is used, new tools are integrated in each fundamental building block to improve the compression.

The coding tools in HEVC version 1 and VVC version 1 are summarized in Table 3.1. The VVC tools are presented in detail in the following sections.

3.2.2.1 Block Partitioning

In VVC, the QT+MTT partitioning scheme, which uses quaternary splits followed by binary and ternary splits, replaces the Quad-Tree with various partition unit types used in HEVC, removing the concept of splitting a CU into PUs and TUs and allowing for more flexible CU partitioning. Due to binary and ternary tree splits, rectangular PU shapes are substituted by rectangular CU shapes. Multiple TUs in a CU can only occur from an implicit split of CUs that are longer than the maximum transform length and from CUs with intra sub-partitions. Furthermore, RQT-based TU partitioning has been removed (see Section 3.2.2.3.3). In addition, the maximum CTU size has been extended to 128 luma samples, while the maximum supported transform length has been increased to 64. The block partitioning structure for VVC is formed by this tree-based CU partitioning technique, which is supplemented by the use of a separate tree for the chroma components and the concept of virtual pipeline data units. This is discussed later in a detailed section.

Coding Quad-Tree with multitype tree:

A Quad-Tree structure is used to partition a CTU. A multitype tree structure can then further split the Quad-Tree tree leaf nodes. Vertical binary splitting, horizontal binary splitting, vertical ternary splitting, and horizontal ternary splitting are the four forms of splitting in the multitype tree structure. The multitype tree leaf nodes are termed CUs, and this segmentation is utilized for prediction and transform processing without any additional partitioning until the CU is too large for the maximum transform length. This means that in most situations, the CU, PU, and TU have the same block size in the QT

Building block of the hybrid scheme	HEVC version 1	VVC version 1
Block Partitioning	64x64 max CTU size	128x128 max CTU size
	Quadtree (QT)	Quadtree& Multi-Type Tree (QT+MTT)
	Coding Units (CU)	Coding Units (CU)
	Prediction Units (PU)	Chroma Separate Tree (CST)
	Residual Quadtree	Local Dual Tree
	Transform (RQT)	Virtual Pipeline Data Units (VPDUs)
Motion Compensated or Inter-Picture Prediction	Merge Mode	Extended Merge Mode and MVP with
	Advanced MVP	- History-based MV Prediction (HMVP) - Pairwise Average MVP candidate - Sub-block-based Temporal MVP (SbTMVP) - Merge with MVD (MMVD) - Symmetric MVD (SMVD)
		Adaptive MV Resolution (AMVR)
	8-taps Interpolation Filters	8-taps Interpolation Filters (IF) 1 6-tap alternative Half-pel IF
		Geometric Partitioning Mode (GPM)
		Bi-prediction with CU-level (BCW)
		Combined Intra/Inter-picture Prediction (CIIP)
		Decoder-side MV Refinement (DMVR)
		Bi-Directional Optical Flow (BDOF)
		Affine Motion
		Pred Refinement with OPtical Flow (PROF)
Intra-Picture Prediction	33 Angles	93 Angles
	Linear interpolation	4-tap IFs (2 sets of filters)
	DC, Planar	DC, Planar
		Position-Dependent Pred. Combination (PDPC)
		Multiple Reference Lines (MRL)
		Matrix-based Intra-picture Prediction (MIP)
		Cross-Component Linear Model (CCLM)
		Intra Sub-Partitions (ISP)
Transform and Quantization	Square transforms(max 32x32)	Non-square transforms (max 64x64)
		Multiple Transform Selection (MTS)
		Non-Separable Secondary Transform (LFNST)
		Sub-Block Transform (SBT)
		Adaptive chroma QP offset
	Sign Data Hiding	Sign Data Hiding (SDH)
		Dependent Quantization (DQ)
		Joint coding of chroma residuals (JCCR)
Entropy Coding	CABAC	CABAC with high-accuracy multi-hypothesis probability estimates
	Coefficient Groups	Additional coefficient group sizes
	Reverse diagonal, hor. and ver. coefficient scan	Reverse diagonal coefficient scan only
		Improved probability model selections for absolute transform coefficient levels
In-loop Filtering	Deblocking	Luma Mapping with Chroma Scaling (LMCS) Deblocking Boundary Handling Modifications Deblocking Long Filter Luma Adaptive Deblocking
	Sample Adaptive Offset (SAO)	SAO Adaptive Loop Filter (ALF) Cross Component ALF (CC-ALF)
Special Modes	PCM 4x4 TS Trans. Quant. Bypass Quantization Bypass	4x4-32x32 Transform skip (TS)
Screen Content Coding		Block-Level Differential PCM (BDPCM)
		Transform Skip Residual Coding (TSRC)
		Intra-picture Block Copy (IBC)
		Palette Mode
		Adaptive Color Transform (ACT)
360-degree Video Coding		MV Wrap-Around
		Virtual Boundaries for in-loop filtering

TABLE 3.1: Overview of the tools adopted in HEVC and VVC.

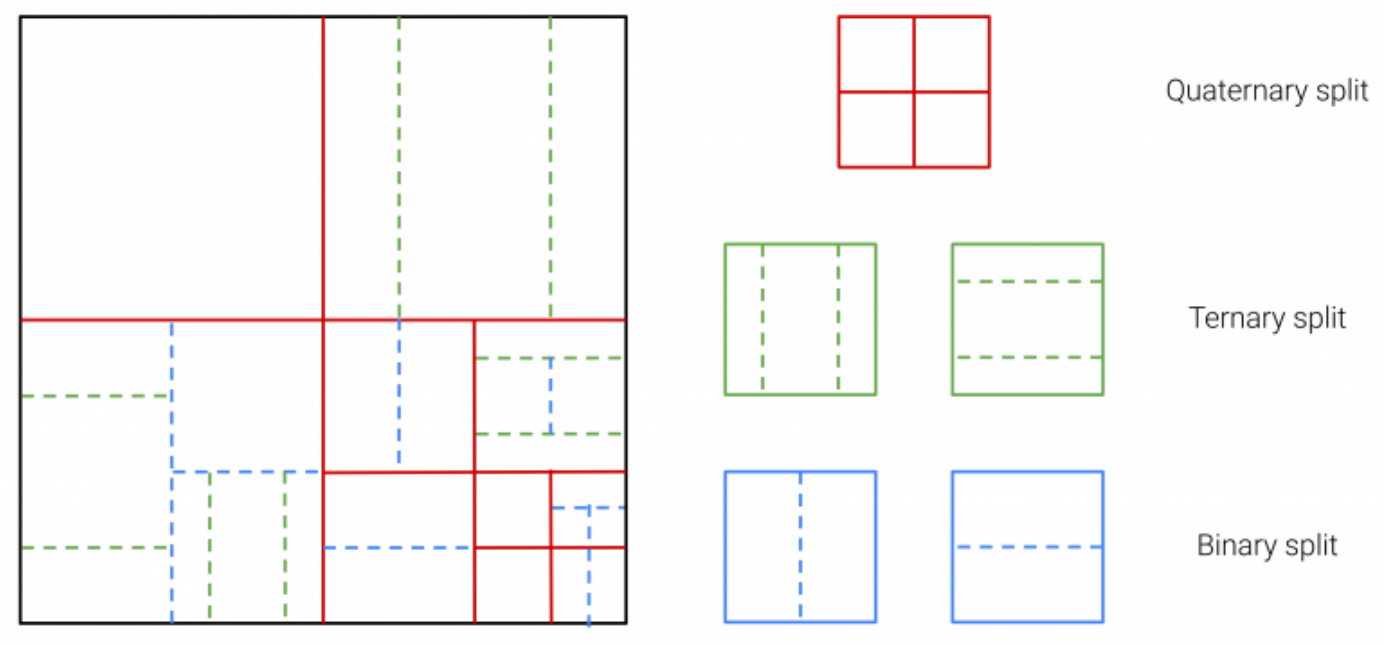


FIGURE 3.3: Example of Quad-Tree with nested multitype tree coding block structure.

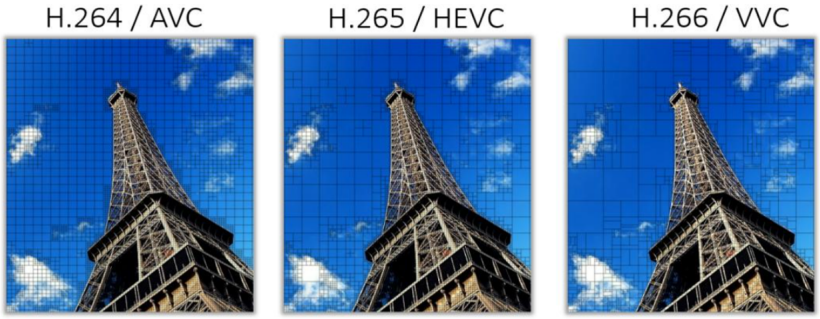


FIGURE 3.4: Comparison between partitionings of AVC, HEVC and VVC.

+MTT coding block structure. Exceptions arise when intra sub-partitions (see Section 3.2.2.3.3) or Sub-Block Transformations (SBTs) (see Section 3.2.2.4.4) are used, in addition to when the CU is too big for the maximum transform size. VVC supports non square TBs as well as square TBs. Figure 3.3 depicts a CTU divided into numerous CUs using a QT+MTT coding block structure, with solid block edges representing Quad-Tree partitioning and dotted block edges representing multitype tree partitioning with binary or ternary splits. The CU may be as large as the CTU or as small as 4x4 in terms of luma sample units. As seen in the example overlay in Figure 3.4, compared to the partitioning of the previous standards, the QT+MTT partitioning provides a very flexible block structure that can adapt to local characteristics. In the case of inter-picture prediction, there is also the option to split a CU into two non rectangular prediction block partitions at the multitype tree's leaf node, using one of 64 geometric partitioning modes (see Section 3.2.2.2.2).

Chroma separate tree:

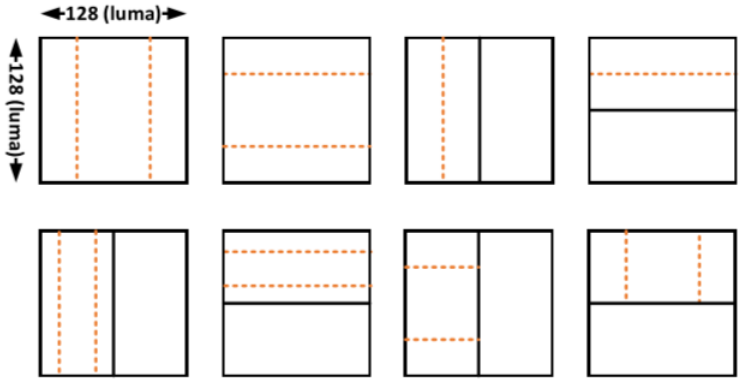


FIGURE 3.5: Disallowed ternary splitting and binary splitting in VVC when the luma coding block width or height is 128 to enable 64×64 VPDU operation.

The coding tree scheme in VVC allows luma and chroma to have their separated own partitioning tree structures. The luma and chroma CTBs in one CTU must have the same coding tree structure for inter-picture coded slices. The luma and chroma can, however, have distinct trees for intra-picture coded slices. In a case where the video is not monochromatic, a CU in an intra-picture coded slice may consist of only the luma component, only two chroma components, or only all three components, but a CU in an inter-picture coded slice always comprises coding blocks of all three color components.

Local dual-tree:

In typical video encoder and decoder implementations, when many small block are present in the coded picture (specifically, small intra-picture coded blocks, which must be decoded sequentially), the average processing throughput drops. In the single-coding tree structure, a CU can be as small as 4×4 luma samples in the single-coding tree structure, resulting in 2×2 chroma coding blocks if the video is 4:2:0. A local dual-tree structure is utilized to prevent small chroma blocks. Additionally, a second tree is used locally for the chroma when necessary to prevent small chroma blocks. The local dual-tree architecture prevents chroma intra-picture coded coding blocks with a size of less than 16 chroma samples or with 2×N sizes.

Virtual pipeline data units (VPDUs):

VPDUs are picture block units that must be stored in memory for processing during decoding. Multiple pipeline stages can be operated at the same time in hardware decoders to handle successive VPDUs. In most pipeline stages, the VPDU size is typically proportional to the memory buffering size, therefore it is important to keep the VPDU size small. For CUs of 128×128 luma samples, the QT+MTT scheme, ternary tree, and binary tree splits could have resulted in a VPDU size that was considered too difficult to handle. To maintain the VPDU size at 64×64 luma samples, normative partitioning restrictions with syntactic signaling modification are applied. In this way, specific splits for CUs with a width or height of 128 are prevented, as illustrated by dashed lines in Figure 3.5.

3.2.2.2 Inter-Picture Prediction

Many of the inter-picture prediction features from HEVC are retained and enhanced in VVC, including the two most important motion information coding methods discussed earlier: AMVP and the merge mode. VVC additionally employs, from HEVC, the eight-tap high-precision motion compensation Interpolation

Filter (IF) for luma fractional positions and the four-tap IF for chroma fractional positions. In addition to the fundamental characteristics, VVC introduces additional coding techniques to improve the efficiency of inter-picture prediction. One of these techniques is subblock-based motion inheritance, in which the current CU is divided into equal-sized subblocks (8×8 luma samples) and the MV for each subblock is derived using temporally colocated blocks in a reference picture. To describe higher-order motion, such as scaling and rotation, a local CU-based affine motion model is employed, with just one set of parameters coded per CU and motion compensation executed separately every 4×4 subblock using six-tap IFs. In some modes, such as the affine mode, VVC improves the MV precision to 1/16 luma sample to improve prediction efficiency for video content with locally varying and non translational motion, whereas HEVC only utilizes quarter luma-sample precision [8]. A block-level AMVR technique is implemented on top of the higher precision MV representations to adjust the balance between prediction quality and bit cost overhead for MV signaling. To better match motion at object borders, the geometric partitioning mode divides a CU into two non rectangular partitions. The Bi-prediction with CU-level Weights (BCW) mode goes beyond normal averaging taking into consideration the weights of the two prediction signals at the block level. Decoder-side MV Refinement (DMVR) and Bi-Directional Optical Flow (BDOF), which enhance motion compensation without adding bit overhead, are added to further improve prediction quality. Finally, VVC offers a method for combining inter-picture and intra-picture prediction to provide a final prediction.

A merge candidate list is built for a CU coded in merge mode, and an index is sent to designate which candidate MVP is used to make the prediction. The merging candidate list in VVC is made up of five different sorts of candidates in the following order:

- MVPs from spatially adjacent CUs.
- Temporal MVP (TMVP) from colocated CUs.
- History based MVP from a FIFO table.
- Pairwise average MVP.
- Zero MVs.

The merge list's length can be found in SPS, with a maximum length of 6. The usage of MVs from spatial neighboring CUs and colocated CUs is the same as it is in the HEVC merge candidate list.

History-based MV Prediction (HMVP):

HMVP provides candidates outside the local spatial–temporal region, enabling the use of MV data from more distant CUs. Both merge and AMVP candidate list generation methods can employ HMVP candidates. A table of MVP candidates for the current CU stores the motion information of previously coded blocks. During the encoding/decoding process, a table containing multiple HMVP candidates is kept; and when a new CTU row is found, the table is reset; ergo, all candidates are deleted. Therein, the related motion information is updated in a First-In-First-Out (FIFO) manner whenever an inter-picture coded CU exists, except CUs coded with affine mode, geometric partitioning or subblock-based TMVP. The HMVP table size is 6.

The pairwise average MVP candidate: is calculated by taking the average of the MVs of the first two candidates in the merge candidate list. Each RPL's averaged MVs is calculated separately. If the merge list is not complete after the pairwise average merge candidate is added, zero MVPs are appended to the end until the merge list reaches the maximum merge candidate number.

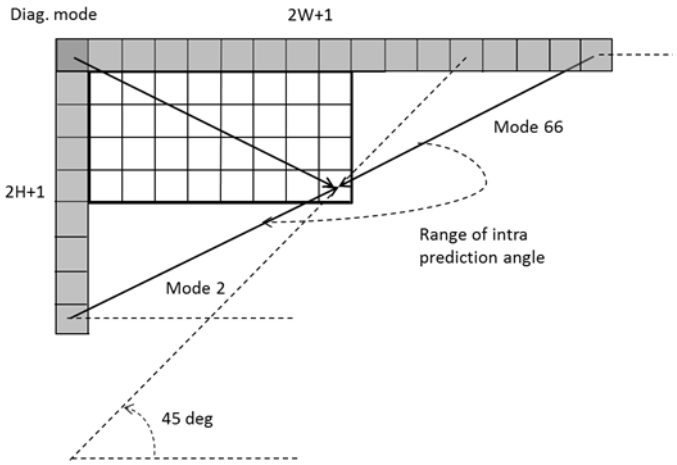


FIGURE 3.6: Wide-angular intra prediction [3].

3.2.2.3 Intra-Picture Prediction

Reference samples in neighboring blocks to the left and above the present block, that have been previously decoded (prior to in-loop filtering) in the same picture, are used to predict the samples of an intra-picture coded block. Planar, reference sample averaging (also known as the DC mode) and 33 directional angular modes are among the 35 intra-picture prediction modes used by HEVC. VVC extends the possibilities by providing additional tools, which are detailed below.

93 intra-picture directional prediction angles: For each luma coding block size, HEVC defines 33 angular prediction directions in a clockwise orientation, ranging from 45 to 135 degrees. VVC effectively doubles the angular precision to provide 65 directional angular modes within the same angular range, in addition to the DC and planar. On top of that, additional 28 "wide-angle" prediction modes are available in this new standard, for non-square blocks beyond the mentioned angular range. Figure 3.6 illustrates an example of "wide-angle" prediction modes for an 8×4 (W×H) block. Angular prediction modes referencing beyond 2H +1 samples from the shorter side to the left (close to 45°) are replaced with wide-angle prediction modes referencing up to 2W +1 samples from the longer side above (beyond -135°). When W > H, there are 14 possible wide angles, and when H > W, there are another 14, bringing the total number of wide angles to 28. The original mode indices are adaptively remapped to the indices of wide angular modes based on the block size after parsing to signal the replacement modes. The total number of intra-picture prediction modes is constant for any block size, 67. Although, the mode coding mechanism is the same for all block shapes, the inclusion of wide angles for non square blocks raises the total number of supported directions to 93, and therefore the total number of modes to 95. Lastly, VVC uses an "MPM" list with six candidates to efficiently code the decision among the 67 options.

Two sets of four-tap Interpolation Filters IFs: The prediction samples placed at fractional sample locations for the angular modes are generated with various frequency cutoffs and 1/32-sample precision. The two sets of four-tap IFs, one of which is a DCT-based IF (DCTIF) and the other of which is a four-tap SIF, replace the lower precision linear interpolation used in HEVC. The DCTIF is designed in the same way as the one used in HEVC and VVC for chroma component motion correction. The SIF is obtained by convolving the two-tap linear IF with [1 2 1]/4 filter. The IF is chosen based on the block size as well as the angular distance between the horizontal and vertical modes. In general, the

DCTIF sharpening is applied for smaller blocks and modes around the horizontal and vertical directions, where the correlation between the reference and original samples is stronger. Luma reference samples are smoothed using a [1 2 1]/4 filter for non fractional diagonal angles and chosen wide angles for blocks with more than 32 samples. This is a simplification compared to HEVC, where an extra strong smoothing can be performed based on the "flatness" of the reference samples. Furthermore, in luma blocks, reference sample smoothing is applied only to integer slope modes, not fractional slope modes, so that it is not cascaded with interpolation filtering.

3.2.2.4 Transforms and Quantization

In HEVC, an integer approximation of the DCT type-II transform is used as the major transform for residual signals with square block sizes ranging from 4×4 to 32×32. This applies for the latter block sizes with the exception of 4×4 intra-picture prediction residual blocks, which are transformed using an integer approximation of the DST type-VII transform. Sign data hiding can be used to expand the conventional uniform reconstruction quantizer design for scalar quantization of the transformed residual in HEVC. VVC provides additional techniques to produce greater energy compaction of residual signals and further minimize the quantization error of the transformed coefficients, which will be discussed in the following.

Non-square transforms In VVC, various length transform kernels sin horizontal and vertical directions are available for non-square TBs. The maximum transform size has been increased to 64×64 in order to improve energy compression for residual signals in large smooth regions.

3.2.2.5 Entropy Coding

CABAC is employed as the single entropy coding method in VVC, similarly to HEVC. Nonetheless, the CABAC design in VVC has a number of coding efficiency improvements when compared to the HEVC architecture. This section goes through the modifications to the two major elements of entropy coding: the CABAC engine and transform coefficient coding.

CABAC engine with multihypothesis probability estimate: A table-based probability transition procedure between 64 distinct representative probability states is used by the CABAC engine in AVC and HEVC. Prior to the new interval range calculation, the range reflecting the state of the coding engine is quantized to a set of four values. To estimate the values of the new probability interval range, the state transition is accomplished using a table containing all the precomputed values. The main concept is retained in VVC, but the binary arithmetic coder is replaced with a multihypothesis probability update model. This model is based on two probability estimates $P0$ and $P1$ that are associated with each context model and updated separately with varying adaptation rates. The probability estimate P, used for the interval subdivision in the binary arithmetic coder, is the average of the estimates from the two hypotheses. Based on the statistics of the associated binary events, the $P0$ and $P1$ adaption rates for each context model are pre-trained.

Improved transform coefficient coding: In HEVC, a coding block's transform coefficients are grouped into Coefficient Groups (CGs or subblocks). Each CG contains the coefficients of a 4×4 subblock inside a square, power-of-2 sized TB. The notion of CGs is also used by VVC for coefficient coding. Due to narrow luma TBs and tiny chroma TBs, additional CG sizes, namely 1×16, 16×1, 2×8, 8×2, 2×4, and 4×2 are added to the traditional 4×4 CG. A single reverse diagonal scan order is used to code the CGs inside a TB and the transform coefficients within a CG. The transform coefficient levels are coded using a mixture of various binarizations, similar to HEVC. This comprises Golomb–Rice coding of the remaining absolute values and truncated unary coding with a cascade of flags. The role of these flags is to

indicate if the absolute value is > 0 (significant), > 1, or > 2. To make the state transition for DQ easier in VVC, the shortened unary portion has been modified by adding an extra parity flag. Compared to HEVC, VVC offered a more sophisticated probability model selection for the syntax components linked to absolute values of transform coefficient levels, based on the values of the absolute levels or partially reconstructed absolute levels in a local neighborhood template. Two neighboring places to the right, two below, and one below-right relative to the current scan point make up the template.

3.2.2.6 In-Loop Filtering

To change the representation domain and relieve different forms of artifacts, a remapping operation and three in-loop filters can be applied successively to the reconstructed picture. To begin, a novel sample-based technique known as LMCS is carried out. The blocking artifacts are then reduced using a deblocking filter. The deblocked picture is then subjected to SAO to reduce ringing and banding problems. Finally, by reducing additional possible distortions produced by the quantization and transform operations, an ALF can be used. The deblocking filter design is similar to that of HEVC, but with longer deblocking filters and a luma-adaptive filtering mode suited to HDR video. While SAO and deblocking are quite comparable to HEVC, LMCS and ALF are novel in comparison to prior standards. In VVC, the design of ALF is divided into two steps: 1) ALF for both luma and chroma data using block-based filter adaptation, and 2) a cross-component ALF (CC-ALF) for chroma samples.

Deblocking filter boundary handling modifications:

Except when the boundary is also a picture boundary, or when deblocking is disabled across slice, tile, or subpicture boundaries, the deblocking filter is applied to samples near to a CU, TU, and subblock boundary (which is an option that can be signaled by the encoder). For CU and transform subblock boundaries, the deblocking filtering process is done on a 4×4 grid, while for prediction subblock boundaries, it is applied on an 8×8 grid. The PU boundaries introduced by the SBTMVP and affine modes are included in the prediction subblock boundaries, whereas the TU boundaries generated by the SBT and ISP modes and transformations owing to implicit splits of big CUs are included in the transform subblock boundaries. The deblocking filter's processing order is defined as horizontal filtering for vertical edges for the full picture first, followed by vertical filtering for horizontal edges. This specific order is also adopted in HEVC allowing either several horizontal or vertical filtering processes to be applied in simultaneous threads. or may still be executed CTB by CTB with little processing delay.

Deblocking long filters: The deblocking filtering method is similar to the HEVC filtering procedure. The values of various syntax components of the two neighboring blocks control the deblocking filter's border filter strength (bS), and two thresholds, tC and β, are computed from predefined tables based on the filter strength and the average QP of the adjacent blocks. Based on β and block size, one of four scenarios is chosen for luma samples: no filtering, weak filtering, short strong filtering, and long strong filtering. For chroma samples, there are three options: no filtering, normal filtering, and strong filtering. Long strong filtering for luma samples and strong filtering for chroma samples are new additions of VVC compared to HEVC. When the samples on each side of a boundary belong to a big block, long luma strong filtering is applied. When the width of a vertical edge is more than or equal to 32, or when the height is greater than or equal to 32 for a horizontal edge, the sample belongs to a big block. In the strong filter, up to seven samples on one side of a boundary are filtered. When both sides of the chroma edge are larger than or equal to 8 (in units of chroma samples), strong chroma filtering is used, and three chroma samples from each side are filtered.

Luma-adaptive deblocking: Luma-adaptive deblocking changes the deblocking filter's tC and β based on the averaged luma level of the reconstructed samples. When luma-adaptive deblocking is enabled, an offset qpOffset is added to the average QPs of the two adjacent blocks, which is determined from the average luma level around the filtering border. A table of thresholds signaled in the SPS determines the value of qpOffset as a function of average luma level. These thresholds are generally established based on the source video content's transfer characteristics (the electro-optical transfer function and opto-optical transfer function).

3.2.2.7 Screen Content Coding Tools

The efficient coding of computer-generated video content, which has different signal characteristics than camera-captured video, is one of the goals of VVC. Lack of high-frequency sensor noise, large uniformly flat regions with sharp edges, repetitive patterns, highly saturated colors, or a limited number of distinct colors are the most common characteristics of the computer-generated contents. The current HEVC REext and SCC extensions include tools for effectively exploiting these features. These tools are also used as the foundation for the following SCC tools in VVC, with some refinements.

Block-level Differential Pulse Code Modulation (BDPCM): BDPCM aims for improved decorrelation of the screen content prediction residuals by using samplewise DPCM instead of a traditional frequency transform on the residual. The DPCM, similarly to the Residual Differential Pulse Code Modulation (RDPCM), introduced in HEVC REext, can be applied horizontally (along rows) or vertically (along with columns). The direction is explicitly signalled for intra-picture predicted CUs, and the intra-picture prediction mode is derived from it; for example, vertical DPCM indicates vertical intra-picture prediction. While in HEVC, the RDPCM can be applied to inter-picture prediction residuals, the BDPCM in VVC is restricted to intra-picture predicted CUs.

Transform Skip Residual Coding (TSRC): adjusts the CABAC entropy coding of the spatial transform skip residual block to screen-content-specific characteristics. This statistical difference was previously taken into account in the HEVC REext extensions by using a specialized context model for the flag that signals an absolute value larger than zero (the significance flag) and 180° rotation of intra-picture predicted transform skip residuals. This comprises three important features in VVC:

1. When reverse scanning diagonally from bottom right to top left, the explicit signaling of the position that indicates the first non zero value is omitted, and the scanning direction is inverted (from top-left to bottom-right), as motivated by the higher probability of trailing zeros or insignificant levels at the bottom right corner of the block (due to the lack of energy compaction by a transform), the scanning direction is inverted (from top-left to bottom-right).

2. Because of non-stationarities in the sequence of sign flags, even though the global empirical distribution is still essentially uniformly distributed, sign indicators can be coded more efficiently thanks to context models.

3. The binarization of absolute level values is altered, resulting in a higher cutoff for the unary binarization prefix, and additional context-coded "greater than X" flags, as well as a modified Rice parameter derivation for the Golomb–Rice code suffix. This is also due to the fact that the empirical distribution of spatial residuals has more non-stationarities than transform coefficients.

Intra-picture Block Copy (IBC):

Repeated patterns are used inside a picture. It is a very simple kind of motion-compensated prediction that uses integer MVs (called block vectors) to refer to previously coded areas of the same picture rather than previously coded reference pictures. The IBC in VVC is simplified in terms of reference sample buffers as compared to the HEVC SCC extensions. IBC uses the inter-picture design in HEVC with few changes, such as the RPL only including the current picture and a motion or block vector that is always in integer precision and has a limitation on the region it refers to, such as only referring to already-decoded samples. By storing reference samples in a smaller local buffer, the IBC in VVC is simplified and separated from inter-picture prediction. Only the previously coded samples in the current CTU and the CTU to its left are allowed in this buffer. Another distinction is the availability of a dedicated IBC merge mode for block vector coding, which is simpler than the VVC inter-picture merge mode. Furthermore, VVC extends the integer block vector precision from HEVC SCC to block level AMVR (see Section 3.2.2.2), but only with full or four-integer sample precision.

Palette mode:

A collection of representative color values is used to represent the sample values in a CU in Palette mode. The palette is the name given to this collection. A palette is first indicated for a CU coded in the palette mode, then a palette index is signaled for each sample in the CU. In VVC, the palette is applied separately to luma (the Y component) and chroma (the Cb and Cr components) for slices having distinct luma/chroma coding trees, with the luma palette entries holding just Y values and the chroma palette entries including both Cb and Cr values. Palette coding is applied to three color components simultaneously for slices with a single coding tree, that is, each item in the palette comprises Y, Cb, and Cr values. By signaling an escape symbol, you can also indicate a sample that is outside the palette. The quantized values of samples inside the CU that are coded via the escape technique are immediately indicated. The palette mode can only be activated in profiles that support the 4:4:4 video format, despite the fact that it may be applied to all chroma types (see Section 3.2.3.8).

Adaptive Color Transform (ACT):

ACT can be used to decrease the correlation between the three color components in the 4:4:4 chroma format, which is particularly useful for video sequences in RGB color spaces. In VVC, the ACT is identical to that of the HEVC SCC extension. In the prediction residual domain, it conducts in-loop color-space conversion by adaptively converting the residuals from the input color space (assumed to be RGB) to the YCgCo-R luma–chroma color representation[49]. The residuals of the CU are coded with the YCgCo-R transformation or in the original color space, depending on a flag set at the CU level. Given that the YCgCo-R transformation is completely reversible, it may be used for lossless coding as well. When ACT is enabled for a CVS, the maximum transform size cannot exceed 32×32 samples in order to decrease cache storage needs, as ACT involves temporarily storing all three TBs.

3.2.2.8 360° Video Coding Tools

Another objective for VVC is to make immersive video coding as efficient as possible. This includes 360 video, which is usually encoded as a 2-D image created by projection mapping from a 3-D sphere. The Equirectangular Projection Format (ERP) is one example of such a mapping, in which the sphere is projected onto a rectangle picture with certain geometric distortions, notably at the poles. The Cube Map Projection (CMP) is another mapping technique, in which the sphere is projected onto the six faces of a cube and then compacted into a single image.

To improve the coding efficiency for video pictures utilizing various projection formats, VVC has introduced the ability to identify such formats as well as the following two techniques:

1. **MV wrap-around:** When an MV points outside of the coded region, MV wrap-around allows prediction samples to "wrap-around" from the opposite left or right boundary. Because of the 360 nature of the projection mapping, which might result in a moving object that is partially at the left boundary and partly at the right boundary of a picture, the content of ERP photos tends to remain continuous throughout such a wrap-around.

2. **Virtual boundaries for in-loop filtering:** This technique prevents in-loop filtering over certain "virtual" boundaries, such as those corresponding to the CMP face boundaries in CMP pictures, rather than slice or tile boundaries. The locations of these boundaries are signaled at the CVS level.

3.2.3 Systems and Transport Interfaces

Many features of the systems and transport interfaces, as well as the related header syntax, were inherited by VVC from HEVC. The bitstream structure is identical to that of HEVC, with the exception of the elementary stream which is removed. The NAL unit syntax and NAL unit header are very identical to those of HEVC, with the exception that HEVC uses six bits for the NAL unit type field, while VVC uses just five bits, allowing for half of the maximum number of specified NAL unit types. The VPS, SPS, PPS, and SH were designed in the same way as in HEVC and include comparable types of header parameters. VVC's support for temporal scalability is the same as HEVC's. Other aspects of VVC's systems and transport interfaces are outlined here, with a focus on the differences compared to HEVC.

3.2.3.1 Random Access Support

VVC supports three types of IRAP pictures: two of them are Instantaneous Decoder Refresh (IDR) pictures and one type of Clean Random Access (CRA) picture. These are nearly identical to those found in HEVC. IDR images, also known as closed-GOP random access points, force the inter-picture prediction structure to not reference any picture before the current GOP. CRA images are less restricted since they enable some pictures to reference pictures that predate the current GOP, which are all deleted when random access is used. CRA images are also known as open-GOP random access points. VVC does not provide the BLA picture types that are used in HEVC. This is due to the fact that the basic functionality of BLA pictures may be achieved with CRA pictures and an end of sequence NAL unit. This signifies that the following picture will start a new CVS in a single-layer bitstream. Another reason for not having BLA pictures in VVC is the need to specify fewer NAL unit types than in HEVC to simplify design comprehension. This is reflected by the use of five instead of six bits for the NAL unit type field in the NAL unit header.

Another significant distinction between VVC and HEVC in terms of random access support is that VVC supports GDR in a more normative manner. Decoding a bitstream in GDR can begin with an inter-picture coded picture, and when certain parts of the picture region may not be correctly decoded at first, it can be recovered after decoding few more pictures, and the entire picture region becomes correct to decode the pictures that follows in the bitstream. AVC and HEVC can also support a form of GDR, using a recovery point indication SEI message for signaling the GDR random access points. In fact, the recovery point is signaled in the Picture Header (PH) syntax structure in VVC, and a bitstream or a CVS inside a bitstream is permitted to start with a GDR picture. This means that an entire bitstream can only include inter-picture coded pictures and cannot contain any intra-picture coded pictures. The major advantage of providing GDR support in this way is the fact of ensuring that GDR functioning is consistent. Actually, GDR allows encoders to smooth out the bit rate. This happens by distributing

intra-picture coded slices or blocks across multiple pictures that also contain inter-picture predicted slices or blocks, rather than intra-picture coding of entire pictures. By doing that, a significant end-to-end delay reduction is guaranteed to improve behavior for ultra low-delay applications like wireless display, online gaming, and drone-based applications.

The virtual boundary signaling described previously is another GDR-related element in VVC. The boundary between the refreshed region (i.e., correctly decoded region) and the unrefreshed region at a picture between a GDR picture and its recovery point can be signaled as a virtual boundary. When this boundary is signaled, in-loop filtering across the boundary is not applied. This results in avoiding any decoding mismatch for certain samples at or near the border, which is helpful for presenting the properly decoded areas throughout the GDR process.

3.2.3.2 Adaptation Parameter Set (APS)

The APS is a new type of parameter set introduced by VVC. An APS transmits picture- and/or slice-level information. This type of information can be shared by multiple slices of a picture and/or slices of different pictures, but also can change frequently from picture to picture, potentially resulting in a large number of variants. For this reason, it is not suitable to include this type of information in the PPS. Three types of parameters are included in APSs: ALF parameters, LMCS parameters, and scaling list parameters for frequency-specific inverse quantization scaling. The primary goal of APSs is to reduce signaling overhead.

3.2.3.3 Picture Header

A PH, which provides header parameters for a certain picture, is also used by VVC. There must be one PH for each image. The PH, when added, carries the parameters that should go in the SH, but with the same value for all picture slices. These include IRAP/GDR picture indications, flags indicating whether inter-picture and intra-picture coded slices are permitted, picture ordering position syntax, RPL information, deblocking, SAO, ALF, QP selection, weighted prediction control, coding block partitioning information, virtual boundaries, and colocated picture information, etc. It is common for each picture in a sequence of pictures to contain only one slice. The PH syntactic structure can be included either in the PH NAL unit or in the SH in this case to avoid having at least two NAL units for each picture. The PH is created for reducing signaling overhead in cases when pictures are divided into several slices.

3.2.3.4 Reference Picture Management

Any video coding scheme that uses multi-picture buffering with generalized inter-picture prediction requires reference picture management as a fundamental functionality. It controls the storage and removal of reference pictures from and into a Decoded Picture Buffer (DPB), as well as the ordering of reference pictures in the RPLs. VVC's reference picture management is more similar to HEVC than AVC, although it is a little more simple and robust. In VVC, as in the other standards, two RPLs, termed list 0 and list 1, are derived, but are not based on the reference picture set idea used in HEVC or the automatic sliding window method used in AVC. Instead, they are indicated more explicitly. Only the active entries may be used as reference indices for inter-picture prediction of CTUs of the current picture. Other pictures to be stored in the DPB for potential referencing by other pictures that arrive later in the bitstream are indicated by inactive entries.

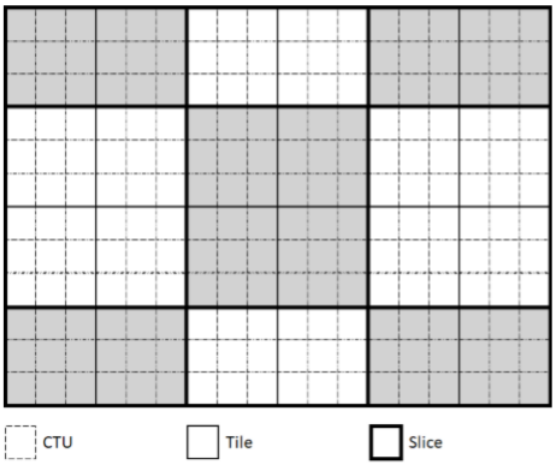


FIGURE 3.7: Picture with 18×12 luma CTUs that are partitioned into 24 tiles and nine rectangular slices.

3.2.3.5 High-Level Picture Partitioning

VVC has four separate high-level picture partitioning algorithms that are not the same as those found in HEVC. With some minor to moderate differences, VVC inherits the tiles and WPP from HEVC. The core concept of slices is preserved between the two standards, albeit it was developed in a very different way. VVC introduces subpictures, which have the same region extraction functionality as MCTSs but are designed differently for improved coding efficiency and application system friendliness. The next sections go through these differences.

Tiles and WPP:

In VVC, an image can be divided into tile rows and tile columns, much like in HEVC, and intra-picture prediction over tile boundaries is prohibited, among other things. However, by adopting a unified syntactic design for both uniform and nonuniform use cases, the syntax for signaling tile partitioning has been simplified. VVC's WPP architecture differs from HEVC in two ways: 1) the CTU row delay is reduced from two to one CTU, and 2) the signaling of entry point offsets for WPP in the SH is optional in VVC, but is obligatory in HEVC.

Slices:

Traditional slices based on CTUs (as in HEVC) or macroblocks (as in AVC) have been eliminated from VVC, meaning that each slice comprises of an arbitrary number of CTUs or macroblocks in raster scan order inside a tile or within a picture. This architectural modification is made for the following reasons. Because of the ever-increasing number and efficiency of intra-picture and inter-picture prediction mechanisms, slice-based error concealment has become practically impossible since 2003 (the year AVC v1 was published). An error-concealed picture is the decoding result of a transmitted coded picture with some data loss (e.g., loss of certain slices) of the coded picture or a reference picture. In other words, at least portion of the decoded image contains errors (e.g., because one or more reference pictures were lost or were error-concealed pictures). For example, when one of the multiple slices of a picture is lost, the error can be concealed by interpolating neighboring slices. While AVC prediction techniques improve coding efficiency, they also make it more difficult for algorithms to evaluate the quality of an error-concealed picture. Note that this was already a difficult challenge with simpler prediction mechanisms. Indeed, when a picture

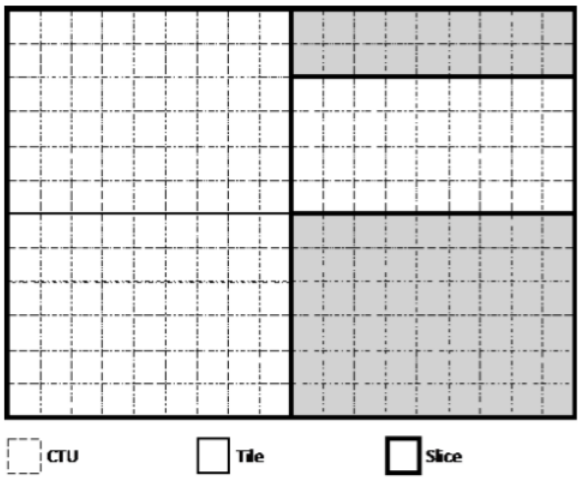


FIGURE 3.8: Picture partitioned into four tiles and four rectangular slices (note that the top-right tile is split into two rectangular slices).

is divided into several slices, advanced intra-picture prediction mechanisms perform considerably worse. In addition, network conditions have significantly improved in the meantime. As a result, only a few implementations have recently employed slices for MTU size matching. Instead, most applications that require low-delay error/loss resilience (such as video telephony and video conferencing) have come to rely on system/transport-level error resilience (such as retransmission and forward error correction). Picture-based resilience tools can be another solution (feedback-based resilience, insertion of IRAPs, scalability with uneven protection of the base layer, and so on). With all of these, it is extremely rare for a picture to be passed to the decoder that cannot be correctly decoded. Yet, when that happens, the system can afford to wait for an error-free picture to be decoded and available for display instead of experiencing frequent and long periods of picture freezing.

In VVC, there are two types of slices: rectangular slices and raster-scan slices. Rectangular slices, as the name indicates, are always rectangle in shape, and generally consist of a number of full tiles that cover a rectangular region of the picture, as seen in Figure 3.7. A rectangular slice, might be a subset of a tile, consisting of one or more consecutive, complete CTU rows inside a tile, as illustrated in Figure 3.8. A raster-scan slice is made up of one or more full tiles arranged in tile raster scan order, and therefore the region covered by a raster-scan slice is generally not a rectangle (as illustrated in Figure 3.9), though it can be.

Based on the configuration of tiles, the PPS signals the layout of rectangular slices (including the position and size of each slice). The SH contains information on which tiles are included in a raster-scan slice.

Subpictures:

As previously stated, the functionality of subpictures has been included during the development of VVC. As illustrated in Figure 3.10, each subpicture is made up of one or more full rectangular slices that collectively cover a rectangular region of the picture. A subpicture can be specified as extractable (i.e., independent coded of other subpictures of the same picture and other subpictures of prior pictures in decoding order) or non extractable. In addition, the encoder can select whether in-loop filtering (including deblocking, SAO and ALF) across subpicture boundaries is enabled for each subpicture separately.

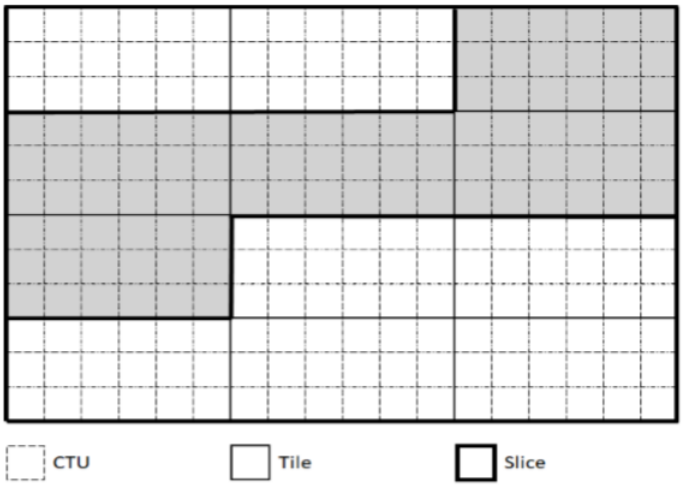


FIGURE 3.9: Picture with 18×12 luma CTUs that are partitioned into 12 tiles and three raster-scan slices.

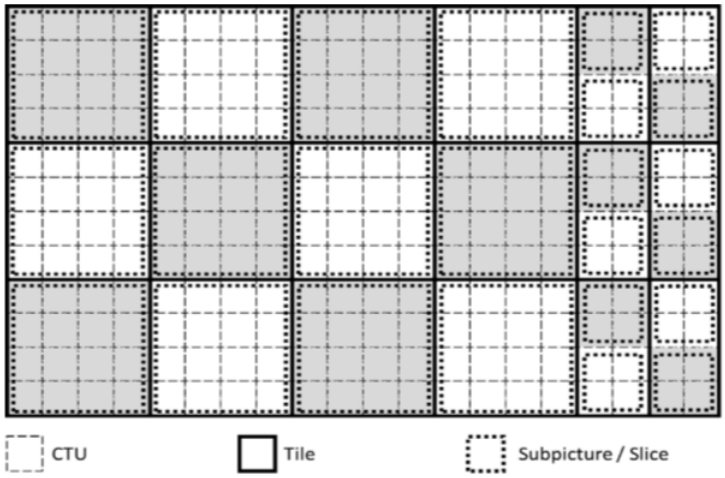


FIGURE 3.10: Picture partitioned into 18 tiles, 24 slices, and 24 subpictures.

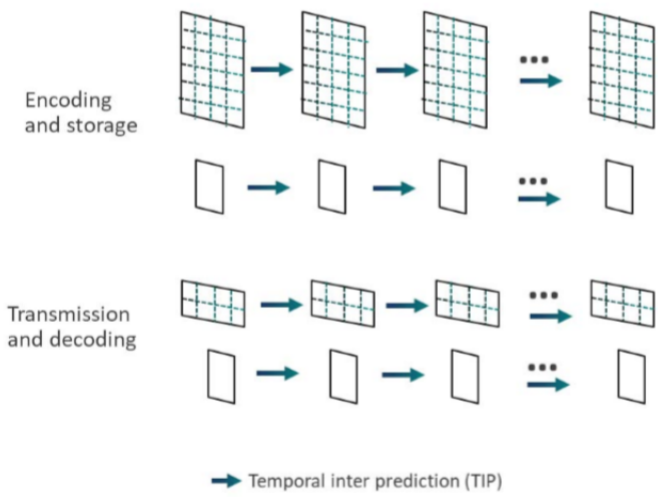


FIGURE 3.11: Sub-picture-based viewport-dependent 360° video delivery scheme.

Subpictures are functionally equivalent to the MCTSs that are supported with SEI messages in HEVC. They both allow for the independent coding and extraction of a rectangular subset of a sequence of coded pictures, which is useful for applications like viewport-dependent 360° video streaming optimization and Region-Of-Interes (ROI) applications.

At any given time, only a subset (i.e., the current viewport) of the entire omnidirectional video sphere is rendered to the user in 360° video streaming, also known as omnidirectional video. In these types of videos, the users can turn their head at any time to change their viewing orientation and, consequently, the current viewport. It is desirable to have a lower-quality representation of the area not covered by the current viewport available at the client and ready to be rendered to the user if they suddenly change their viewing orientation to somewhere else on the sphere. A high-quality representation of the omnidirectional video is only required for the viewport that is actively being rendered to the user. Such an optimization can be achieved by splitting the high-quality representation of the whole omnidirectional video into subpictures of suitable granularity.

Figure 3.12 shows an example subpicture-based viewport-dependent 360° video delivery scheme, in which a higher resolution representation of the full video scene uses subpictures. A lower resolution representation does not use subpictures and can be coded with fewer random access points than the higher resolution representation. In the lower resolution, the client receives the entire video, whereas in the higher resolution, it only receives and decodes the subpictures that cover the current viewport.

The sub-picture feature in VVC allows the MVs of a coding block to point outside of the sub-picture even though the sub-picture is extractable, depending on decoder padding at sub-picture boundaries in this case, similar to how decoder padding is used at picture boundaries. Compared to the tight encoder-side motion limitations used in MCTSs, this allows for better coding efficiency. When extracting one or more VVC sub-pictures from a series of pictures to generate a sub-bitstream that is a conforming bitstream, rewriting of the SHs (and PH NAL units, if present) is not required. Rewriting of SHs is required for sub-bitstream extraction using HEVC MCTSs. Rewriting of SPSs and PPSs is required in both extraction scenarios. Although their respective numbers per bitstream are small, each picture has at least one slice and the amount of data in the slices can be substantial. Thus, rewriting of SHs can be a significant burden for application systems. Furthermore, VVC specifies HRD and level definitions

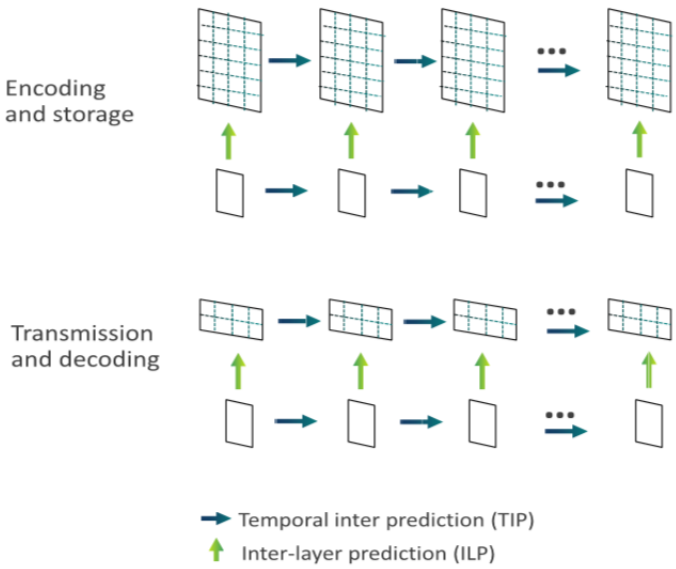


FIGURE 3.12: Sub-picture-based viewport-dependent 360° video delivery scheme making use of inter-layer prediction.

for sub-picture sequences; hence, system operations such as sub-picture-based bitstream extraction and merging, may rely on the compliance of the sub-bitstream of each extractable subpicture sequence.

The layout of subpictures in VVC is signaled in the SPS, and thus,it is constant within a CVS. The transmission of subpicture IDs is the technique that allows the extraction of subpicture sequences without rewriting SHs and PHs. The subpicture ID can differ from the value of the subpicture index, and the subpicture ID mapping (a list of subpicture IDs, one for each subpicture) is signaled, which may be constant within a CVS (in which case it is signaled in the SPS) or allowed to change in the pictures within a CVS (in which case it is signaled in the PPS). The subpicture-level slice index and the subpicture ID of the subpicture containing the slice are signaled in the SH. The sub-picture ID and sub-picture-level slice index work together to inform the decoder where to put the decoded tiles or in-tile CTU rows in the decoded picture slice. The same subpicture ID value would be signaled in the rewritten SPS or PPS in an extracted sub-bitstream including a subset of the sub-pictures in each picture of an original bit stream, even if the sub-picture now has a new sub-picture index number. The unchanged sub-picture ID and sub-picture-level slice index in the SH can still correctly determine the position of each CTU in the decoded picture of the extracted bit stream. This is also valid even when the raster-scan CTU address of the first CTU in a slice in the sub-picture has changed.

3.2.3.6 Picture Resolution Changes With Inter-Picture Prediction

The spatial resolution of pictures in AVC and HEVC cannot be changed until a new CVS is formed with a new SPS and an IRAP picture. Without encoding an IRAP picture, VVC enables picture resolution changes inside a CVS, allowing inter-picture prediction with references to pictures of different resolutions than the current picture being decoded. This feature is known as RPR because it necessitates resampling of the reference photos used for inter-picture prediction when their resolution differs from that of the current picture.

RPR's scaling ratio must be greater than or equal to 1/2 (factor-of-two downsampling from the reference to the current picture) and less than or equal to 8. (factor-of-8 up sampling). To manage varied scaling ratios between a reference picture and the current picture, three sets of resampling filters with varying frequency cutoffs are specified. The three sets of resampling filters are applied for the scaling ratios ranging from 1/2 to 1/1.75, 1/1.75 to 1/1.25, and 1/1.25 to 8, respectively. Each set of resampling filters presents 16 luma phases and 32 chroma phases, which is the same for the motion compensation interpolation filters. Actually, the traditional motion compensation interpolation method is a particular case of the resampling method, with a scaling ratio ranging from 1/1.25 to 8. The horizontal and vertical scaling ratios are infered depending on the width and height of the picture. For the reference and current picture, left, right, top, and bottom scaling offsets are specified.

3.2.3.7 Scalability Support

As VVC supports RPR, a bitstream containing multiple spatial scalability layers, such as two layers with SD and HD resolution, does not require any additional signal processing coding tools. In fact, RPR unsampling filter can be used for spatial scalability when unsampling process is required. Nevertheless, scalability support necessitates certain high-level syntax modifications (compared with not supporting scalability).

VVC v1 specifies scaling support, and when compared to prior video coding standards' scalability support approaches, such as extensions of AVC and HEVC, the architecture of VVC scalability has been made more friendly to single-layer decoder designs. When decoding multilayer bitstreams, the decoding capacity is provided as if the bitstream had only one layer. Decoding capabilities for example, such as DPB size, is defined independently of the number of layers in the bitstream to be decoded. In general, a decoder built for single-layer bitstreams can decode multilayer bitstreams without much modification. The HLS elements are considerably simplified compared to the designs of multilayer extensions of AVC and HEVC, at the expense of some flexibility. An IRAP AU, for example, is required to include a picture for each of the layers in the CVS.

In VVC v1, scalability support enables not just conventional spatial scalability, quality scalability, and multiview scalability, but also various scalability and subpicture combinations. We can see from Figure 3.12 that interlayer prediction enhances the subpicture-based viewport-dependent 360° video delivery system described in Figure 3.11.

Profile, Tier, and Level (PTL) Aspects: In VVC, two new features of PTL have been introduced: general constraints and the concept of sub-profiles. The PTL syntax structure of HEVC has a few generic constraint fields indicating for example, whether the bitstream may contain interlaced source content. Almost every significant tool or feature in VVC is associated to a generic constraint flag associated with it. The main reason for this is to allow third parties, such as an application system standards body or even a company, to easily indicate that certain tools are not used in the bitstream. In case these tools are not convenient for their use case, there is no need to go through the time-consuming process of specifying a new VVC Profile and the difficult consensus negotiations. For a similar reason, the subprofile idea was created. This allows a third party to establish a subprofile that contains a subset of the tools/features included in an existing VVC profile by simply registering an identifier (as described by Rec. ITU-T T.35 [50]).

Six profiles are defined by VVC version 1:

1. two single-layer video profiles, the Main 10 profile and the Main 10 4:4:4 profile, which enable all coding tools but limit the bitstream to one layer. However, temporal scalability support of sublayers is unrestricted.

2. The Multilayer Main 10 profile and the Multilayer Main 10 4:4:4 profile are two multilayer video profiles, with the main distinction being that the bitstream can include several layers, as opposed to the two single-layer video profiles.

3. The difference between the two single-layer video profiles is that the bit stream can only include one picture, which must be intra-picture coded. There are two still picture profiles, the Main 10 Still Picture profile and the Main 10 4:4:4 Still Picture profile.

3.2.4 Performance

3.2.4.1 Objective evaluation

The JVET has defined certain Common Test Conditions (CTCs) [51] for conducting experiments in a well-defined manner so that the results may be compared fairly. During the development of VVC, the CTCs were utilized to assess the proposals. The CTC definition contains three mandatory test conditions that reflect all-intra, random access, and low-delay settings, with the random access scenario being more essential than the others due to its considerably larger application use. In the experiments, a total of 18 video sequences were used, comprising Classes A1 and A2 (3840×2160), Class B (1920×1080), Class C (832×480), Class D (416×240), Class E(1280×720), and Class F (different resolutions of screen contents). Camera-captured video sequences are represented by Classes A to D, while video conferencing sequences are represented by Class E, and screen content sequences are represented by Class F. The random access case is not evaluated for Class E. This latter case has an unacceptable delay exceeding the authorized one for video conferencing. In the low-delay case, Class A sequences are not mandatory to be examined, since such source material would seldom be used in low-delay applications. All of these test materials are scanned in a progressive manner, using 4:2:0 color sampling and 8 or 10 bits per sample. The structural delay is set to 16 frames in the random access scenario, and the IRAP random access period is set to about 1 second. Four rate points are tested with constant QP settings, with the base QP set to 22, 27, 32, and 37. The QP of higher temporal sub-layers is calculated using fixed offsets from these values. The compression performance is evaluated using the JVET CTC conditions and the Bjøntegaard delta bit rate (BD-rate) measurement technique [52], [53] based on the weighted average of Peak Signal-to-Noise Ratio (PSNR) values per color component:

$$\mathrm{PSNR}_{YUV} = \left(\frac{1}{8}\right)(6 * \mathrm{PSNR}_{Y} + \mathrm{PSNR}_{CB} + \mathrm{PSNR}_{CR}) \tag{3.1}$$

PSNR_{Y} is given a higher weighting to compensate for the fact that most of the bits are utilized to encode the luma component of video pictures. Not to mention that luma component is the most perceptually important component.

In order to compare the performance of VVC to its predecessors, [1] conducted some tests. The VVC Test Model (VTM-9.0) [54], the HEVC Test Model (HM-16.20) [55], the HEVC SCC Extension Test Model (SCM-8.8) [56] for Class F, and the AVC Joint Test Model (JM-19.0) [57] were used as reference software encoders. Due to capacity restrictions on the DPB for AVC, a random access configuration with an eight-frame structural delay is employed, which may be found in the JM software package's "cfg/HM-like"

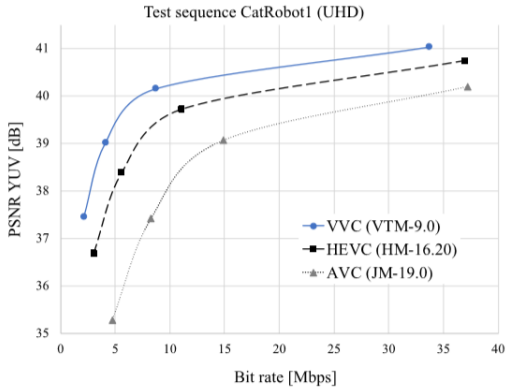


FIGURE 3.13: Rate-distortion plots of VVC, HEVC, and AVC for the CatRobot1 video test sequence (random access configuration) [1].

Class	AVC High 10 Random Access	HEVC High 10 Random Access	HEVC High 10 Low Delay B	HEVC High 10 All Intra
A1 (3840x2160)	71,6%	39,2%	-	30,4%
A2 (3840x2160)	69,8%	42,9%	-	28,6%
B (1920x1080)	65,0%	37,2%	31,8%	23,0%
C (832x480)	55,6%	30,3%	28,1%	22,3%
E (1280x720)	-	-	34,0%	25,7%
Average	**64,8%**	**36,9%**	**31,1%**	**25,5%**
D (416x240)	52,0%	27,5%	24,4%	17,5%
F (Screen Content)	64,9%	42,4%	43,1%	39,8%

TABLE 3.2: YUV BD-Rate Savings of VVC (VTM-9.0) Over AVC and HEVC.

configuration files. Furthermore, [58] provides a more thorough comparison of HEVC's coding efficiency to that of its predecessors. Table 3.2 shows the average BD-rate savings of VVC over AVC and HEVC for each type of sequences. The total average BD-rate savings are based on test sequences in class A/B/C/E, which are believed to be representative of VVC's target user situations. The BD-rate savings of VVC over AVC for random access may be found to be 65 percent on average, with up to 72 percent for 4k resolutions. When compared to HEVC in various configurations, VVC achieves the largest coding gain in the random access situation, saving an average of 36.9% YUV BD-rate, and saving more over 40% for test sequences with 4k resolutions. When comparing the VTM to the HEVC reference software (HM) in the HEVC Main 10 profile configuration, VVC provides even larger BD-rate reductions for Class F, which represents SCC. While Main 10 is the most widely used HEVC profile, the HEVC Screen-Extended Main 10 profile, introduced in version 4 of HEVC, can achieve higher coding efficiency for screen content. In the HEVC Screen-Extended Main 10 profile configuration (the final row of the table), VVC still saves 26.4 percent YUV BD-rate for random access, 32.5 percent YUV BD-rate for low delay, and 17.8 percent YUV BD-rate for all-intra. Figure 3.13 depicts a random access rate-distortion plot of VVC, HEVC, and AVC for the CatRobot1 UHD video test sequence.

3.2.4.2 Subjective evaluation

The HEVC and VVC projects' compression capability aim has been to lower the bit rate for a certain level of subjective video quality, or the quality perceived by human observers. Despite being a useful objective measuring approach, PSNR is insufficient to replace subjective quality evaluation. This prompted the JVET to begin formal testing operations using rigorous subjective assessment methodologies in order to validate the final standard's coding efficiency. The first verification test, completed in October 2020, included UHD SDR content in a random access configuration, as utilized in modern streaming and broadcast television applications [59]. Outside of the JVET test set, five challenging UHD SDR sequences were chosen and encoded at five different quality levels [1], ranging from irritating to practically undetectable deficiencies. Although the major focus was on comparing the VVC reference software VTM to the HEVC reference software, the tests also included an open-source encoder implementation (VVenC) [60]. VVenC version 0.1 in the "medium" preset runs significantly faster(110x) than VTM and offers subjective quality enhancement methods such as input video temporal filtering and perceptually adjusted bit allocation [61]. For all five test sequences, Table 3.3 presents the subjective Mean Opinion Score (MOS) and objective PSNR-YUV-based BD-rate savings. This test confirms that the VTM and VVenC encoders for VVC significantly enhance compression, with the VTM decreasing the bit rate by 43% on average for the same perceived quality compared to the HM, and VVenC reducing the bit rate by another 12% compared to the VTM. The PSNR-YUV BD-rate savings for VVenC against the VTM, on the other hand, are significantly smaller and even negative (i.e., a bit rate increase). The subjective quality benefit evaluated compared to the HM for both tested VVC encoders exceeds the benefit indicated by PSNR-YUV BD-rate numbers—a phenomenon that was previously observed for HEVC relative to its AVC predecessor [58]. The arithmetic average of the MOS values over the geometric average of the associated rate points is plotted in Figure 3.14 to illustrate pooled findings for all five test sequences. It can be observed that the VTM and HM have similar quality levels.

UHD SDR Sequences	VTM vs HM		VVenC vs VTM	
	MOS	PSNR YUV	MOS	PSNR YUV
DrivingPOV3	61%	43%	11%	-12%
Marathon2	37%	33%	7%	-16%
MountainBay2	37%	39%	3%	0%
NeptuneFountain3	41%	37%	16%	-5%
Average	**43%**	**36%**	**12%**	**-9%**

TABLE 3.3: MOS and PSNR-YUV BD-Rate Savings of VVC (VTM-10.0) Over HEVC (HM-16.22) and of an Optimized VVC Encoder (VVenC-0.1) Over VTM.

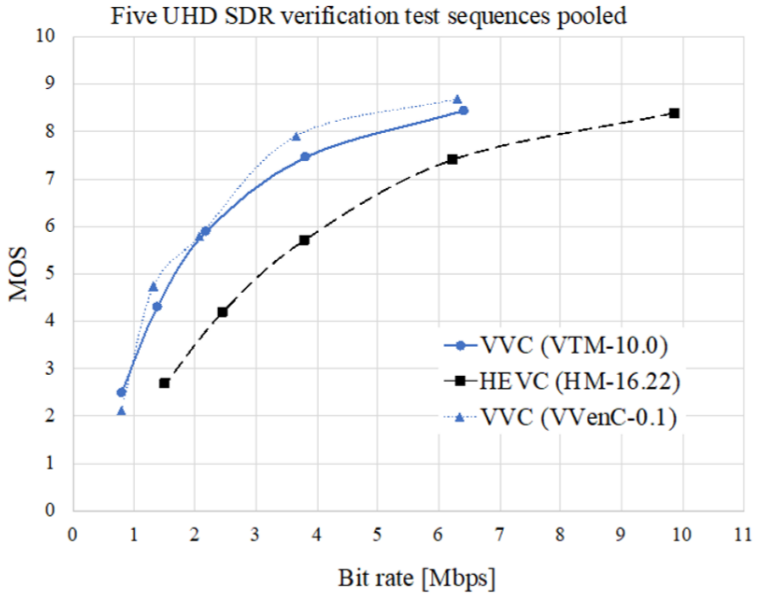


FIGURE 3.14: Average (arithmetic) MOS and (geometric mean) bit rates of VVC (VTM and VVenC encoders) and HEVC (HM encoder) pooled over the five UHD SDR sequences used in the verification test [1].

3.3 Basics of Machine Learning (ML)

Machine Learning [62, 63, 64, 65] is a "field of research that provides computers the ability to learn without being explicitly programmed", as Arthur Samuel defined it in 1959 [66]. Although the phrase machine learning has its roots in computer science, numerous vector quantization algorithms [67] for coding and compression have been established in telecommunications and signal processing [68].

Learning in computer and data science is based on examples (data samples) and experience. In most cases, the feature extraction stage retrieves compact data-bearing parameters that may be used to describe the data. A machine learning method is used to train the classification stage to recognize and categorize the collection of features. The area of machine learning is broad, and applications are quickly growing, especially with the advent of fast mobile devices with cloud computing capabilities [69]. The topic of compressing and retrieving big data has lately sparked new attention. Smart city projects, mobile health monitoring, networked security, manufacturing, self-driving cars, surveillance, and intelligent border control are all examples of smart city initiatives. Every application necessitates its unique features, adaptive learning, and data fusion. Data compression and statistical signal and data analysis play a significant role in transferring and understanding data, as well as creating useful analytics. supervised, unsupervised, and semi-supervised algorithms are three types of machine learning algorithms that may be categorized based on their properties, learning style, and data usage [70]. This categorization is useful for determining the function of the input data, as well as the value of algorithms and learning models in relation to the applications.

3.3.1 Supervised Learning

In supervised learning, "true" or "correct" labels of the input dataset are available. Because the algorithm is "trained" with the labeled input dataset (training data), ground truth samples are accessible for training. The algorithm produces suitable predictions on the input data during the training process, then improves its estimations using the ground truth and repeats the process until the algorithm gets the desired degree of accuracy. A a cost function or an objective function is optimized in virtually all machine learning models. The cost function is generally a measure of the difference between the algorithm estimates and the ground truth. We train our model to provide predictions that are near to the true values by minimizing the cost function (ground truth). Gradient descent is commonly used to minimize the cost function [71, 72, 73, 74]. Many machine learning training paradigms have utilized variations of gradient descent techniques such as stochastic gradient descent for a mini-batch, momentum based gradient descent [75, 76], and Nesterov accelerated gradient descent [77]. Assume we have m number of training examples, each of which is a labelled data set that may be represented in the pair (x, y), where x and y represent the input data and the class label, respectively. The input data x can be an n dimensional. Each dimension refers to a feature or a variable. Supervised learning approaches are utilized in a variety of domains, including phytoplankton species identification [78], mapping rainfall-induced landslides [79], and biomedical data categorization [80]. There are different supervised learning methods:

- **Linear Regression** is a statistical approach [81, 82, 83] used to estimate the relationship between input and output variables. The input variables are mapped into a continuous function. A simple univariate linear regression [84, 85, 86, 87] model is shown in Figure 3.15. The linear regression model assumes a linear connection between the independent and dependent variables and fits the data points with a straight line. A hypothesis function or a prediction function expresses this

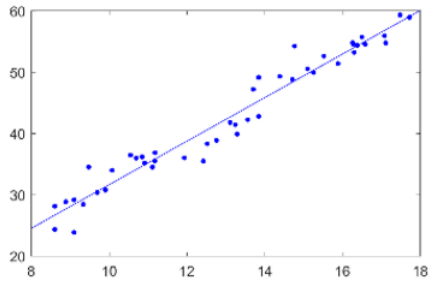


FIGURE 3.15: A simple Linear regression example with one feature/variable.

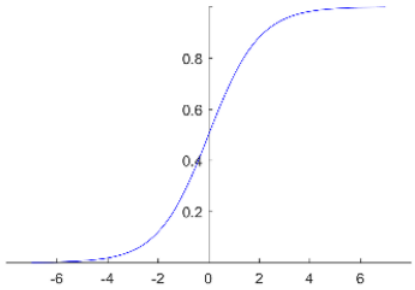


FIGURE 3.16: Sigmoid curve having a bound between 0 and 1.

relationship. It is written as:

$$h(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n \tag{3.2}$$

where x_1,$x_2, \ldots, x_n$ are the features and w_1,$w_2, \ldots, w_n$ are the weights of the model.

We can also have quadratic, cubic or higher polynomial [88, 89] terms to obtain completely different hypothesis function. The output is the linear sum of the weighted input features. The weights are typically learned by weighted least squares minimization process.

- **Logistic Regression:** The goal of a multivariate regression model is to find a hypothesis function that produces a continuous result. Now, we present another type of supervised learning algorithm: Classification. Obtaining a discrete output, is another class of supervised learning algorithm, called Classification. As before, the input can have one or more features (or variables) and the output can be either 0 or 1, performing binary classification of positive class from negative class. Logistic regression [90, 91] uses a sigmoid curve shown in Figure 3.16 to output a probability value and thus performs the classification. For a logistic regression, the hypothesis function is given by:

$$h(x) = S(w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n) \tag{3.3}$$

where S is a sigmoid function given by

$$S = \frac{1}{1 + \exp(-z)} \tag{3.4}$$

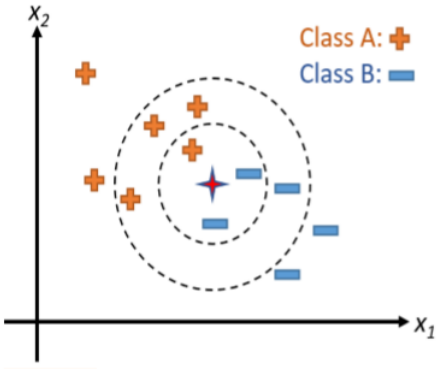


FIGURE 3.17: A simple k -NN model for different values of k.

- **Support Vector Machines (SVM):** Support Vector Machines are one of the popular supervised learning models [92, 93, 94]. They are mainly used for binary classification as well as multi-class classification. SVM tries to divide these points to two separate groups by a hyperplane, such that the width of separation between the two groups is maximized. A major advantage of SVM is that it avoids overfitting and is non-probabilistic.

- **Naïve Bayes:** Naïve Bayes [95] classifiers are simple probabilistic classifiers. They are based on the following assumption: all the input features are independent of each other and no correlation exists between them. The algorithm is used for text classification [96], credit scoring [97], emotion classification and recognition [98], and detection of epileptic seizures from EEG signals [99].

- **k-Nearest Neighbors:** One of the simplest supervised machine learning methods is the k-Nearest Neighbors (k-NN) algorithm [100]. It may be used to classify discrete outcomes from discrete input points. In Figure 3.17, where a basic k -NN is presented, the test point (star) is identified as belonging to class B for k = 3 and class A for $k = 6$. Since k -NN is a non-probabilistic and non-parametric model [101, 102, 103], it is the first option for classification studies when data distribution is unknown. To classify any unknown sample, k -NN saves all of the labelled input points, and this classification is based on the similarity measure (a distance metric). k -NN classifier is used for stress detection using physiological signals in [104] and detection of epileptic seizures [99].

3.3.2 Unsupervised Learning

There are no explicit labels connected with the training dataset in unsupervised algorithms [105, 106]. The goal is to learn more about the data by drawing inferences from the input data and then modeling the hidden or underlying structure and distribution in the data. The most frequent example of an unsupervised algorithm is clustering.

- **Clustering** [107, 108] is the process of identifying a structure or pattern in a group of unlabeled data. Clustering algorithms divide data into K number of clusters for a given dataset, with data points inside each cluster being comparable to each other. The similarity measure or distance metric is utilized in the same way as the k -NN method. Euclidean, Mahalanobis, cosine, Minkowski, and other distance metrics are employed. The K-means method is one of the most basic clustering algorithms available, and it is both straightforward and iterative. It clusters the data by dividing it into

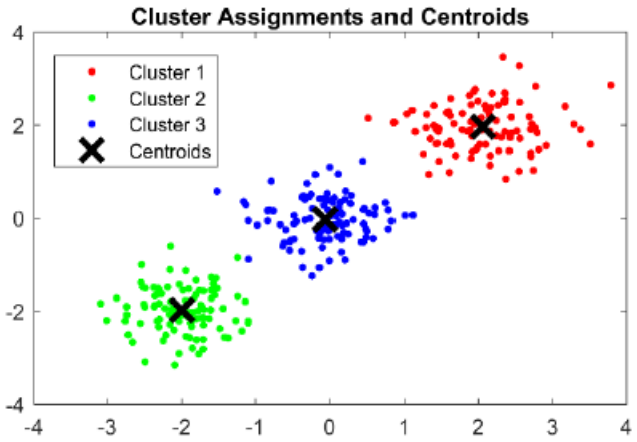


FIGURE 3.18: The K-means and the cluster centroids.

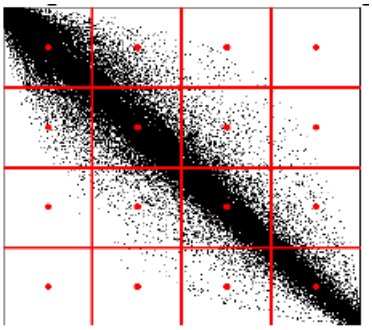


FIGURE 3.19: Uniform quantization of 2-dimensional Data.

K equal-variance groups by minimizing the inertia or within-cluster sum-of-squares. A converged K-means method is presented in Figure 3.18. Clustering has a wide range of applications in a variety of disciplines. Many examples can be cited, like in biology to identify groups of genes having similar functions [109, 110, 111], for brain tumor detection in [112], cardiogram data clustering [113], in business and e-commerce analysis [114] and information retrieval [115], image segmentation [116] and compression [117], in the study of quantitative resolutions of nanoparticles [118], and in fault detection in Solar PV panels [119, 120].

- **Vector quantization** Vector quantization [120, 67] arranges data into vectors and represents them by their centroids in its most basic form. The quantizer is usually trained using a K-means clustering method. The centroids are used to create codewords, which are then stored in a Codebook. The lossy compression method vector quantization is utilized in a variety of coding applications. As a result, mistakes are inversely related to density of compressed data. Figure 3.20 depicts this feature, which is contrasted with Figure 3.19 depicting uniform quantization. Speech coding [121], emotion identification [122], audio compression [68], large-scale picture classification [123], and image compression [124] are all applications that employ the Vector quantization approach.

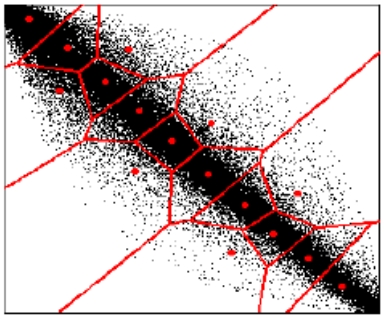


FIGURE 3.20: Vector quantization of 2-dimensional Data.

3.3.3 Deep Learning

Although, the inception of neural networks dates in 1960 [125], deep learning gained more popularity since 2012 [126] thanks to the great advancements in the GPUs [127] and availability of large labeled datasets. The past decade has witnessed the emerging and success of deep learning, which is a class of techniques that are progressively adopted in the hope of approaching the ultimate goal of artificial intelligence [128].

Deep learning belongs to machine learning technology, and has the distinction of its computational models, known as deep artificial neural networks, which are composed of multiple (usually more than three) processing layers, each layer is further composed of multiple simple but non-linear basic computational units. In Figure 3.21, a simple artificial neural network with 4 hidden layers is shown. The last layer, namely the output layer, performs classification. Each successive layer takes the output of the previous layer and feeds the result to the next layer. Convolutional Neural Networks (CNNs) are feed-forward, multi-layer neural networks, structured into a feature extraction stage followed by an inference stage [129]. The feature extraction stage includes a number of convolutional layers, each encompassing a number of learnable filters. Each filter activates upon the detection of one specific feature in the input. The output of the feature extraction stage is processed by one or more fully-connected layers, the actual number of layers and learnable parameters in each layer depending on the specific application. Finally, the last layer of the network provides the desired network output such as the class an object belongs to (classification problems) which corresponds to the cases considered in this manuscript, or the object position in the image (regression problems). One benefit of such deep networks is believed to be the capacity for processing data with multiple levels of abstraction and converting data into different kinds of representations. Note that these representations are not manually designed; instead, the deep network including the processing layers is learned from massive data using a general machine learning procedure.

There are several network architectures including the one shown in Figure 3.21, which consists of dot product layers (fully connected layers). A convolutional layer [130] creates feature maps by processing a volume of activations rather than a vector. To minimize the size of the feature maps, it additionally employs a subsampling or maxpooling layer. Figure 3.22 depicts a Convolutional Neural Network (CNN) in action. Deep learning eliminates the necessity of handcrafted representations. Also, CNNs have emerged as the cornerstone to efficiently solve a number of computer vision problems.

Typical artificial neural networks challenges include initialization of the network parameters, overfitting, and long training time. We now have various techniques to address the above problems. Batch normalization [131], normalization propagation [132], weight normalization [133], layer normalization [134] all help in accelerating the training of deep neural networks. Dropouts [135] help in reducing overfitting.

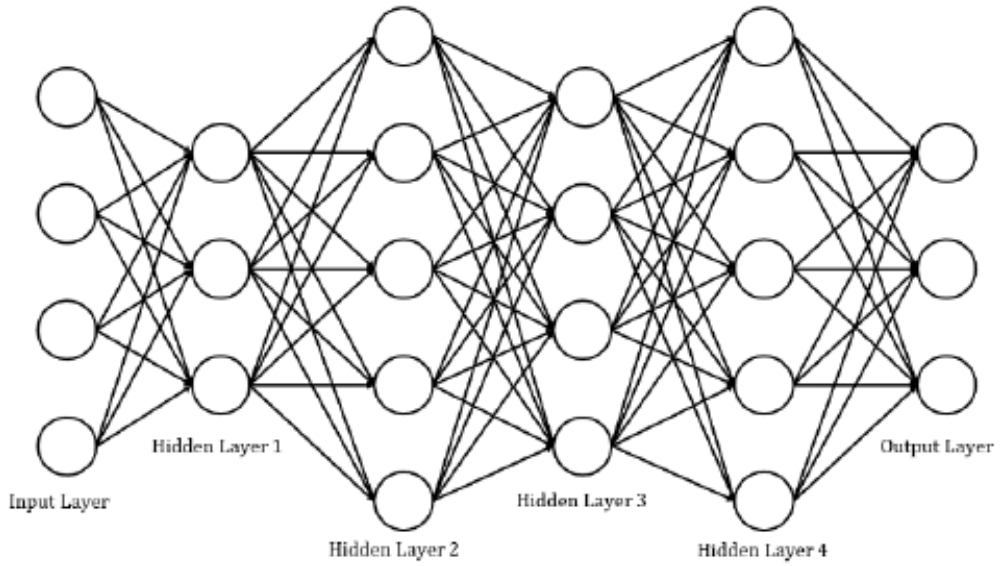


FIGURE 3.21: Artificial Neural Network with four hidden layers.

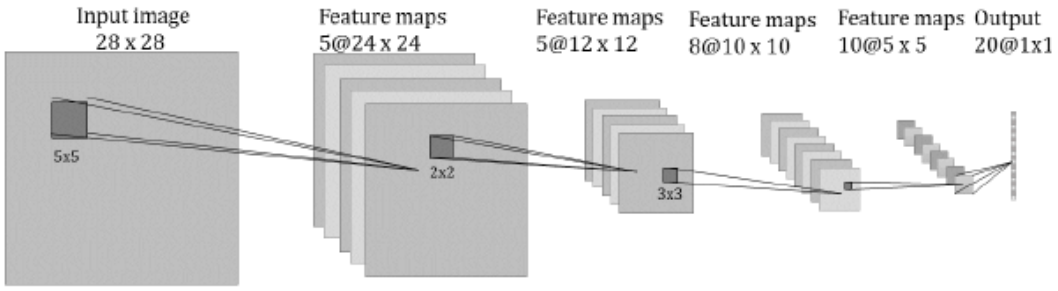


FIGURE 3.22: A CNN with 3 convolutional, 2 subsampling layers.

Witnessing such successful cases, experts cannot help but ask whether deep learning can benefit video coding as well. In history, artificial neural network is not strange to the image/video coding community. From 1980s to 1990s, a number of research was conducted on neural network-based image coding [136, 137]. The exploration of using deep learning for image/video coding is worthy of reconsideration, and indeed has been an actively developing research area since 2015. At present, research has shown promising results, confirmed the feasibility of deep learning based image/video coding. Nonetheless, this technology is far from being mature and calls for much more research and development efforts.

Readers may know that integrating Deep learning into video coding is divided into two categories. The first category is deep schemes, i.e. new coding schemes that are built primarily upon deep networks [138, 139]; the second category is deep tools, i.e. deep network-based coding tools that are embedded into traditional, non-deep coding schemes [140, 141]; a deep tool may either replace its counterpart in the traditional scheme, or be newly added into the scheme.

Chapter 4

State of the Art

This thesis mainly consists of three contributions. In this chapter, we provide an overview of the related works for each of them.

4.1 Decoder-side intra mode derivation techniques

The general concept of DIMD is not currently in any standard codec, even though in the literature, some studies have been presented to implement the idea [142, 143]. However, most of these works rely on Template Matching (TM) algorithms, that are too complex to be implemented in real-time low-power decoders. In fact, using the TM-based method, several intra modes have to be actually applied at the decoder side to compute their distortion from a template [142]. One of the most recent studies about derivation of the intra mode information at the decoder side uses a template area around current block as well as an additional line of reference for predicting that template area [143]. More precisely, this method simulates the prediction in a neighborhood of the current block, where reconstructed pixels at the decoder side can be used for distortion measurement of a prediction hypothesis. This algorithm sparsely applies the IPMs of Joint Exploration Model (JEM) and achieves about -0.9% gain with significant complexity overhead at both encoder and decoder sides. In this report, an algorithm based on decoder-side intra mode derivation is proposed. As opposed to the aforementioned schemes that rely on complex decoder-side operations, the method relies on a simple texture analysis method. The algorithm was inspired by existing encoder-side optimisations based on gradient analysis [144], some of which have been used in commercial encoder solutions. This manuscript focuses on exploiting similar texture analysis techniques normatively at the decoder, with the goal of improving the coding efficiency without severely affecting the encoder and decoder complexity. As shown in a previous work [145], this allows the proposed technology to be potentially used in practical applications.

Other similar techniques, that are based on the derivation of information on the decoder side, existed in the Inter coding mode of the JEM software. A first tool included in the JEM corresponds to a special mode of coding based on Frame-Rate Up Conversion (FRUC) techniques. With this mode, motion information of a block is not reported but derived at decoder side. This derivation is based on matches found between the blocks that adjacent the current block in the current frame and in the reference frame. This process is commonly called: Template Matching. A second tool that can be cited is the Bi-Directional Optical flow (BDOF) which is pixel-wise motion refinement and is performed on top of block-wise motion compensation in a case of bi-prediction. Sample-level motion refinement does not require exhaustive search or signaling since there is an explicit equation which gives fine motion vector for each sample.

The Decoder-side Motion Vector Refinement (DMVR) is a method used to refine the motion vectors using a Template Matching, As in the case of FRUC. The DMVR is only applied to bi-predicted blocks.

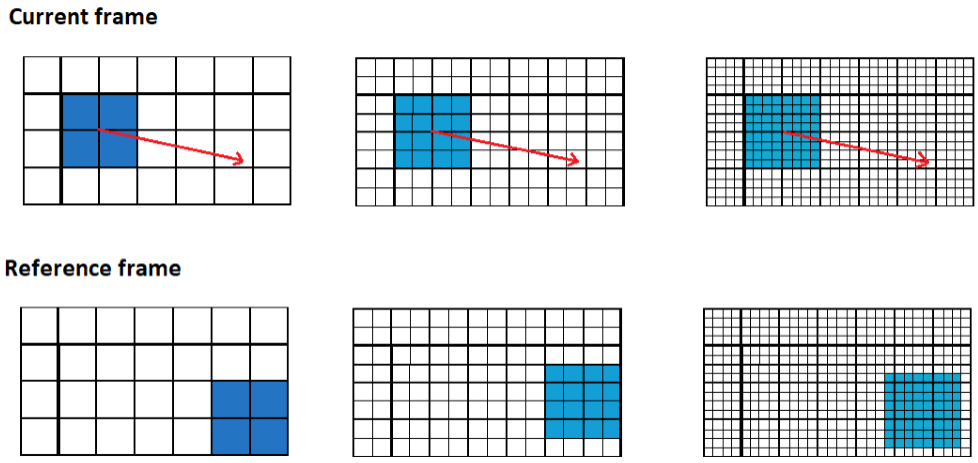


FIGURE 4.1: Introducing 4-pel precision to improve accuracy of MCP.

These methods are found in inter mode allowing the reduction of signaling of prediction information, but the absence of such techniques for intra mode motivates us to adapt the principle of deriving the Intra prediction modes in order to reduce the bitrate dedicated to these modes.

4.2 Interpolation Filtering

As presented in Section 3.2.2.2, MCP is an essential technique used by video coders to reduce the amount of information transmitted to a decoder by exploiting the temporal redundancy present in the video signal. The relative position of the prediction with respect to the original block is called a Motion Vector (MV) and is transmitted to the decoder along with the residual signal. The true displacements of moving objects between pictures are continuous and do not follow the sampling grid of the digitized video sequence. This is graphically described in Figure 4.1. The precision of MV can be represented in different accuracy. The 4-pel precision has been introduced in H.264/AVC and is also used in HEVC. The 4-pel precision refers to using a quarter of the distance between pixels (or luma sample positions) as the MV precision for motion estimation and MCP. VVC went further in the precision. In some inter prediction modes such as the affine mode, motion vectors are derived at 1/16 th-luma-sample precision and motion compensated prediction is performed at 1/16 th-sample-precision. Hence, by utilizing fractional accuracy for motion vectors instead of integer accuracy, the residual error is decreased and coding efficiency of video coders is increased. The samples of the Prediction Block (PB) for an inter-coded Coding Block (CB) are obtained from those of a corresponding block region in the reference picture, which is at a position displaced by the horizontal and vertical components of the MV. Except for the case when the MV has an integer value, fractional sample interpolation is used to generate the prediction for non-integer sampling positions. Interpolation filters are used to help predict the accurate values of pixels. Though higher precision motion vectors take more bits to encode, they can sometimes result in more efficient compression overall by increasing the quality of the prediction signal.

The interpolation filter in H.264/AVC uses various combinations of separable one-dimensional filters according to the fractional sample position [146]. The main features of the interpolation filter used in AVC are:

- 6-taps interpolation filter for obtaining luma samples at half-pel positions.
- Cascade averaging for obtaining luma samples at quarter-pel positions
- Bilinear interpolation for chroma samples

In HEVC, there is only one interpolation filter for Luma and one interpolation filter for Chroma, for each fractional position. The design used is based on the simplified form of the DCT-IF [147]. The main features of DCT-IF interpolation filter used in HEVC standard are:

- symmetric 8-taps filter for luma samples at half-pel positions
- asymmetric 7-taps filter for luma samples at quarter-pel positions
- 4-taps filter for chroma samples

In all previous video coding standards, the interpolation process for inter predicted frames has been static, therefore, all the coefficients are the same and are independent from the content of the frames. Filter coefficients for luma samples in HEVC, where just 4-pel precision existed, are shown in Table 6.2, in frac =0 (integer), frac = 4 and 12 (Quarter-pel) and frac = 8 (Half-pel). The interpolation filter in the HEVC standard is a DCT-based interpolation filter (DCT-IF) [148]. The H.266/VVC standard, inherits the DCT-IF from HEVC with an extension from the quarter-pel precision to the high precision (1/16 pel precision) in VVC. Table 6.2 represents the new coefficients of VVC. Although DCT-IF is efficient, it is not adapted to objects whose speed vary, since it cannot compensate variations in the amount of motion-related blurriness incurred by the camera capture. Furthermore, DCT-IF tries to maximize accuracy, but this may not be a desirable characteristic when interpolating some particular cases.

4.2.1 Non-CNN based switching interpolation filters

Switching interpolation filters concept allows for using different interpolation filters for the same fractional sample position in different parts of a video sequence. Various methods related to the switched interpolation filtering have been proposed in the literature; namely, non-separable/separable Adaptive Interpolation Filter (AIF/SAIF) [149], Switched Interpolation Filter with Offset (SIFO) [150], Enhanced Adaptive Interpolation Filter (EAIF) [151]. However, a common drawback of these methods is that the switching operation among different interpolation filters is performed only at a coarse granularity, i.e. a slice level. Furthermore, the selection of the interpolation filter may be explicitly signalled, possibly by transmitting the individual Finite Impulse Response (FIR) filter coefficients, as for AIF and its variants. Allowing to select an interpolation filter at a finer granularity, such as Coding Tree Unit (CTU) or Coding Unit (CU) level, should enable better adaptation to local image characteristics.

Recently, a new related work proposed the idea of enabling the interpolation filter selection at a finer granularity, the block level [5]. However, this work only proposed an alternative for the half pel precision, without taking the advantage of switching also the quarter pel precision which can potentially bring additional benefits. Note that this work was only applied in the Adaptive Motion Vector Resolution (AMVR), where coding motion vector difference is allowed at different precisions. Moreover, the potential of this work is limited by the signalization fact which can firstly cause the loss of a big proportion of the gains and secondly, limits the number of proposed filters since increasing this number would require the increasing of signaling cost. This can obviously reveal the interest of the proposed method. The novelty of avoiding the signaling presented in this work, gives more freedom in proposing more filters as opportunities to interpolate.

4.2.2 Related Deep Learning works

In the context of Inter-prediction works, and inspired by the fractional-pixel Motion Estimation (ME) and Motion Compensation (MC), a number of researches is conducted on the fractional-pixel interpolation problem. Yan et al. propose to use a CNN for half-pixel interpolation [152]. This method is inherited in [153], where the authors analyze the effect of different blurring degrees. Zhang et al. propose another method, which formulates the fractional interpolation as a resolution enhancement problem [154]. [140] considers a different formulation, treating the fractional-pixel MC as an inter-picture regression problem. Yan et al. further discover a key characteristic of the fractional interpolation problem [155], namely its invertibility: if fractional pixels can be interpolated from integer pixels, then integer pixels should also be interpolated from fractional pixels. Based on the invertibility, they propose an unsupervised manner to train CNN for half-pixel interpolation. Wang et al. also refine the inter prediction signal by a CNN [156], where the CNN inputs include the inter prediction signal, the context of the current block, and the context of the inter prediction block.

4.3 Encoder optimization

4.3.1 Related works

In the literature, the problem of encoder optimization has been extensively addressed. Many proposals have focused on streamlining the HEVC encoder's Rate Distortion Optimization (RDO) process. An algorithm to quickly determine the Coding Unit (CU) size and depth has been proposed in [157] for both Intra and Inter modes, focusing on the CU size and depth decision. [158] employs the coding information from neighboring blocks to reduce the number of Intra mode candidates. On the other hand, [159] proposes a decision algorithm that uses local samples to compute the dominant edge of the Prediction Unit (PU). [160] proposes a method for determining the PU partition by examining all CUs across surrounding $N \times N$ partitions, focusing on the Motion Vectors (MVs) search in Inter prediction. All of the preceding methods change the algorithm of the non-normative encoding process and this can sometimes draw back the compression efficiency of some algorithms with the aim focusing on reducing the complexity.

Concerning the works related to VVC, several works have been proposed with the intention of reducing the remarkable complexity imposed by its encoder compared to the previous standard. Some of these works, similarly to the ones cited previously, succeeded to reduce the complexity by making algorithmic changes [161]. Most of these works targeted the block partitioning structure of intra prediction employing statistical analysis and/or machine learning approaches. For instance in [162], two new methods for fast VVC intra-picture encoding are proposed. Both are based on an approach that uses a CNN for block adaptive parameter estimation. This work evaluates hypothetical and actual encoding time reductions for VVC.

Other related works focusing on complexity reduction solutions for VVC through algorithmic optimizations can be cited. Some works have presented ways to simplify the evaluation of intra prediction modes, MTS steps, and inter prediction block partitioning. In this regard, Fu et al. propose a rapid CU partitioning technique based on a Bayesian decision rule classifier [163]. As a model input feature, the information generated from the present CU and horizontal binary splitting is employed. The All-Intra configuration was used to test this approach, which resulted in a reduction in encoding time and an improvement in BD rate. Yang et al. present a two-pronged approach to reducing complexity [164]. To minimize superfluous block partition assessments, the first one use decision trees. To decide whether to split the current block, a set of decision trees was trained for each partitioning type. The second uses a

gradient descent search to speed up the evaluation of particular angular intra prediction modes. When using an All-Intra configuration, the suggested method can cut 62.46 percent of the encoding time while increasing the BD-rate by 1.93 percent.

Chen et al. used the Support Vector Machine (SVM) to decide between horizontal and vertical partitioning in a complexity reduction approach [165]. Six SVM classifiers were evaluated for this task, based on the block size. Each classifier is taught online during the encoding of the first frame, and the following frames are encoded with the trained classifiers' decisions in mind. An All-Intra configuration was used to test this solution. With a 1.55 percent improvement in BD-rate, their approach decreases encoding complexity by 50.97 percent. Lei et al. propose a quick method for avoiding unnecessary block partition evaluations in advance by evaluating a subset of directional intra-frame prediction modes for virtual sub-partitions of the current block to estimate the horizontal and vertical partitioning cost of the current block [166]. This method may select whether to avoid horizontal or vertical divisions based on the expenses. The proposed approach is tested with the All-Intra configuration. The trial findings reveal that this method saves 40.7 percent of encoding time while increasing the BD-rate by 0.84%.

In the intra-frame prediction, Fu et al. presented an early decision strategy for the MTS assessment [167]. The suggested technique checks if all nearby spatial blocks are encoded with DCT-II using information from neighboring spatial blocks. If this is the case, the DST-VII and DCTVIII evaluations are skipped; otherwise, the frequency of each transform type employed in the neighborhood is calculated, and the transform list is sorted from most often to least frequent. If an intra prediction mode with the current transform has a greater cost than the previously examined transform, it is eliminated from the evaluations of the following transform types. Using the All-Intra configuration, this approach decreases encoding complexity by 23% and boosts the BD-rate by 0.16 %.

Cui et al. presented a complexity reduction approach for determining the optimal block partitioning structure based on the direction of sample gradients [141]. In this approach, the system choose between three partitioning options: split or not split, horizontal or vertical, and BT or TT. The gradients of the current block's sub-partitions are computed in four directions: horizontal, vertical, 45, and 135 degrees, and compared to preset threshold values. With an All Intra configuration, the suggested approach decreases 51.01% of the encoding complexity while increasing the BD-rate by 1.23%.

For selecting the direction of BT/TT partitions, Saldanha et al. suggest a rapid block partitioning determination technique [168]. The suggested technique takes into account the current block's intra-frame prediction mode as well as the variance of subpartitions. To determine whether the texture orientation is horizontal or vertical, the variance of subpartitions is computed. The method chooses by omitting vertical partitions if the variance identifies the block as horizontal texture and the specified angular intra prediction mode is horizontal. This choice is made for bypassing horizontal divisions as well. Furthermore, vertical partitions are skipped if the best mode in the current block is horizontal ISP; alternatively, if the best mode is vertical ISP, the horizontal partitions are skipped. The All-Intra configuration was used to test this method, which resulted in a 31.41% reduction in encoding time and a 0.98% improvement in BD-rate.

Amestoy et al. propose a Random Forest (RF) classifier-based approach [169]. A collection of RF classifiers was created for this purpose, with three primary options to consider: (i) split or not, (ii) split using QT or BT/TT, and (iii) split with horizontal or vertical partitioning. Beyond encoding context information such as QP value and motion vectors, these classifiers use information from current block samples such as variance and horizontal and vertical gradients. The suggested method was tested with a Random-Access configuration, yielding a decrease of 30.1% in encoding complexity and a 0.61% improvement in BD-rate.

To identify the optimum block partitioning and decrease encoding complexity, [170], [171], and [172] suggested complexity reduction techniques based on CNN. The All-Intra configuration was used to test the solution provided in [170]. With a 0.75% increase in BD-rate, the testing findings indicated a 42.2% reduction in encoding time. Using the All-Intra configuration, the proposed solutions in [171] and [172] were also tested. The first one cuts encoding time by 39.39% while increasing the BD rate by 0.86%. The second one reduces encoding time by 44.65% while increasing the BD rate by 1.32%.

QTMT, intra prediction modes, and MTS coding tools were the focus of these studies. Most of these studies focus on VVC block partitioning, primarily for intra-frame prediction, as its method has a much higher computational overhead than previous standards. In addition, one project works on speeding up the QTMT decision by taking into account inter-frame prediction, while another focuses on the MTS transforms phase. Some of the mentioned publications offer heuristics based on statistical analysis, while others develop solutions based on machine learning, such as the Bayesian theorem, decision trees, SVM, and CNN.

These works use encoder context information and/or statistical information of block samples to build efficient complexity reduction solutions capable of reducing the number of encoding modes evaluated.

Another work that can be mentioned is [173]. This work evokes optimizations of intra mode and coding unit size decisions using statistical fast decision and deep learning approaches. For the various binary depths of the QTBT structure, a fast intr amode decision method is presented. As a result, deep learning optimization for square blocks is also provided. The results demonstrate that by combining these two techniques, the complexity of the VVC encoder may be considerably reduced.

From another side, instead of changing the already integrated algorithms, some works seek rather to find coding techniques that can be neglected in some particular scenarios while maintaining a high level of coding efficiency. This might lead to the creation of application-oriented profiles, in which some tools are deactivated by default in the high-level syntax. [174] proposes an optimization framework to determine the efficiency at low resolution and low bitrate of some of the new coding tools introduced in VVC. It has been shown shown in such coding conditions, reduction in terms of encoding complexity may be obtained by disabling some of the VVC coding tools, with a negligible impact in terms of compression efficiency. Fixing sub-sets of tools for all kinds of sequences can bring advantages for some scenarios but on the other hand can damage the coding efficiency on many other scenarios.

4.3.2 Presentation of VVenc

The VVC test model (VTM) serves as a common reference implementation, i.e., a test bed for evaluation and verification of proposed technologies during standardization. While VTM used to be the only publicly available encoder and decoder implementation of the VVC standard, it is aimed at correctness, completeness and readability and should not serve as a real-world example of an VVC encoder and decoder. Therefore, Fraunhofer HHI initiated a Versatile Video Encoder (VVenC) development to provide a publicly available, fast and efficient VVC encoder implementation [175]. The VVenC software is based on VTM, with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations, improved encoder search algorithms and basic multi-threading support to exploit parallelization. Additionally, VVenC supports real-world encoder features, including frame-level rate control and perceptually optimized encoding in order to provide a flexible, fast and easy to use video encoding solution for the VVC standard. Bit-streams encoded with VVenC can be decoded by any VVC standard compliant decoder, e.g. the VTM-11.0 reference software decoder. Alternatively, the fast Fraunhofer

Versatile Video Decoder (VVdeC) solution can be used. Regarding the advantages that VVenc provides, the approach proposed in this chapter is tested using this software.

The source code of the Fraunhofer Versatile Video Encoder, VVenC, is accessible for both business and non-commercial usage on GitHub under a 3-clause BSD copyright license [60]. After an initial release in September 2020, barely two months after VVC was completed, version 1.0.0 was issued in May 2021, and the findings were provided.

VVenC has been taken from the VTM reference software at first. Its primary goal is to keep VTM's great coding efficiency while running at a much higher speed and offering real-world application functionality. While VVenC is developed in C++, the software package also contains a C library interface and two sample apps that may be used as independent command line encoders: a simple application for beginners and an expert application that looks like the VTM interface. A raw YUV video input is converted into a VVC compatible bitstream by VTM and VVenC.

4.3.2.1 Main features

The encoder is much more user-friendly than the VTM software. It contains five preset settings that allow for varying trade-offs between encoding speed and compression efficiency: faster, fast, medium, slow, and slower. While the slower option achieves complete VTM compression in less than half the time, the other presets give significantly quicker encoding at the expense of increased bitrate. Multithreading can be used to get additional speed with minimal compression efficiency degradation. Thread number to runtime reduction scaling is achieved via a combination of coding tree unit line and frame parallel processing [176]. The implementation of single-pass and two-pass frame-based rate control modes has resulted in extremely efficient encoding at specified bit rates. Both have been tuned for multi-threaded operation, with frame-level parallelism implemented without sacrificing speed. VVenC also uses a weighted extended PSNR (XPSNR) distortion metric to incorporate subjective improvements. Using a simplified model of the human visual system to determine weights, this overcomes the low correlation of frequently used Sum-of-Squared Differences (SSD) or Sum-of-Absolute Differences (SAD) computations with observed distortions. Furthermore, XPSNR may be computed block-by-block and therefore integrated into rate-distortion decisions at the block level. It is performed using local Quantization Parameter Adjustments (QPA) and is intended to increase rate control accuracy as well as performance [177]. VVenC is compatible with the Main 10 profile (4:2:0 chroma subsampling and up to 10 bit per sample) and designed for both Standard Dynamic Range (SDR) and High Dynamic Range (HDR) video with wide color gamut as of the current version. Only one slice and tile per picture are currently supported, with no sub-pictures. By integrating all Main 10 profile screen content coding techniques and offering a restricted encoding mode designed for usage in open-GOP adaptive streaming with resolution change [178], VVenC anticipates a wide range of application situations.

4.3.2.2 Results

Figure 4.2 shows the multi-threaded runtime and PSNR-based Bjontegaard Delta (BD) rate compression performance for VVenC 1.0.0 compared to the HEVC reference software (HM-16.22) (negative values indicate bit-rate decrease) [4]. The results are based on JVET random access conditions encoding of HD and UHD sequences from the JVET common test conditions [179] test-set. In addition, the graphic shows findings for VTM 12.0 as well as two more publicly accessible encoders: HEVC (x265 3.4) and AV1 (aomenc 3.0). When VVenC is run multi-threaded, the findings indicate that it can attain VTM coding efficiency at significantly lower runtimes. It offers a consistent 20% bit-rate decrease at similar runtimes

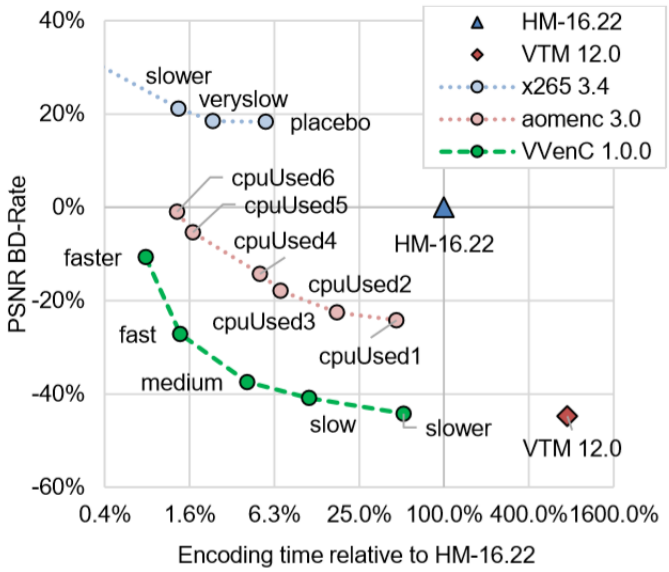


FIGURE 4.2: Runtime and BD rate VVenC 1.0.0, x265 3.4, aomenc 3.0 and VTM 12.0 compared to HM 16.22. All codecs but VTM and HM are run multithreaded with 8 threads [4].

when compared to aomenc. VVenC's three faster presets can equal the three slowest x265 presets in terms of encoding time while saving about 50% in bit rate. However, no VVenC presets that match the encoding performance of faster x265 presets are currently available.

Initial versions of VVenC have been subjectively evaluated with VTM and HM for UHD (VVenC v0.1) and HD (VVenC v0.1) in recent JVET VVC verification tests (VVenC v0.3). These studies have revealed that VVenC in medium preset with subjective optimizations surpassed VTM visually in a controlled laboratory environment with naive observers.

It is important to note that VVenC only uses a subset of VVC's coding tools. The standard's modular architecture enables for the configuration of many operational points using a variety of tools. Furthermore, lowering partitioning fidelity can reduce encoding complexity even further. In this approach, VTM may be set with an operating point that allows for a 30% BD-rate reduction over HM while only using 120 % of the encoding duration of the reference software HM. Despite software optimization, it was shown that the VTM software can be speeded up by 33% by adding further platform-specific SIMD-optimizations and eliminating some hotspots. The VVenC software can obtain even more efficient operating points by using an enhanced encoding search method and software implementation. However, changes to the search algorithm may result in coding loss. In the VVenC software's Pareto set of encoder tools, the new speedup outweighs the loss, resulting in superior tradeoffs. With all speedups enabled, the operational point is indicated, delivering a 30.35 percent BD-rate decrease over HM at 39% of its runtime. At relatively comparable BD-rate levels, this is about 3 times faster than VTM, or 22.8 times faster than the present VTM CTC configuration at 12.71 percent BD-rate loss. The VTM software encoding complexity is not inherent to the VVC standard, which allows for the flexible configuration of various operating points. Even with much reduced encoding complexity and runtime, as proven by the VTM and optimized VVenC software, the standard can outperform its predecessor HEVC.

Chapter 5

Decoder-side Intra Mode Derivation (DIMD)

5.1 The problem

To achieve 50% gain for VVC over HEVC, a proposed solution is to add new prediction modes, allowing for more effective predictions, and less energy coefficient residues, thus bringing coding gains. On the other hand, this proposal is accompanied by an increase in signaling to encode the chosen prediction mode, which can offset some of the gains that may be obtained. Solutions to address this problem exist in inter mode but not in intra and hence the interest of the method. These solutions consist of deriving the prediction information on the decoder side instead of sending it from the encoder into the bit stream.

5.2 Motivation

5.2.1 Experimental settings

All the work in this section is implemented using C++ Language (Visual Studio - Version 15 – 2017) and integrated in the code of the test model released by JVET which is called VTM. Tests are performed as specified in the JVET Common Tests Conditions (CTC) [180] on different QP values. CTC are defined by JVET to conduct experiments in a well-defined environment and ease the comparison of the outcome of experiments. The sequences, classified into classes named from A to E based on their resolutions, are presented in Table 5.1.

The tests are launched on a server hosting two Intel(R) Xeon(R) Platinum 8180 CPUs running at 2.50GH by writing some scripts. Generally, the tests are launched on sequences encoded with the All-Intra configuration, unless it is mentioned that a test is done on another type of configuration.

5.2.2 The motivation of the work

We start by making sure of the relevance of our choice: focusing on reducing the signaling related to the luma intra prediction modes. To do so, we run a first test measuring the proportion of bits dedicated to the signaling of those modes in the total bit stream. Table 5.2 shows these proportions in VTM-1 with All-Intra configuration, since at this level of the thesis, VVC has been still in his early stage.

These percentages define our maximum upper bound gains. In other terms, as our method is aimed to reduce the luma intra prediction mode signaling, thus in the best case, we can reach a reduction equal

Class	Resolution	Sequence name
A	**3840x2160 (4K)**	Tango2 Campfire Foodmarket4 Catrobot1 Daylightroad2 Parkrunning3
B	**1920x1080 (Full High Definition (Full HD))**	MarketPlace RitualDance Cactus BasketballDrive BQTerrace
C	**832x480**	BasketballDrill BQMall PartyScene RaceHorses
D	**416x240**	BasketballPass BQSquare BlowingBubbles RaceHorses
E	**1280x272**	FourPeople Johnny KristenAndSara

Table 5.1: Different classified sequences from CTC.

Class	Sequence name	Proportion (%)
B(1920x1080)	MarketPlace	4,70
	RitualDance	10,50
	Cactus	9,34
	BasketballDrive	10,66
	BQTerrace	8,34
C(832x480)	BasketballDrill	14,03
	BQMall	10,99
	PartyScene	9,43
	RaceHorses	8,53
D(416x240)	BasketballPass	11,42
	BQSquare	9,67
	BlowingBubbles	10,12
	RaceHorses	10,53
E(1280x272)	FourPeople	11,4
	Johnny	11,5
	KristenAndSara	11,3

Table 5.2: Proportion of bits dedicated to intra luma mode signaling.

QP	22	27	32	37
Proportions(in %)	5,73	8,24	11,49	15,15

Table 5.3: Proportions based on QP values.

to these percentages. Here, the best case scenario refers to the case where all modes can be derived at the decoder side, i.e., no need to send them in the bit stream.

As we can clearly notice in Table 5.3, the proportion dedicated to the signaling of intra luma prediction modes increases with the QP values raise from 22 to 37. This happens for a reason that the growth of QP value causes the decrease of the number of coefficient to be encoded after quantization which will lead to decrease the total bit rate while the amount of bit dedicated to coding intra luma prediction mode remains the same and this will lead to augment the proportion of signaling intra luma modes compared to the total bitrate.

5.3 Overview of the method

Our work is proposed to improve the efficiency of intra-prediction in the context of next generation video codecs, referred to as Decoder-side Intra Mode Derivation (DIMD). When using DIMD, an analysis of the texture surrounding the current block is performed to extract predominant prediction directions. These directions are then combined together by means of prediction fusion, to compute a final prediction for the current block. Due to the fact that the analysis is performed both at the encoder and decoder side, DIMD does not require any overhead information to be explicitly signaled in the bitstream. In consequence, DIMD is able to considerably reduce the amount of information required to compute the intra-prediction. Moreover, the fusion of different modes to compute the prediction of a single block may ensure that more accurate prediction samples can be produced, potentially increasing the prediction accuracy even in the case of complex textures. This may reduce the need to rely on finer block partitioning or more complex transform or entropy coding solutions, hence reducing the amount of bits necessary to compress the frame.

Intra-prediction is typically based on the assumption that neighbouring reference samples carry information on the content of the current block. This information can be exploited by means of various possible intra-prediction modes, with the goal being to predict the block as accurately as possible. Traditional intra-prediction methods include directional intra-prediction, Planar prediction, or DC prediction. When using directional modes, the reference samples in the neighbourhood of the block are interpolated and then propagated within the block depending on a certain angle. Conversely, DC and Planar mode rely on the averaging or filtering of neighbouring samples to produce a prediction which can accurately predict gradients or smooth areas of content. Directional intra-prediction is a simple yet effective way of producing accurate prediction samples in highly-textured picture regions. Obviously, having more directions used, implies closer approximations of the actual content of the block, hence more accurate block predictions. This is why state-of-the-art video coding standards include a large number of possible directions: up to 65 different angles can be considered in the latest VVC draft for a given block [181]. On the other hand, allowing this large variety of options requires a considerable number of bits to signal the selected intra mode, which can negatively impact the efficiency of the codec.

Given that the information in the reference region in the neighbourhood of the current block can be exploited to compute intra-predicted samples, it is sensible to assume that these same reference samples may carry additional information to also accurately infer the texture directionality in the block. Such assumption is at the basis of the method presented in this report.

5.4 Description of the Implicit Intra Mode Derivation

In details, a texture analysis scheme based on the Histogram of Gradients (HoG) is proposed [182]. Input to this process is a set of reference reconstructed samples extracted from the neighborhood of the current block. A set of three reference rows and columns above and to the left of the current block is used, referred to as the template region. The corresponding design is shown in Figure 5.1-(a).

For each sample s considered for the HoG analysis, a 3×3 window W^s of neighbouring samples is defined, as shown in Fig 5.1-(b). Then, given a 3×3 gradient operator (e.g. Sobel filter) S, the horizontal and vertical angle intensities G^s_{hor} and G^s_{ver} for sample s are computed as:

$$G^s_{dir} = W^s * S_{dir}, \quad dir = hor, ver. \tag{5.1}$$

Finally, an angle gradient descriptor D^s for sample s is computed as:

$$D^s = \begin{cases} \tan\theta^s = \frac{G^s_{ver}}{G^s_{hor}} & \text{if } G_{ver} > G_{hor}. \\ \\ \cot\theta^s = \frac{G^s_{hor}}{G^s_{ver}} & \text{if } G_{ver} \leq G_{hor}. \end{cases} \tag{5.2}$$

The computed value D^s can be taken as a descriptor of the texture direction within the 3×3 window of samples surrounding the sample s. Given that directional intra-prediction modes are defined by the angle used to extrapolate the prediction within the block, it is then possible to map the value D^s to the closest angle among those allowed by the available intra-prediction modes. In VVC, a total of 65 angular IPMs unevenly cover a range of 180 degrees. The intentional unevenness of this design provides a finer texture modeling of horizontal and vertical patterns that are more common in video contents. More precisely, the elapsed angles between consecutive IPMs around the vertical and horizontal IPMs are smaller than those around the diagonal IPMs. Figure 5.2 presents the distribution of angular IPM in the second quarter of the angle range.

Assuming that a total of M possible directional intra-prediction modes are available, a specific mode m_s is then selected as the mode whose angle is the closest to the direction defined by D^s, as shown in Figure 5.1-(c).

This process can be repeated for all samples in the template region considered for the HoG analysis as in Figure 5.1-(a). The HoG can therefore be computed, denoted as $H = \{H[m] |\ m = 0, \ldots, M-1\}$. All entries in the HoG are first initialized with zeroes. For each sample s, the corresponding mode m_s is considered. Then, the local intensity of m_s is computed as:

$$I^s = |G^s_{hor}| + |G^s_{ver}|. \tag{5.3}$$

Finally, the HoG is updated with the local intensity value as:

$$H[m_s] = H[m_s] + I^s. \tag{5.4}$$

Upon completion of this process, each entry $H[m]$ in the HoG eventually corresponds to the cumulative intensity of intra-prediction mode m, based on the texture analysis performed on the template region. The complete process is illustrated in Figure 5.1.

Clearly, the information in the HoG can be exploited while performing intra-prediction for the current block, assuming some degree of spatial correlation between the reference region and the current block.

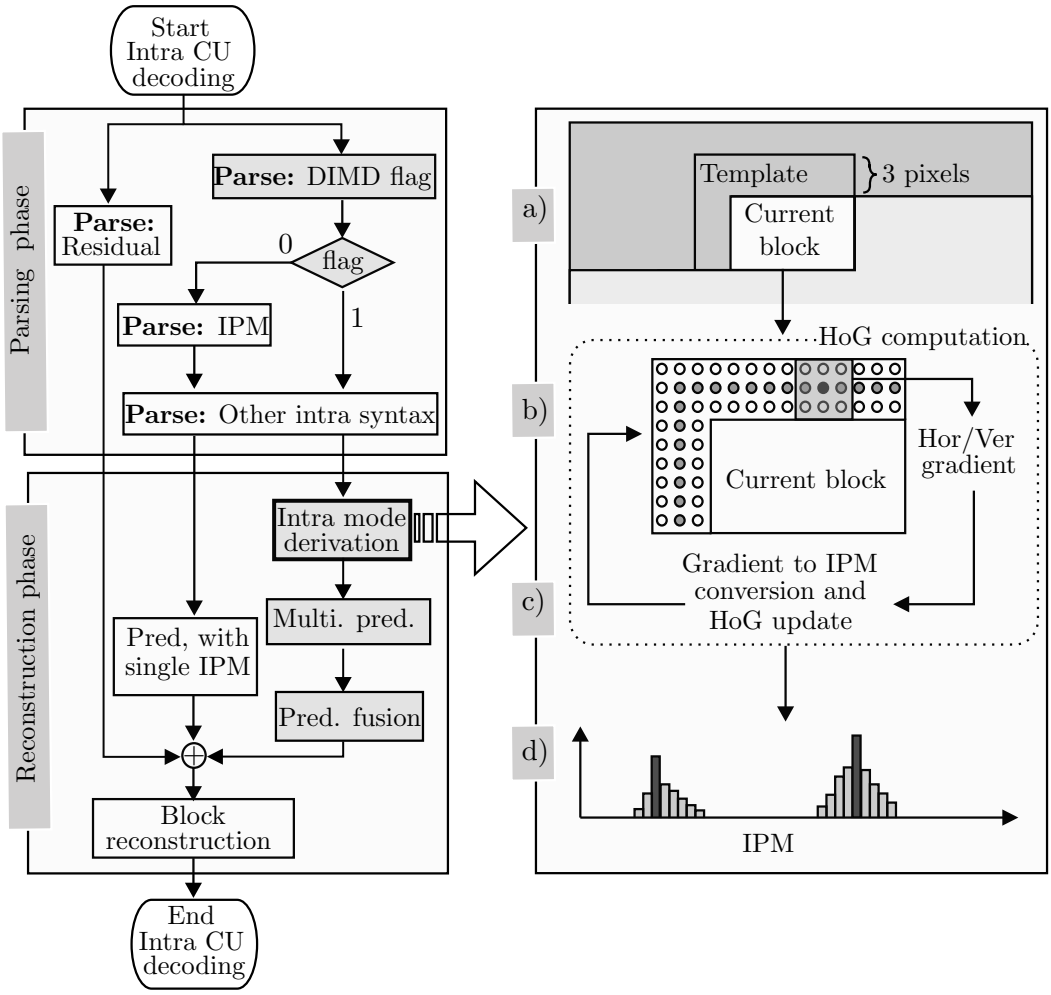


FIGURE 5.1: The proposed design of the VVC decoder, integrated with DIMD. The parsing phase is modified to incorporate the use of the DIMD flag. In case of DIMD blocks, a different reconstruction path is taken which includes the intra mode derivation and multiple prediction fusion steps.

Specifically, the proposed method is based on the assumption that the directional modes with the highest cumulative intensities in the HoG are likely to be useful to predict the texture in the current block. This is illustrated in Figure 5.4. A frame of a test sequence is partitioned into blocks. The HoG analysis is then performed on a variety of blocks in the frame, and a corresponding direction is identified based on the highest bar in the HoG. The directions predicted using the proposed analysis are then shown in the figure to highlight how these match with the content in the picture. As can be seen, the predicted directional modes match the actual texture direction in most cases.

5.5 Prediction Fusion

Areas with complex content, in particular multiple direction texture patterns, are often split into small blocks for prediction. This is due to the fact that even state-of-the-art video coding solutions (e.g. VVC)

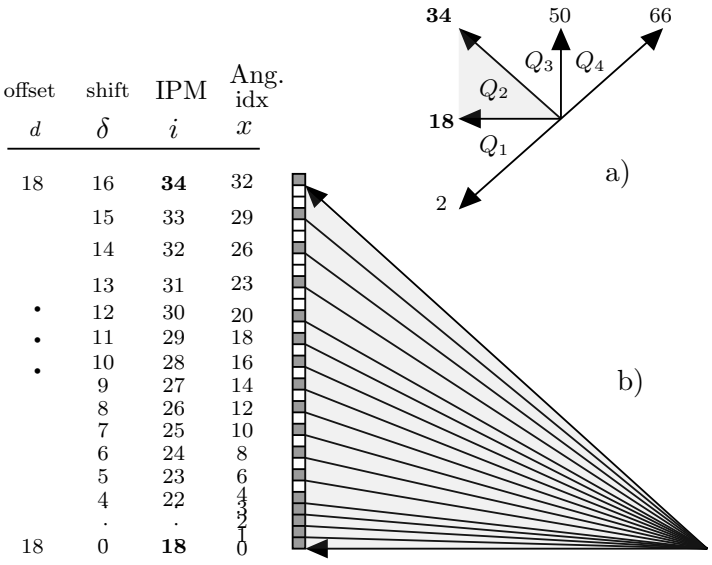


FIGURE 5.2: Angular IPMs of VVC (i.e., IPM #2 to IPM #66): a) Covering the range of 180 degrees in four quarters (i.e., Q_1 to Q_4); b) Uneven angle distribution in the second quarter (i.e., the range between IPM #18 to IPM #34).

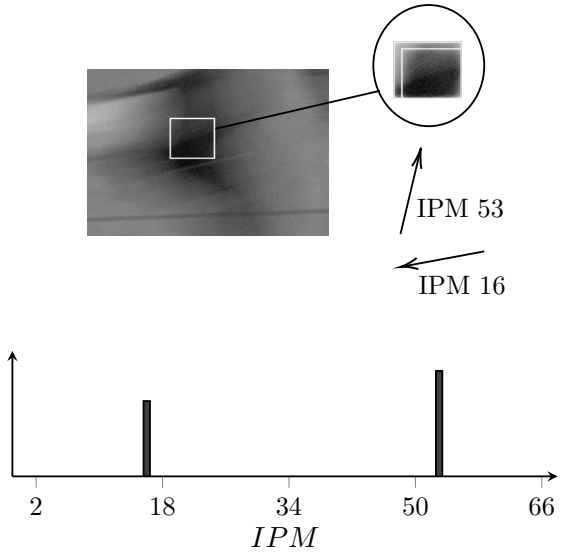


FIGURE 5.3: A 32×32 block example from BasketballDrive. The major texture directions can be visually detected. As a consequence, the calculated HoG from each block shows two IPMs that clearly correspond to the dominant directions of the texture in the image.

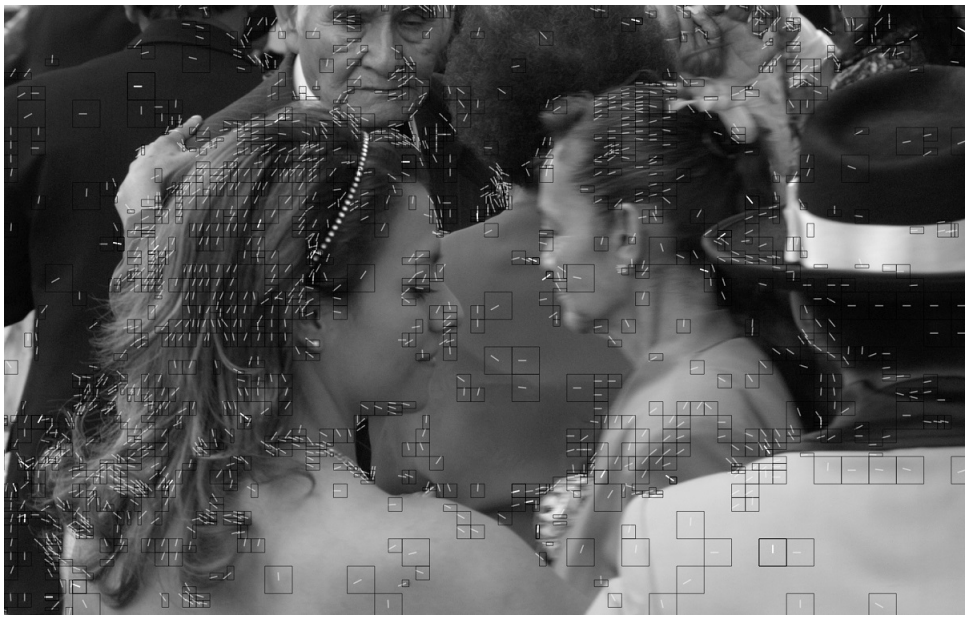


FIGURE 5.4: Correlation between content texture and implicitly derived angular IPMs using DIMD.

do not support the use of more than one IPM for prediction of such blocks. Therefore, their image region is further split into smaller blocks to adapt the single IPM designs.

The use of multiple IPMs for block prediction has been sparsely studied in the literature [183, 184, 185, 186]. Most of these studies propose explicit signaling of additional IPMs. The assumption is that a prediction with multiple IPMs would be more accurate, hence, resulting in a smaller residual and/or partitioning rate. Nonetheless, this goal is rarely achieved and the rate overhead of explicit signaling of multiple IPMs is still dominant, with respect to the residual and partitioning rate saving.

To address the problem of explicit fusion, the texture information obtained by the proposed intra mode derivation is used for implicit fusion. Figure 5.3 shows an example that demonstrates the effectiveness of this idea, where the entries stored in the histogram of gradients are highly correlated to the texture information. Different configurations of fusion were tested to pick up finally the best one that will be presented in this section.

5.5.1 Fusion modes

Higher number of fusion modes, in general, provides more flexibility to the intra-prediction module and, if exploited efficiently, can improve the prediction accuracy. However, such flexibility comes at the cost of both encoder and decoder complexity. As a compromise between the performance and throughput, several fusion configurations using either two or three fusion modes were tested. Adaptive and fixed fusion mode selection are used in the tested configurations.

As for the fixed mode, different IPMs of VVC have been evaluated and after observing outstanding performance, the planar mode has been selected as the fixed fusion mode [183]. This performance can be explained by noting that in all intra configuration, about 15 to 30 percent of blocks in the Common Test Condition (CTC) sequences are coded with the planar mode.

Cfg	Modes used	Weights		
		w_0	w_1	w_2
1	$M_0 = H_0, M_1 = H_1$	$\left\lfloor 64 \times \frac{H[m_0]}{H[m_0]+H[m_1]} \right\rceil$	$\left\lfloor 64 \times \frac{H[m_1]}{H[m_0]+H[m_1]} \right\rceil$	–
2	$M_0 = H_0, M_1 = Planar$	32	32	–
3	$M_0 = H_0, M_1 = H_1, M_2 = Planar$	$\left\lfloor 43 \times \frac{H[m_0]}{H[m_0]+H[m_1]} \right\rceil$	$\left\lfloor 43 \times \frac{H[m_1]}{H[m_0]+H[m_1]} \right\rceil$	21

TABLE 5.4: Selected configurations for prediction fusion.

Unfortunately using the Planar mode totally disregards any directional texture present in the block. Clearly, combining directional modes with Planar may therefore be highly beneficial [183], as the fusion of these techniques could accurately predict directional textures, while preserving the smoothing quality of Planar prediction.

Regarding the selection of the adaptive fusion mode, different sources of mode can be considered. First source is the HoG information obtained in the previous section. More specifically, up to two IPMs corresponding to the highest histogram amplitude can be used as fusion mode. Such modes are referred to as HoG modes, As the derived modes from the HoG are all angular, one can expect that the combination of such modes with the planar mode could produce accurate predictions for certain types of content.

Another source of the adaptive fusion mode selection is the Matrix-based Intra Prediction (MIP) mode [9]. This tool replaces the angular reference projection of the traditional intra prediction with matrix multiplication operations. For this purpose, a set of matrices are trained for textures in different angles, defined as MIP modes. The design of this tool in VTM-5 ensures a correspondence between the MIP modes and the regular angular IPMs of VVC. For that, the MIP design consists of three fixed tables, depending on the block size as mentioned above, that maps each conventional intra prediction mode to a corresponding MIP mode and vice versa. In the proposed fusion configuration, this property has been exploited to use equivalent MIP mode to an HoG mode as fusion mode. The IPM mode is mapped to the corresponding MIP mode, by using the existing mapping functionality of MIP tool, and use the MIP prediction as fusion mode along with the conventional IPM mode. It is important to highlight that this mapping functionality between MIP and IPM modes was present in the algorithm of VVC until the VTM-5.

5.5.2 Selected fusion configuration

To understand the performance of the proposed DIMD with fusion, several configurations were tested. These configurations consist of combining different numbers of modes (two or three) from different sources of fusion modes (HoG modes, MIP modes and planar). Examples of these prediction fusion configurations can be obtained by combining the first two HoG modes, or by combining the first HoG mode with the planar, or for instance, by combining the first HoG mode, its corresponding MIP mode and the second HoG mode. Taking into account all performance measurement aspects, we picked the third fusion configuration that provides the best results. These configurations are detailed in Table 5.4.

During the analysis, we have observed that best two HoG modes of a given DIMD block have a high correlation with the conventional best intra modes selected using the rate-distortion cost. We also noticed that for DIMD blocks with relatively simple directional patterns, the first two intra modes are usually close. The interpretation can be as follows: fusing adjacent prediction modes provides even finer angle representation than what VVC provides with 65 angular modes. On the contrary, it was observed that

for DIMD blocks with complex textures, the first two HoG modes are relatively distant, which can be explained by the presence of multiple local angular textures within the block.

Given above observations, a total number of three fusion modes are used in the selected configuration. The first and second fusion modes are always the HoG modes corresponding to the two highest histogram amplitudes. However, the choice of the third fusion mode depends on the relative index of the two selected HoG modes. More precisely, if the absolute difference between the IPM index of the two HoG modes is more than a threshold thr, then the block is assumed to have multiple dominant local angular textures, most likely with a smooth transition between them. In this case, the third fusion mode is selected to be planar mode. Otherwise, the block is assumed to have simple directional pattern that corresponds to the best HoG modes. Therefore, the third fusion mode in this case is selected to be the MIP mode equivalent to the first HoG mode. In our experiments, thr is selected to be 10 and the three fusion modes are combined.

5.6 Experimental Results

5.6.1 Implementation

At the time of writing the manuscript, VVC was under development. In order to assess the presented technique, DIMD was implemented in the context of the VVC Test model (VTM) 5.0. Note that the proposed method is applied only on luma blocks. Obviously, the integration of DIMD into the VVC coding scheme requires normative changes to the way intra-prediction is signalled in the bitstream and parsed by the decoder. More precisely, a new block-level syntax element was introduced, to indicate whether DIMD is used or not on a block. If the DIMD flag is parsed to be true, then DIMD is used and no other intra prediction information is subsequently parsed. Later, when performing the intra-prediction, the corresponding prediction fusion is performed according to the configuration being used. Conversely, if the DIMD flag is parsed to be false, the proposed method is not used, and the rest of the parsing and decoding process is performed according to the conventional VVC draft [181]. It is important to note that the development of this work was strongly influenced by feedback gathered during the VVC standardization process, in order to satisfy hardware friendliness requirements, decoder complexity issues, and general implementation constraints. As a result, rather than merely maximizing the coding efficiency performance of the tool, these aforementioned limitations have also been taken into consideration. All tests were carried out using the encoder configuration and sequences defined in the Common Test Conditions recommended by JVET [180].

The proposed DIMD algorithm only affects intra-prediction. However, given that intra blocks may be used in inter slices as well, results under Random Access and Low-Delay-B are provided in addition to the All-Intra configuration whereby only intra blocks are used to encode all frames in the sequence. The simulations were conducted on an Intel Xeon Platinum 8180 machine with AVX2, turbo boost and hyperthreading, with CentOS Linux 7 and a GCC 4.8.5 compiler.

5.6.2 Statistics

It is interesting to note that the proposed method is quite often selected and prove its benefits in a variety of different situations. For this purpose, Table 5.6 shows the percentage of intra blocks that use DIMD depending on the Quantization Parameter (QP) used for encoding and the block size. The selection rate of a newly added tool can reflect its effectiveness in a video codec. Moreover, Figure 5.5 visualizes a

FIGURE 5.5: Spatial distribution of DIMD blocks in different sequences of Tango (3840×2160) and RitualDance (1920×1080).

population of DIMD blocks. One interesting point in this figure is that DIMD is used not only for sharp angular contents, but also both simple and complex content.

5.6.3 Performance

Table 5.5 presents the coding performance of the selected fusion configuration of DIMD. In this table, Bjontegaard-Delta Rate (BDR) [52], Encoder run-Time (ET) and Decoder run-Time (DT) are presented for All-Intra (AI), Random Access (RA) and Low Delay B (LB) configurations. Noteworthy to mention that negative BDR values correspond to efficiency gains with respect to the anchor and ET/DT values more than 100 indicate the increase in the run-time.

As can be seen in 5.5, the average BDR gain of DIMD on top of VTM-5 are -0.60%, -0.31% and -0.23%, in AI, RA and LB configurations, respectively. Moreover, all performance metrics are rather consistent which proves that the DIMD tool can be useful for different content characteristics.

5.6.3.1 Coding efficiency

Additional experiments were carried out to evaluate the performance of DIMD in more testing conditions that are of particular benefit for specific practical applications. In particular, the proposed method was tested in both high-rate conditions, corresponding to using four QP values ranging between 15 and 32, as well as low bitrate conditions, corresponding to using four QP values ranging between 27 and 42. In the former case, average BDR gains of -0.59% were observed with the encoder and decoder runtimes of 110% and 103%, respectively. In the latter case, average BDR gains of -0.64% were observed, with encoder and decoder runtimes of 109% and 104%, respectively. Again, these tests reveal consistent gains across a variety of conditions and as such could be useful for different applications targeting a wide range of bitrates.

5.6.3.2 Complexity

According to Table 5.5, the run-time complexity overheads of the proposed DIMD in AI configuration are 109% and 104%, at the encoder and decoder sides, respectively. Different aspects of the proposed design impact the complexity. More precisely, following modules are added by DIMD to the encoder and/or decoder sides of VTM:

Class	Sequence	All Intra			Random Access			Low Delay B		
		BDR	**ET**	**DT**	**BDR**	**ET**	**DT**	**BDR**	**ET**	**DT**
A1	Tango2	-0.97%	108%	104%	-0.51%	102%	102%	-0.18%	103%	101%
	FoodMarket	-0.97%	108%	104%	-0.37%	102%	102%	-0.29%	102%	100%
	Campfire	-0.59%	109%	104%	-0.44%	102%	102%	-0.25%	103%	101%
	Average	**-0.84%**	**108%**	**104%**	**-0.44%**	**102%**	**102%**	**-0.24%**	**103%**	**101%**
A2	CatRobot	-0.55%	109%	103%	-0.26%	102%	102%	-0.17%	102%	101%
	DaylightRoad2	-0.43%	106%	104%	-0.20%	102%	102%	-0.26%	102%	101%
	ParkRunnig3	-0.47%	108%	104%	-0.17%	102%	102%	-0.32%	102%	101%
	Average	**-0.48%**	**108%**	**104%**	**-0.21%**	**102%**	**102%**	**-0.25%**	**102%**	**101%**
B	MarketPlace	-0.47%	107%	104%	-0.21%	101%	101%	-0.24%	102%	101%
	RitualDance	-0.85%	109%	104%	-0.43%	101%	102%	-0.29%	101%	102%
	Cactus	-0.51%	108%	104%	-0.34%	102%	102%	-0.28%	102%	101%
	BasketballDrive	-0.68%	109%	105%	-0.46%	102%	102%	-0.25%	102%	100%
	BQTerrace	-0.25%	108%	104%	-0.14%	102%	102%	-0.13%	102%	101%
	Average	**-0.55%**	**108%**	**104%**	**-0.31%**	**102%**	**102%**	**-0.24%**	**102%**	**101%**
C	BasketballDrill	-0.32%	109%	104%	-0.34%	102%	101%	-0.25%	101%	101%
	BQMall	-0.73%	108%	104%	-0.38%	102%	102%	-0.18%	102%	102%
	PartyScene	-0.61%	109%	105%	-0.36%	102%	102%	-0.23%	102%	101%
	RaceHorses	-0.70%	108%	105%	-0.37%	102%	101%	-0.27%	102%	101%
	Average	**-0.59%**	**109%**	**105%**	**-0.36%**	**102%**	**102%**	**-0.23%**	**102%**	**101%**
D	BasketballPass	-0.67%	109%	104%	-0.36%	102%	102%	-0.04%	102%	101%
	BQSquare	-0.42%	110%	104%	-0.18%	102%	101%	-0.29%	102%	101%
	BlowingBubbles	-0.89%	110%	104%	-0.25%	103%	102%	-0.29%	103%	102%
	RaceHorses	-0.68%	111%	104%	-0.28%	102%	101%	-0.14%	103%	102%
	Average	**-0.67%**	**110%**	**104%**	**-0.27%**	**102%**	**102%**	**-0.19%**	**103%**	**102%**
E	FourPeople	-0.74%	108%	104%	-0.41%	102%	102%	0.04%	102%	101%
	Johnny	-0.79%	108%	104%	-0.48%	103%	102%	-0.11%	103%	101%
	KristenAndSara	-0.72%	109%	104%	-0.47%	102%	103%	-0.45%	102%	101%
	Average	**-0.75%**	**108%**	**104%**	**-0.45%**	**102%**	**102%**	**-0.17%**	**102%**	**101%**
F	BasketballDrillText	-0.28%	111%	104%	-0.30%	102%	102%	-0.27%	102%	101%
	ArenaOfValor	-0.80%	111%	104%	-0.39%	102%	102%	-0.28%	102%	101%
	SlideEditing	-0.16%	110%	104%	-0.17%	102%	102%	-0.23%	102%	102%
	SlideShow	-0.13%	110%	104%	-0.24%	103%	103%	-0.21%	102%	101%
	Average	**-0.34%**	**111%**	**104%**	**-0.28%**	**102%**	**102%**	**-0.25%**	**102%**	**101%**
All		**-0.6%**	**109%**	**104%**	**-0.31%**	**102%**	**102%**	**-0.23%**	**102%**	**101%**

TABLE 5.5: Coding performance of the proposed DIMD with fusion method, in All-Intra (AI), Random Access (RA) and Low-Delay B (LB). The reference anchor of this experiment is VTM-5.

			Block width				
			4	**8**	**16**	**32**	**64**
Block height	QP22	**4**	36%	39%	36%	31%	–
		8	34%	32%	29%	24%	–
		16	32%	29%	27%	21%	–
		32	34%	29%	28%	24%	–
		64	–	–	–	–	17%
	QP37	**4**	41%	31%	34%	31%	–
		8	28%	27%	28%	26%	–
		16	21%	24%	26%	25%	–
		32	23%	23%	26%	26%	–
		64	–	–	–	–	28%

TABLE 5.6: Selection rate of the proposed DIMD with fusion, in block sizes and with two Quantization Parameters (QP).

- DIMD full RD-check: To determine whether or not to use DIMD for a block, we compute its exact rate-distortion cost rather than its estimation. Afterwards, we compare it against the cost of regular coding. This process, called full RD-check, is specific to the encoder side.
- Intra mode derivation: This process is applied both at the encoder and decoder sides, however, with a subtle difference in terms of complexity. At the encoder side, each coding mode combination (e.g. partition, transform, reference line, etc) of a given image region must perform the intra mode derivation step. Conversely, the decoder applies this process once and only on DIMD blocks.
- Fusion: This process is also shared between encoder and decoder but is performed differently on each side.

Finally, a set of profiling experiments are carried out to estimate the impact of each additional codec module of DIMD on the run-time. For this purpose, the execution run-time of VTM source codes corresponding to each DIMD module are isolated. Then the approximate complexities of the isolated parts are extracted by running VTM several times and averaging. Table 5.7 shows the encoder run-time overhead of different DIMD modules. As can be seen, the full RD-check of the DIMD mode plays the most important role in the encoder run-time increase. This is due to the fact that a single full RD-check at the encoder side consists of several computationally complex operations such as prediction, residual calculation, transformation/quantization and eventually rate and distortion calculation. While the intra mode derivation and the multiple prediction modules are far less complex. Similarly, as can be seen in Table 5.7, the process related to prediction fusion has more impact on the decoder side complexity. This can be justified by noting that the complexity order of prediction function is $O(N^2)$, while the complexity order of HoG computation is $O(N)$.

DIMD module	Encoder run-time overhead
Full RD-check	89%
Intra mode derivation	6%
Fusion	5%
Total encoder overhead	**100**%
DIMD module	Decoder run-time overhead
Intra mode derivation	28%
Fusion	72%
Total decoder overhead	**100**%

TABLE 5.7: Share of each DIMD module from total run-time overhead of encoding and decoding each DIMD block.

Chapter 6

CNN-based Switchable Sub-pel Interpolation Filters (C-SSIF)

6.1 The problem

The efficiency of video compression methods mainly depends on the inter prediction. Motion compensation, a fundamental stage in the inter prediction, has become more efficient with the introduction of fractional-sample (frac) accuracy in modern video coding technology. However, the sub-pixel interpolation process uses fixed filters that seem not able to fit for all types of video signals with various kinds of structures and contents. Therefore, interpolation filters for each fractional sample position need to be more adaptive to the content and the local characteristics of the image at a fine granularity. Interpolation process should be improved, from being fix to be adaptively dynamic.

Despite H.266/VVC improved motion compensation by introducing an alternative half-pel filter to the DCT-IF, this radical change remains not sufficient to optimally adapt to all types of video contents. Especially since these interpolation filters are not employed adaptively based on the content of the local characteristics, thus they may be unable to minimize the encoding rate for all the possible coding units, e.g., within the same picture.

Moreover, the aforementioned modification is limited to an interpolation filter designed only for half pel precision in VVC. However, the problem remains for the other sub-pel precision. Thus, the first challenge is to set up a new technique able *i)* to generate alternative filters for half pel precision and *ii)* to be extended to support other sub-pel precision, for instance. the quarter pel. Second, in VVC the alternative is somehow explicitly signaled for all possible coding units sizes and modes, which may set off any RD gain from reduced residuals. These are mainly the two challenges our contribution considers.

6.2 Motivation

[5] proposes to alternate the DCT-IF filter with two different filters switchable at coding unit level [187]. Motivated by the experimental evidence of [5], we propose two alternative filters to DCT-IF with different cut-off frequencies, giving thus to the predictor, the possibility of retaining high frequencies when useful, or to be blurrier otherwise.The idea is to give the encoder different interpolation filtering options with clear differences in their frequency responses.

The experiments show consistently RD gains for specific contents. Interestingly, filters with cut-off frequency lower than DCT-IF seem to be the key towards better encoding performance.

In our proposal Sub-Pel Switchable Interpolation Filters (SSIF), we foresee the two technical challenges, mentioned in Section 6.1, to overcome VVC. In order to answer these two challenges, SSIF proposes

a framework for designing interpolation filters at half and quarter pel precision without any need to add signalization.

Our preliminary experiments confirm that the H.266/VVC encoder efficiency increases when it is allowed to switch between DCT-IF and the proposed alternative filters. This is shown and further discussed in the following.

6.3 Overview of the method

The proposed method consists of deriving one among different filters for interpolation in inter-picture prediction, at the level of each block of the image. Therefore, a technique to design the proposed filters is needed. The steps followed to design these filters are detailed in Section 6.4. Then, the integration of this non-existent flexibility of interpolation filtering into VVC is described in Section 6.5 to show the potential provided gains.

6.4 Sub-pel Interpolation filters Design

This section describes SSIF a framework for designing interpolation filters at half and quarter pel precision.

6.4.1 Filter design

6.4.1.1 Kaiser Window

In signal processing, a window function is a mathematical function that is zero-valued outside of some chosen interval. In typical applications, these functions are non-negative, smooth, "bell-shaped" curves. Most famous window functions are Rectangular window, Triangular window, Parzen window, Blackman window, Hamming window, etc. Although all of these window functions have their advantages and disadvantages, the windowing function used in the proposed innovation is the Kaiser window.

The Kaiser window function is defined as:

$$w(x) = \begin{cases} \frac{I_0(\beta\sqrt{1-(\frac{x}{L})^2}}{I_0(\beta)}, & \text{if } |x| \leq L \\ 0, & \text{otherwise,} \end{cases} \tag{6.1}$$

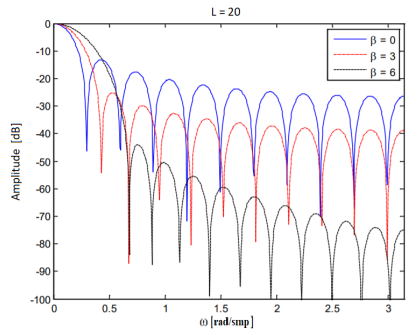
where $I_0(x)$ represents the modified 0 -th order Bessel function of the first kind, and β is a window shape tuning parameter that adjusts the trade-off between stop-band attenuation or pass-band ripple and transition bandwidth. This function can be computed using the following approximation:

$$I_0 \cong 1 + \sum_{k=1}^{K} (\frac{(\frac{x}{2})^k}{k!})^2 \tag{6.2}$$

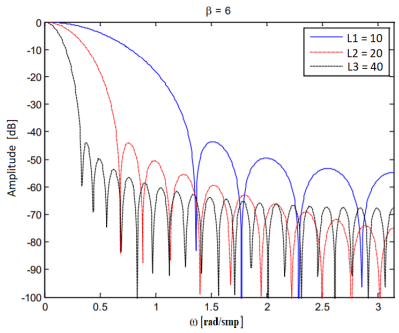
Kaiser window is explicitly chosen for its capacity to adapt the number of lobes, the width and the slope of the window on two simple coefficients:

- The parameter β controls the slope of the window in the time domain, hence it controls the trade-off between the main-lobe width and the side-lobe area in the frequency domain. As β increases, the main lobe increases in width, and the side lobes decrease in amplitude, as can be seen in Figure 6.1a.

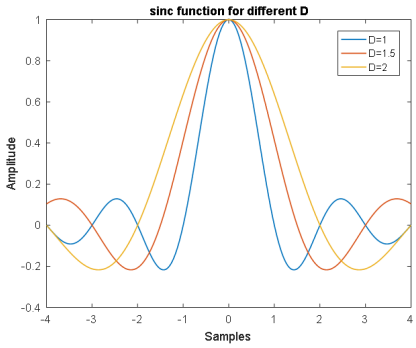
- L represents the window duration in the time domain, and the impact of L on the impulse response in the frequency domain can be seen in Figure 6.1b. As L increases in time, the window gets narrower in frequency.



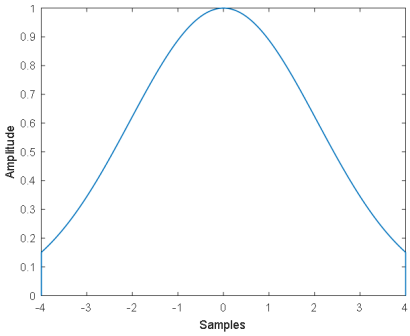
(A) Different values of β.



(B) Different values of L.



(C) Three different values of D.



(D) Kaiser window for $L = 4$ and $\beta = 3$.

FIGURE 6.1: Impulse response of sinc function in time domain for different configurations.

6.4.1.2 Filter coefficients generation

SSIF relies on Kaiser-windowed FIR filters theory (e.g., [188] to downsample chroma samples) with an adjustment of the bandwidth size. Firstly, the non-scaled filter impulse response $g(i)$ is defined as:

$$g(i) = \text{sinc}\left(\frac{\pi(i+p)}{D}\right) * \text{w}(i+p) \tag{6.3}$$

where $\star$ represents the convolution operator and $\cdot$ is regular multiplication, whereas the given symbols are explained as given:

- $\text{sinc}(x) = \lim_{t \to x}\left(\frac{\sin(t)}{t}\right)$.
- D is the low-pass factor of the sinc function.
- p is a fractional phase offset, in this case it is the fractional pixel position.
- w is a Kaiser-windowing function that is symmetric around zero and has a zero value outside the range of plus or minus L ($2*L$ being the number of taps of the filter) and nonzero within that range,

except possibly at the extreme boundaries. In this case, Kaiser window function is used since it is a well-designed adjustable window which is exactly what is needed in order to get the appropriate band passing.

- In this formula, discrete convolution is used:

$$f(i) * g(i) = \sum_{m=-\infty}^{\infty} f(m).g(i-m) \tag{6.4}$$

Secondly, all the coefficients are summed up in order to get the scaled values:

$$T = \sum_{m=-L+1}^{L} g(i) \tag{6.5}$$

Thirdly, the floating-point filter impulse response is adjusted to provide a DC gain equal to 1 by the following formula:

$$h(i) = \frac{g(i)}{T} \tag{6.6}$$

Performing all these steps, we obtain $h(i)$, which represents the output that contains the values of the filter coefficients to be used to interpolate the current block at fractional pixel position p.

6.4.2 Rationale behind the choices made

In signal processing, a sinc function a.k.a. sinc filter is an idealized filter that removes all frequency components above a given cut-off frequency, without affecting lower frequencies, and has linear phase response. The filter's impulse response is a sinc function in the time domain, and its frequency response is a rectangular function. It is so called "ideal" low-pass filter in the frequency sense, perfectly passing low frequencies, and perfectly cutting the higher ones. The zero crossing of the first lobe depends on the argument of the sinc function, which in this case D. As the argument of the sinc function increases, the width of the first lobe decreases. Since the argument of the used sinc function is inversely proportional to D, it can be concluded that as D increases, the width of sinc increases and the filter gets blurrier as shown in Figure 6.1c, and vice versa. In other words, for the proposed filters, which are blurrier than the DCT-IF, D should be bigger than for DCT-IF. Blurrier filters compared to DCT-IF are filters having their cut-off frequency smaller than DCT-IF.

6.4.3 Different implementations of SSIF

We can have several implementations of the main formula defined (6.3) depending on the set of factors and coefficients we choose. A constant Kaiser window function is used with parameters $L = 4$ and $\beta = 3$ in order to obtain the closest filters to the ones used in [5]. The filters used in [5] have proved their efficiency compared to many other tested filters. Therefore we aimed to design similar filters that can be extended for any other precision, namely quarter pel. The kaiser window function is symmetric around 0 and does not have zero values for bandwidth values -4 and 4. This can be seen in the graph of the impulse response presented in Figure 6.1d.

Proposed filters	Parameter D	Pixel resolution (frac)	Parameter p
KCS-Gauss	3	Half Pel Quarter Pel	8 4 or 12
KCS-FlatTop	1,8	Half Pel Quarter Pel	8 4 or 12

TABLE 6.1: Summary table of the parameters tuned in the formula (6.3) used finally to generate the different filters.

frac	DCTIF	KCS-G	[5]	KCS-FT	[5]
0	0,0, 0,64, 0, 0, 0, 0	0,0, 0,64, 0, 0, 0, 0	0,0, 0,64, 0, 0, 0, 0	0,0, 0,64, 0, 0, 0, 0	0,0, 0,64, 0, 0, 0, 0
1	0,1, -3,63, 4, -2, 1, 0	0,1, -3,63, 4, -2, 1, 0	0,1, -3,63, 4, -2, 1, 0	0,1, -3,63, 4, -2, 1, 0	0,1, -3,63, 4, -2, 1, 0
2	-1,2, -5,62, 8, -3, 1, 0	-1,2, -5,62, 8, -3, 1, 0	-1,2, -5,62, 8, -3, 1, 0	-1,2, -5,62, 8, -3, 1, 0	-1,2, -5,62, 8, -3, 1, 0
3	-1,3, -8,60,13, -4, 1, 0	-1,3, -8,60,13, -4, 1, 0	-1,3, -8,60,13, -4, 1, 0	-1,3, -8,60,13, -4, 1, 0	-1,3, -8,60,13, -4, 1, 0
4	-1,4,-10,58,17, -5, 1, 0	0, 4, 13, 20, 18, 8, 1, 0	-1,4,-10,58,17, -5, 1, 0	-1, -4, 12, 35, 25, 0, -3, 0	-1,4,-10,58,17, -5, 1, 0
5	-1,4,-11,52,26, -8, 3,-1	-1,4,-11,52,26, -8, 3,-1	-1,4,-11,52,26, -8, 3,-1	-1,4,-11,52,26, -8, 3,-1	-1,4,-11,52,26, -8, 3,-1
6	-1,3, -9,47,31,-10, 4,-1	-1,3, -9,47,31,-10, 4,-1	-1,3, -9,47,31,-10, 4,-1	-1,3, -9,47,31,-10, 4,-1	-1,3, -9,47,31,-10, 4,-1
7	-1,4,-11,45,34,-10, 4,-1	-1,4,-11,45,34,-10, 4,-1	-1,4,-11,45,34,-10, 4,-1	-1,4,-11,45,34,-10, 4,-1	-1,4,-11,45,34,-10, 4,-1
8	-1,4,-11,40,40,-11, 4,-1	0, 2, 11, 19, 19, 11, 2, 0	0,3,9,20,20,9,3,0	0, -4, 5, 31, 31, 5, -4, 0	0,-3,4,31,31,4,-3,0
9	-1,4,-10,34,45,-11, 4,-1	-1,4,-10,34,45,-11, 4,-1	-1,4,-10,34,45,-11, 4,-1	-1,4,-10,34,45,-11, 4,-1	-1,4,-10,34,45,-11, 4,-1
10	-1,4,-10,31,47, -9, 3,-1	-1,4,-10,31,47, -9, 3,-1	-1,4,-10,31,47, -9, 3,-1	-1,4,-10,31,47, -9, 3,-1	-1,4,-10,31,47, -9, 3,-1
11	-1,3, -8,26,52,-11, 4,-1	-1,3, -8,26,52,-11, 4,-1	-1,3, -8,26,52,-11, 4,-1	-1,3, -8,26,52,-11, 4,-1	-1,3, -8,26,52,-11, 4,-1
12	0,1, -5,17,58,-10, 4,-1	0, 1, 8, 18, 20, 13, 4, 0	0,1, -5,17,58,-10, 4,-1	0, -3, 0, 25, 35, 12, -4, -1	0,1, -5,17,58,-10, 4,-1
13	0,1, -4,13,60, -8, 3,-1	0,1, -4,13,60, -8, 3,-1	0,1, -4,13,60, -8, 3,-1	0,1, -4,13,60, -8, 3,-1	0,1, -4,13,60, -8, 3,-1
14	0,1, -3, 8,62, -5, 2,-1	0,1, -3, 8,62, -5, 2,-1	0,1, -3, 8,62, -5, 2,-1	0,1, -3, 8,62, -5, 2,-1	0,1, -3, 8,62, -5, 2,-1
15	0,1, -2, 4,63, -3, 1, 0	0,1, -2, 4,63, -3, 1, 0	0,1, -2, 4,63, -3, 1, 0	0,1, -2, 4,63, -3, 1, 0	0,1, -2, 4,63, -3, 1, 0

TABLE 6.2: Luma interpolation filter coefficients for DCT-IF in VVC and the proposed alternative filter.

By tuning the value of D, different filters could be generated. Note that by setting D to 1, the obtained filter is almost DCT-IF.

In order to obtain the nearest possible frequency response to the wanted Gaussian filter in [5], the method described above is used with $D = 3$. By replacing p with the desired pel precision, two filters corresponding to the Gaussian function are obtained for the half and the quarter pel. We refer the proposed Gaussian filter as Kaiser Convoluted with Sinc - Gauss (KCS-G).

In a similar manner and in order to generate a filter close to the FlatTop from [5], our KCS-FlatTop (KCS-FT) filter is obtained by changing the D parameter form 3.0 to 1.8, as shown by the yellow curve in Figure 6.2 (in blue the FlatTop filter used in [5]).

On the left of Figure 6.2, we illustrate for half sample precision, the differences of the frequency responses between DCT-IF (black curve), the Gaussian filter from [5] (red curve), the proposed Gaussian filter KCS-G described above (green curve), the FlatTop filter from [5] (blue curve) and finally the KCS-FlatTop filter proposed in our work (yellow curve). On the right of the same Figure 6.2, we illustrate for quarter sample precision, the difference between the DCT-IF (black curve) and only our proposed filters (KCS-G and KCS-FT) (green and yellow curves, respectively), since in [5], no alternative filter is proposed for quarter pel precision.

Figure 6.2 emphasize the impact of D on the frequency responses of the filters. This confirms what has been mentioned in the previous section and what can be observed in Figure 6.1c regarding the relation between the value of D and the blurriness of the filter. When D increases, the filter gets blurrier. To recap, the choice of our filters results from the target of being as close as possible to the filters from [5]

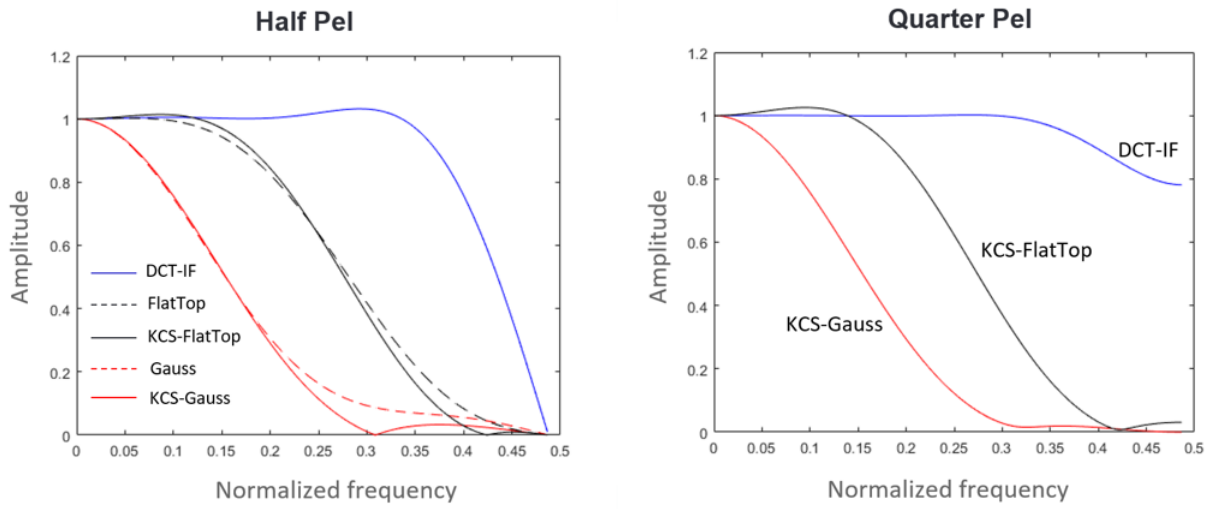


FIGURE 6.2: Frequency responses of the DCT-IF filter, the Gauss and FlatTop filters in [5] and our proposed KCS-Gauss and KCS-FlatTop counterparts for half-pel (left) and quarter-pel (right) precision.

at half pel. Then, we extended these filters to quarter pel precision. Table 6.1 summarizes the tuned parameters in formula (6.3) in order to get our different filters.

Figure 6.2 shows how the KCS-Gauss and KCS-FlatTop filters are close to the Gauss and FlatTop filters that provided gains at half-pel resolutions in [5]. The coefficients of interpolation filters for luma samples from KCS-Gauss filter and the DCT-IF can be seen in Table 6.2. From the given table, it is noticeable that only the coefficients of the quarter (frac=4 and frac=12) and half (frac=8) pel precisions differ from DCT-IF values. All the other coefficients of the other precisions are kept the same.

6.5 Preliminary filter evaluation

We evaluate our interpolation filters within the VTM11 reference H.266/VVC encoder to assess the gains they enable.

For these preliminary experiments, we assume that filter indices are delivered to the decoder on a side-band channel outside rather than within the encoded bit-stream. Equivalently, we can assume the decoder includes an *Oracle* able to predict the correct filter index for each coding unit without any need for the encoder to signal it. This Oracle decoder can be seen as an upper bound to the performance of the proposed filters since it does not entail signaling costs.

As a first experiment, we replace VVC Gaussian filter with our KCS-Gauss at half pel, and we further extend it to quarter-pel precision. For each CU, the encoder puts the Gaussian filter in competition with DCT-IF and selects the filter that minimizes the Sum of Absolute Differences (SAD) for that CU. Also, the encoder is allowed to switch between DCT-IF and KCS-Gauss only for the Y channel, whereas DCT-IF is always used for U and V channels. Finally, the same filter is always used both for horizontal and vertical filtering. For these preliminary experiments, we consider Class B JVET CTC sequences [180].

Full HD Sequence (1920x1080p)	Oracle						Signaling		
	DCTIF, KCS-G			DCTIF, KCS-G, KCS-FT			DCTIF, KCS-G, KCS-FT		
	Y	U	V	Y	U	V	Y	U	V
MarketPlace	-0,66%	-0,49%	-0,10%	-1,91%	-2,46%	-2,33%	1,50%	-2,38%	0,08%
RitualDance	-0,61%	-0,74%	-0,42%	-1,30%	-1,43%	-0,82%	1,16%	0,81%	0,80%
Cactus	-0,81%	-1,31%	-1,07%	-1,58%	-1,28%	-1,79%	0,37%	-1,12%	-1,29%
BasketballDrive	-0,98%	-1,69%	-0,89%	-1,29%	-2,26%	-1,71%	1,38%	-0,81%	0,07%
BQ-terrace	-0,71%	-0,63%	-0,73%	-1,59%	-1,28%	-1,41%	0,40%	-0,61%	-1,28%
Average	**-0.75%**	**-0,97%**	**-0,64%**	**-1,53%**	**-1,74%**	**-1,61%**	**0,96%**	**-0,82%**	**-0,32%**

TABLE 6.3: BD-rate results compared the VTM11 anchor in case of one or two alternative sub-pel interpolation filters at half plus quarter pel precision.

The coding gains are provided in Table 6.3. On the left, the decoder relies on an Oracle to predicts each CU filter index without signaling. On the right, the filter indices are signaled into the bit-stream.

Table 6.3 (left) shows that the KCS-Gauss filter enables direct BD rate reductions on the Y channel. Reductions are also obtained for U and V channels as both of them are predicted from Y. Such gains result mainly from the switching option, allowing to alternate between filters at half-pel and quarter-pel. As a second experiment, we allow the encoder to choose between the DCT-IF, the KCS-Gauss and the KCS-FlatTop filters for each CU. The rest of the experimental setup remains the same as above. We recall that the KCS-FlatTop filter cut-off frequency lies somewhere in between DCT-IF and KCS-Gauss filters. The goal of this experiment is to check the coding efficiency of allowing the encoder to choose among three interpolation filters in place of two. Table 6.3 (middle) shows that the additional KCS-FlatTop filter boosts further the coding efficiency, with Y channel gains in excess of 1.5%.
Figure 6.13 shows the distribution of the interpolation filter actually selected by the encoder for most frequent CU sizes.

As a third and last experiment, we implement a complete signaling scheme within the VTM encoder. Namely, we signal the filter indices for the three DCT-IF, KCS-Gauss and KCS-FlatTop filters at both half- and quarter-pel precision within the encoded bit-stream. Table 6.3 (right) shows that previously recorded BD-rate gains on the Y channel turn into losses when the signaling rate is kept into account.

Our experimental results prove that *i)* extending the Gaussian-based alternative filter switching method to quarter-pel using KCS-Gaussian and *ii)* extending such method to a ternary scheme including the KCS-FlatTop filter improve the encoder efficiency; however, our experiments also show that *iii)* the above gains are largely offset by the cost of signaling the filter indices. In the next section, we propose a radically different approach to filter signaling where the decoder infers the filter index for each inter CU rather than the encoder signaling it.

6.6 CNN-based Interpolation Filters Prediction

This section describes a method enabling the decoder to predict the interpolation filter index at the receiver side for inter-coded CUs without the need for the encoder to signal the filter index. We have shown that when the encoder is allowed to switch among different interpolation filters at both half- and quarter-pel, the encoder efficiency improves under the assumption that the filter index signaling cost is negligible. We cast problem of predicting at the decoder side the interpolation filter selected by the encoder as a supervised classification task. Namely, we tackle this problem using a neural network that we assume shared by the encoder and the decoder. The remainder of the section is organized as follows:

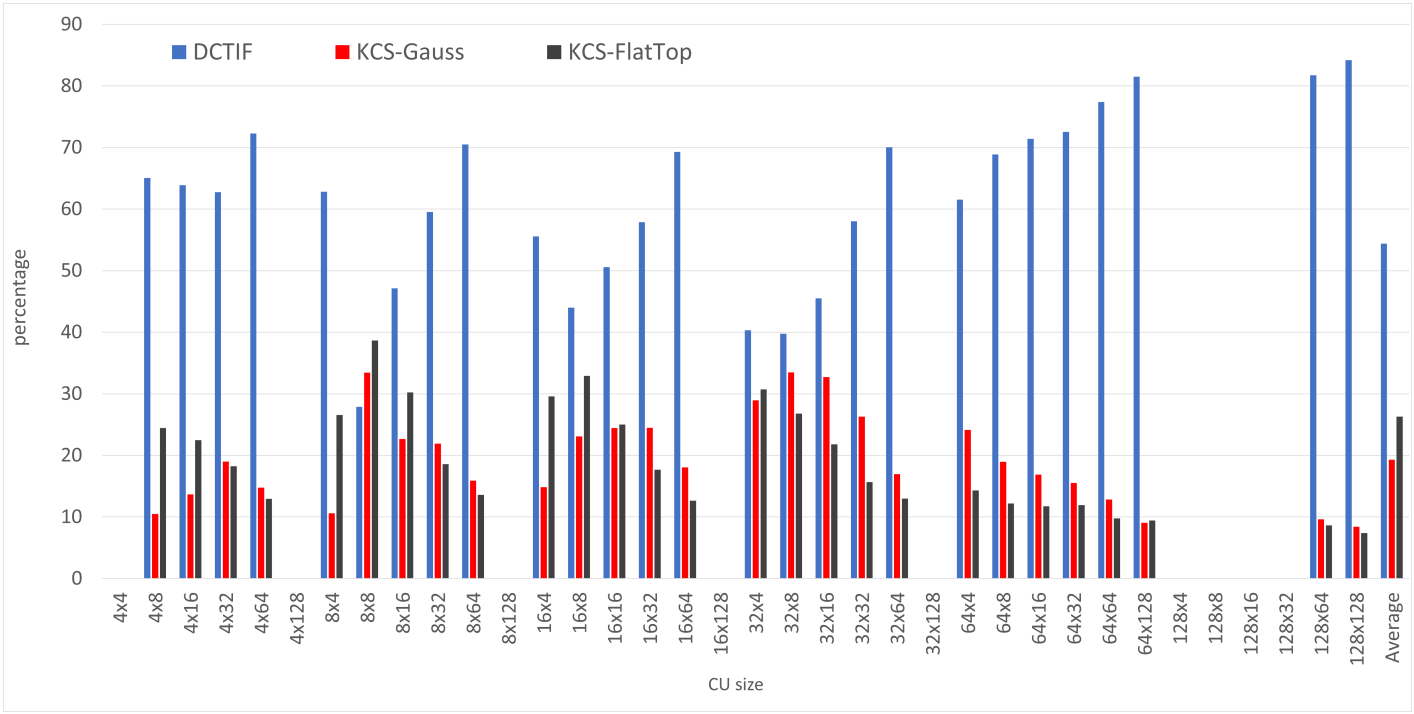


FIGURE 6.3: Distribution of the DCT-IF, KCS-Gauss and KCS-FlatTop filters selection as a function of the block size.

first, we detail the network topology, then the training procedure and we provide finally a preliminary performance assessment of the classification.

6.6.1 General Design Considerations

After some experiments, we noticed that the rate does not have significant variations between the the different interpolations but it is mainly the distortion. In a nutshell, we train a neural network to predict the distortion-optimal filter for each inter-coded CU. This network is shared by encoder and decoder so that, for any inter-coded CU, both sides predict the same filter. The resulting design is constrained by the fact that the neural network shall relies only on information available both at the encoder and the decoder. While the original uncompressed pixels of the CU to be inter-coded are the most natural candidates for training a neural network to predict the distortion-optimal interpolation filter, they are unfortunately not available at the decoder. Hence, the neural network takes as input the motion-compensated predictor(s) of the CU as it represents the best approximation available at the decoder side. Through a number of convolutional layers, the network extracts features from the motion-compensated predictor of the inter-coded CU. These features are then provided to a nonlinear classifier composed of multiple fully connected layers.

We obtain the network output by casting the interpolation filter index prediction problem as a ternary classification task, where the neural network predict the distortion-optimal interpolation filter among DCT-IF, KCS-Gaussian and KCS-FlatTop.

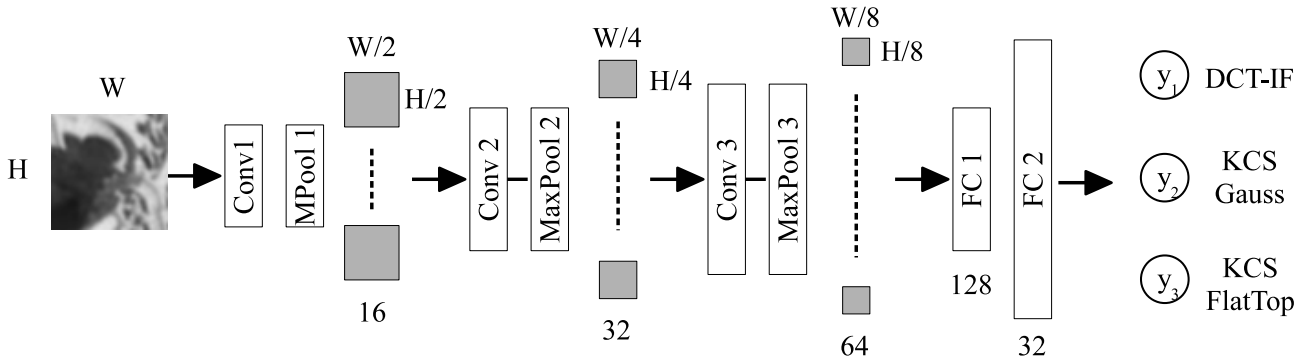


FIGURE 6.4: The convolutional architecture used to predict the optimal interpolation filter: it includes three convolutional layers and size fully connected layers for a total of about 150k learnable parameters.

6.6.2 Neural network topology

Before detailing the network topology, we list the constraints that drive its design.
First, the network shall in principle be able to deal with the different inter-coded CU sizes supported by the H266/VVC encoder, i.e., from 4×4 to 128×128. Our analysis of encoded VVC bit-streams revealed that the potential gains of our approach are distributed on the inter-coded CUs like shown in Figure 6.5. This analysis consists in measuring the distortions differences between the DCT-IF and the proposed filters, when the latter ones are selected. This case represents the source of the obtained coding gains. The total distortion reductions are distributed on different sizes of CUs. From Figure 6.5, we can detect the most promising block (CU) sizes, like 8×8, 16×8, 16×16, 32×8, 32×16, 32×32 and 64×64, that show the highest potentials.

Therefore, we focus in the following on learning the optimal interpolation filter on these CU sizes only to keep training and experimenting time to a manageable level. Nevertheless, the proposed approach can be extended to arbitrary CU sizes as well.
Second, the network must be invoked for each CU at the encoder side and for each inter-coded CU at the decoder side. That puts an upper bound to both its computational complexity and memory footprint.

We take inspiration from research in handwritten digit classification [189] proposing a network topology based on a convolutional design. The network includes 3 convolutional layers for feature extraction followed by 2 hidden fully connected layer for linear feature projection and one output layer acting as linear classifier (see Figure 6.4).

Single channel (grayscale) patches of size $W \times H$ are the input to the first convolutional layer. The layer includes 16 neurons where each neuron includes one 5×5 filter and is followed by a ReLU activation function. The convolutional layer is followed by a maxpooling layer that yields 16 feature maps sized $\frac{W}{2} \times \frac{H}{2}$. The second convolutional layer includes 32 neurons with 16 filters 3×3 and is followed by a maxpooling layer that yields in output 32 feature maps sized $\frac{W}{4} \times \frac{H}{4}$. The third convolutional layer includes 64 neurons with 32 filters 3×3 and is followed by a maxpooling layer that yields in output 32 feature maps sized $\frac{W}{8} \times \frac{H}{8}$. It turns out that this architecture can handle in principle inputs as small as 8×8 due to the included number of pooling layers.

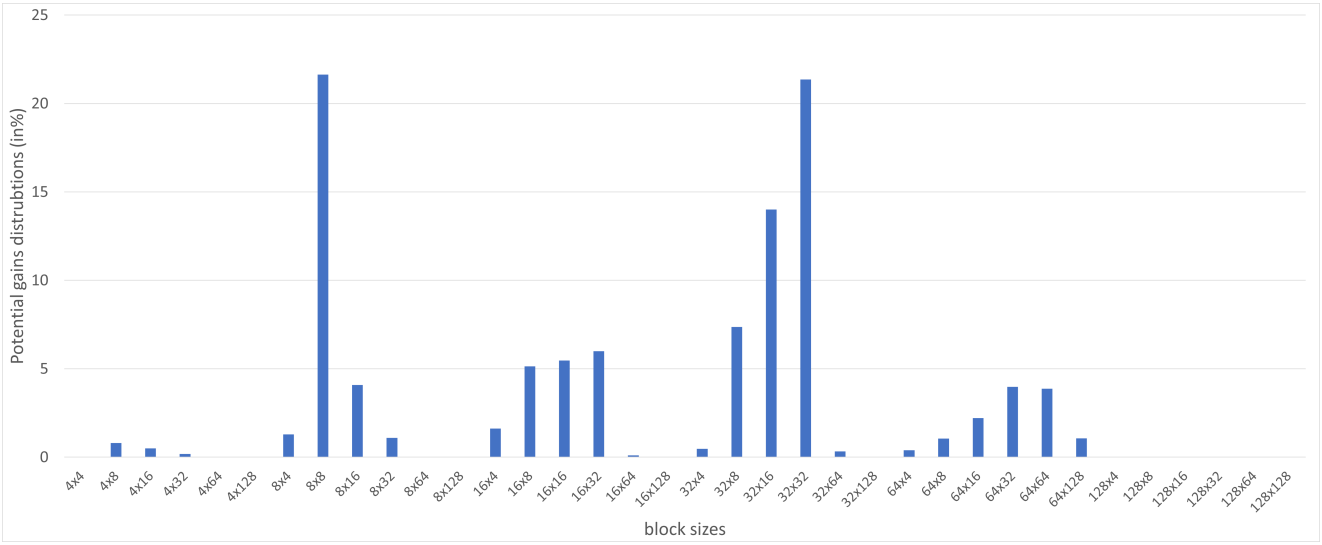


FIGURE 6.5: Source of the potential gains distributed on the different block sizes.

Next, the feature maps are vectorized and projected over a linearly separable feature space by two hidden fully connected layers, with 128 and 32 neurons, respectively. An output layer with three neurons followed by a SoftMax activation acts as a linear classifier and outputs a three-classes scores distribution corresponding to the DCT-IF, KCS-Gaussian and KCS-FlatTop interpolation filters.
Concerning the memory foot print of this architecture, the three convolutional layers include only 26k learnable parameters. The number of parameters of the first hidden fully connected layer is about equal to $\frac{W}{8}\times\frac{H}{8}\times64\times128$ and tops to 131k params for 32×32 CUs. The footprint of the two last fully connected layers is about 4k and 100 parameters, respectively. Overall, the complexity of this architecture lies above 150k learnable parameters for 32×32 CUs.
In our experiments, we also evaluated the deeper architectures VGG16 [190] and Mobilenet [36], which in other classification tasks with simlarly sized images (e.g,, CIFAR10, CIFAR100) achieve far superior accuracy with respect to our original convolutional architecture in reason of the deeper topology and thus larger receptive field. However, these architectures did not show significant advantages despite the larger footprint impacting on the encoder and decoder complexity.

6.6.3 Training set generation

To end the training of the above described network, we generate a training dataset as follows. We encode 110 Full HD and 4K sequences from the BVI-DVC dataset [35] with the VVC encoder VTM11 at QPs 10, 12, 17 and 20. Training at low QPs showed better performances than normal QPs. BVI-DVC contains 800 sequences at various spatial resolutions from 270p to 2160p, carefully selected including diverse content. This database is chosen since it offers significantly improved training effectiveness compared to other commonly used image and video training databases. Note that the cross validation technique was used to test the models. For the Y channel of each CU of each non-intra frame, we extract from the encoder the motion compensated inter predictor and the index of the interpolation filter minimizing the expected distortion as computed by the encoder RDO scheme. The predictors are dumped on a file as training patches and indexed according to CU size, while the distortion optimal filter index is saved in the form

of side annotation for use as ground truth. The result is a dataset of hundreds of thousands patches at different CU sizes with relative ground for training a neural network to predict the behavior of VVC RDO algorithm at interpolation filter switching.
Finally, for the purpose of testing, we perform the same procedure over JVET CTC class A, B and C sequences (Market Place, RitualDance, Cactus, etc.) in addition to some extra sequences not part of the JVET CTC. We clarify that there is no overlap in terms of frames or patches between the training set and the test set.

6.6.4 Training procedure

The network is trained over the dataset described above via standard error gradient back-propagation and steepest gradient descent to minimize the cross entropy between the network outputs and the respective ground truth vector. We have found that despite the limited number of parameters and our training set including hundreds of thousands of patches, the network would overfit over the training data. We addressed this issue by augmenting patches with random horizontal and vertical flips at training time. A learning rate of 0.0001 is used when updating the parameters and batches of 64 patches are required for the network to converge The entire neural network is implemented in Keras 2.4 and is trained over a GeForce RTX 2080 GPU.

6.7 Experimental Results

6.7.1 Experimental setup

At the time of starting the research on this topic, the VVC standard was under development. In order to assess the presented technique and since we were relying on several studies, analysis and statistics done on the third VTM version, SSIF was implemented in the context of the VVC Test model (VTM) 3.0. Afterwards, the method was integrated into the code of VTM-11. Obviously, the integration of SSIF into the VVC coding scheme requires normative changes. More precisely, the neural networks introduced at the level of the encoder are identically used at the level of the decoder.

All tests are carried out using the LD-P encoder configuration and sequences defined in the Common Test Conditions recommended by the JVET [180] and other sequences not defined in the CTC. Since the purpose from VVC is to support high resolution videos (full HD and beyond), we test the proposed method on full HD sequences (1920×1080p or Class B in the CTC) and 4K (3840×2160p or class A in the CTC). The simulations were conducted on an Intel Xeon Platinum 8180 machine with AVX2, turbo boost and hyperthreading, with CentOS Linux 7 and a GCC 7.3.1 compiler. We used the deep learning framework tensor-flow to train the CNNs on an Intel(R) Xeon(R) with 8 CPUs and 1 GPU running at 3.70GHz.

6.7.2 Architecture evaluation

Based on the statistics of Figure 6.5, the choice of the block sizes to exploit is made. Mainly, these CUs are the major source of the obtained gains. We train the described architecture of neural networks on these promising blocks. On small blocks (8×8, 16×8, 16×16), the designed neural network encounter some difficulties to reach good predictions accuracy. For the other blocks, the classification performance is summarized in Table 6.4. We can notice that the prediction accuracy increases with the size of the blocks.

Block size	Architecture	Accuracy
8x8	Simple CNN (3 Conv layers + 2 FC)	50%
8x16	Simple CNN (3 Conv layers + 2 FC)	56%
16x8	Simple CNN (3 Conv layers + 2 FC)	57%
16x16	Simple CNN (3 Conv layers + 2 FC)	68%
16x32	Simple CNN (3 Conv layers + 2 FC)	74%
32x4	Simple CNN (3 Conv layers + 2 FC)	67%
32x8	Simple CNN (3 Conv layers + 2 FC)	74%
32x16	Simple CNN (3 Conv layers + 2 FC)	74%
32x32	Simple CNN (3 Conv layers + 2 FC)	76%
	VGG16 (16 Conv layers + 2 FC)	76%
	MobileNet (28 Conv layers + 2 FC)	79%
64x8	Simple CNN (5 layers)	82%
64x64	Simple CNN (5 layers)	82%

TABLE 6.4: Performance of the different Neural Networks classifying the three filters.

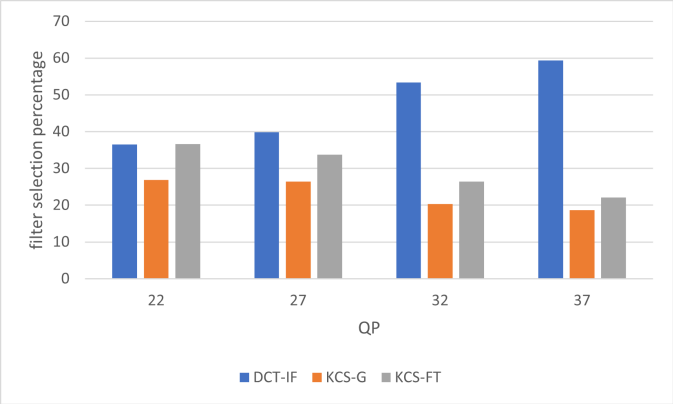


FIGURE 6.6: Interpolation filter distribution as a function of the QP.

Clearly, the block size plays an important role in the selection of the filters put into competition in the interpolation process. In other words, the intervention of the proposed filters occurs depending on the block size of the current CU. Four block sizes, presenting clear high peaks in the histogram, are chosen finally. These blocks are 32×8, 32×16, 32×32 and 16×32. The choice of the blocks is made based on a compromise between the highest accuracy from Table 6.4, the highest potential gains provided by these blocks in Figure 6.5 and a reasonable complexity added to the encoder. For the remaining blocks, only DCT-IF is used, thus the proposed method does not need to be applied.

Figure 6.7 illustrates the areas where the neural networks succeed to make the right filter prediction and the other blocks where bad predictions occur. In this figure, a frame from the original cactus sequence is presented with its partitioning colored with a gradient map. Green color, which corresponds to the 0 value in the map, indicates that the right prediction of the filter is made by the neural networks, hence no waste of the potential gain guaranteed in the oracle mode. Then, colors going from green to blue (left part of the map) show the quantity of loss caused by a bad prediction of the proposed filters, whether KCS-Gauss or KCS-FlatTop, instead of DCT-IF, or a KCS predicted filter instead of the other. The more the color tends to blue, the more the loss is bigger. Similarly for the colors going from green to red (right part of the map), showing the other way round, where a bad prediction of DCT-IF instead of one of the two proposed KCS-filters.

Resolution	Sequence	Y	U	V
4K (3840x2160p)	Tango2	-0,68%	-0,21%	-0,59%
	FoodMarket	-0,77%	-0,64%	-1,09%
	Campfire	-0,40%	-0,45%	-0,27%
	CatRobot	-0,59%	-0,20%	-0,63%
	DaylightRoad2	-0,37%	-0,33%	-0,02%
	ParkRunnig3	-0,62%	-0,39%	-0,25%
	TrafficFlow	-0,12%	-0,27%	-0,35%
	Drums	-0,41%	-0,02%	-0,32%
Average		**-0,50%**	**-0,31%**	**-0,44%**
Full HD (1920x1080p)	MarketPlace	-0,17%	-0,52%	-0,44%
	RitualDance	-0,56%	0,21%	-0,22%
	Cactus	-0,41%	-0,20%	-0,53%
	BasketballDrive	-0,69%	0,76%	-0,44%
	BQTerrace	-0,09%	-0,08%	0,50%
	Kimono	-1,06%	-1,08%	-0,40%
Average		**-0,50%**	**-0,15%**	**-0,26%**
WVGA (832x480p)	BasketballDrill	-0,52%	0,03%	-0,94%
	BQMall	-0,24%	-0,41%	0,07%
	PartyScene	-0,11%	-0,20%	-0,38%
	RaceHorses	-0,56%	0,48%	0,16%
Average		**-0,36%**	**-0,03%**	**-0,27%**
All		**-0,47%**	**-0,20%**	**-0,34%**

TABLE 6.5: Coding performance for CTC QP 22-27-32-37.

6.7.3 Performance evaluation

The proposed method achieves on average -0,47%, -0,20% and -0,34% BD-rate reduction for the CTC QP range: 22-27-32-37. The obtained results are summarized in Table 6.5.

Although the proposed technique presents a consistent performance on the sequences, yet on BQ-terrace, we can notice tiny gains. This can be justified in 6.8 by the fact that the majority of the gains on this sequence come from small blocks (8×8, 8×16, 16×8, etc.), on which our CNNs are not applied. The blocks, on which our method intervene, can generate only 18% of the total gains.

In conclusion, the main purpose of this work is to show the strong need of adapting the interpolation filter to the content, at a fine granularity like the block level. Finally, the presented experimental results prove the efficiency of switching from the concept of interpolating with a fix filter to dynamically alternating between two or more filters without signaling by using artificial neural networks to predict the best filter.

6.7.4 Statistical analysis

In the beginning of this work, the idea was to integrate the concept of alternating between DCT-IF and KCS-G into VTM-3. As a sanity check, we compare our filter index predictor based on a CNN with a much simpler predictor based on statistical features that are available both at the encoder and the decoder.

Several logical simple criterion were tempted. These criteria, for instance the motion in the sequence, the texture complexity, the QP value and the block size, prove somehow some impact on the filter choice. Therefore a deep analysis concerning these features is detailed in the following.

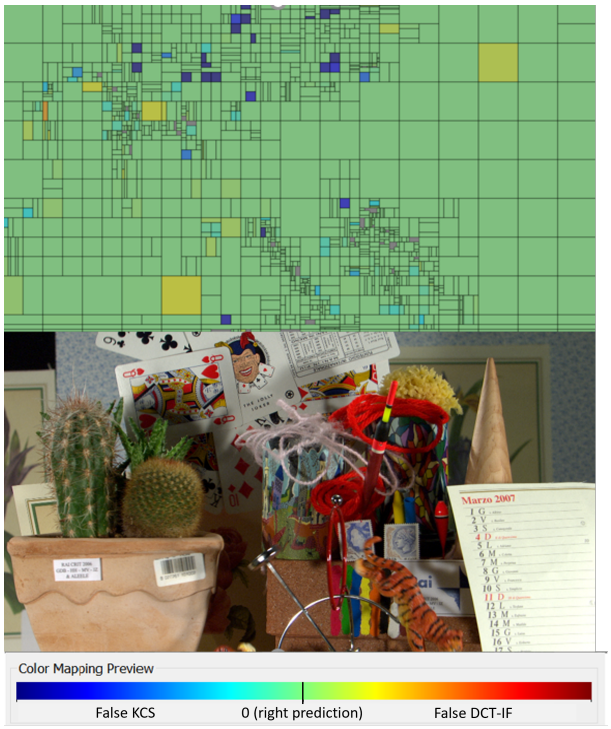


FIGURE 6.7: Illustration of the bad and the right predictions of the interpolation filters on cactus sequence.

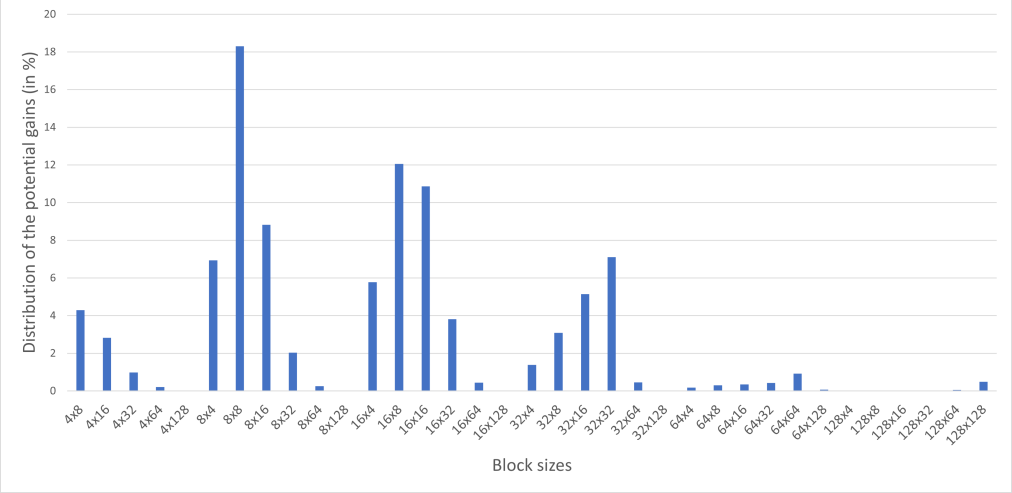


FIGURE 6.8: Source of the potential gains distributed on the different block sizes on BQ-terrace.

FIGURE 6.9: An example of high motion in a video sequence BasketballDrive.

6.7.4.1 The Motion

After the calculated of the motion vector of each block, motion compensation and interpolation filtering are performed to obtain the prediction value of each block. According to the precision or the phase of the MV, the interpolation filter is selected among the 16 filters from Table 6.2. Therefore, a first tempted track can be to find a relation between the motion and the filter chosen by the encoder. The idea is to differentiate the filters according to the motion speed for instance. We assume that for slow motions, DCT-IF (the sharp filter) should be used while for high-speed motions, the blurry filter (Gaussian for example) would be rather preferred.

One can believe that the inter prediction must be blurrier when there are large motions and sharper when there are slow motions. This assumption seems like a logical choice since the camera cannot catch the motion of high-speed objects in a given exposure time. It is nonetheless beneficial to do so because of the physical characteristics of the capture device that makes it blur out moving objects.

Furthermore, let us take a video sequence with high motion, BasketballDrive for instance, seen in Figure 6.9. It is obvious that the hands of the players move in high speed, so the camera cannot correctly catch all the details. On the this sequence, which presents the highest motion among the considered sequences (see Figure 6.10), we can notice the highest coding gains. One can think that there is a correlation between the motion and the need of this new kind of filters. In other words, this assumption means that the increase in the gains provided by the KCS-G filter is justified by the motion increase. Nevertheless, based on the analysis of Figure 6.10, it is true that in some sequences, a correlation exists between the motion and the obtained BD-rate gains, but this is not always the case. Figure 6.10 shows the augmentation of the gains with the increase of the MV average in the considered sequences except on RitualDance. It should be noted that the presented BD-rate gains are in the oracle mode in VTM-3. In this version of VTM, the alternative half pel filter was not yet present. Therefore, presenting such proposal brings significant gains to VTM.

What can be concluded is that the motion has more or less some impact on the filter choice, but it is not a sufficient criterion to always discriminate the filters.

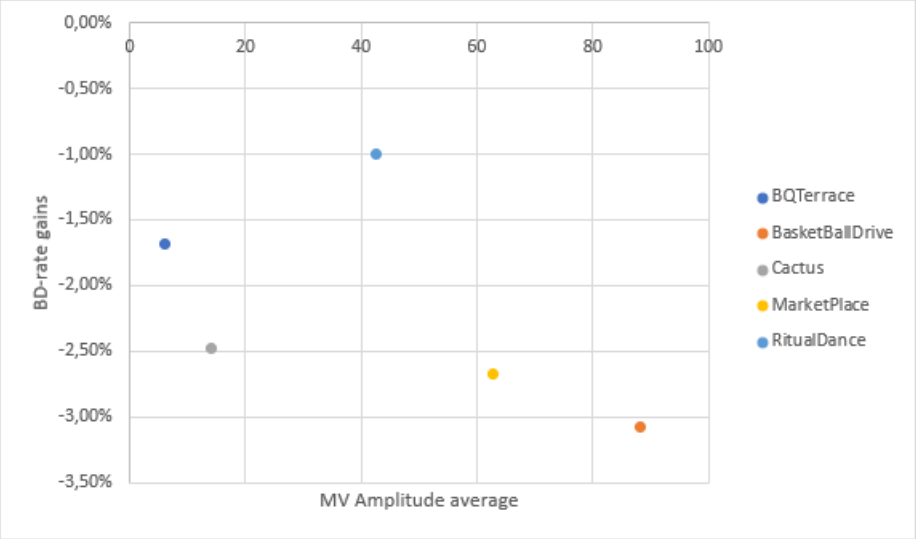


FIGURE 6.10: The relation between the average of the motion vector amplitude and the BD-rate gains obtained.

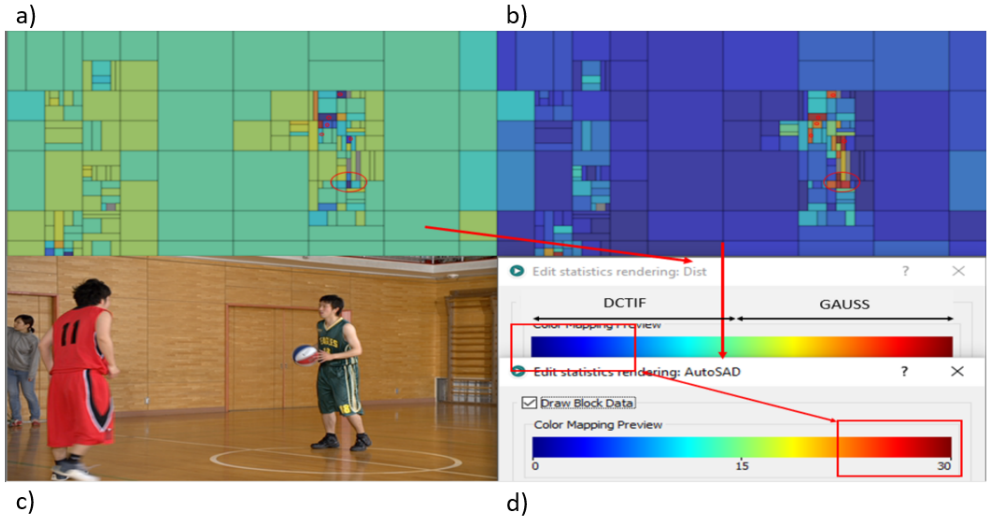


FIGURE 6.11: a) Partitioned BasketballDrive frame colored with the first gradient map in d) according to the distortion value. b) Partitioned BasketballDrive frame colored with the second gradient map in d) according to the *AutoSAD* value. c) Original BasketballDrive frame. d) Gradient color maps.

6.7.4.2 The AutoSAD

Since the motion, which is a temporal feature, does not show a solid correlation with the filter selection, the exploration switched to a spatial feature, where the texture complexity is considered. We call this feature *"AutoSAD"*. This metric is a way to represent the texture complexity by taking the average difference of the luma pixel value between the considered block and its right shifted one (shift of 1 pixel to the right) and with the down shifted (shift of 1 pixel to the bottom).

To emphasize the relation between the *"AutoSAD"* and the selected filter, the study presented in Figure 6.11 is carried out. In Figure 6.11-a) the distorsion color map is presented. Colors going from green to blue correspond to the blocks where DCT-IF is the selected filter, and the more the color goes towards blue, the more the difference between the distortions of the filters is big which implies the increasing need for DCT-IF. Colors going from green to red correspond to the cases where Gaussian filter is selected. Similarly, the more the color tends to red, the more the difference is big. For b), the color map corresponds to the *AutoSAD*. We can notice that flat regions, where there is no high texture complexity like the wall, the color tends to blue, which corresponds to small *AutoSAD* values. Whereas for the regions with more texture complexity like the players for example, we can clearly see that the *AutoSAD* values increase. The purpose from presenting the *AutoSAD* with the distortion difference between the two filters is to show that there is somehow a correlation between them. Blocks surrounded by red circles present a common attitude. The *AutoSAD* of these blocks, colored in red in b), is high and these same blocks are colored in blue in a) indicating that DCT-IF filter is strongly needed for these blocks since this filter can produce a very small distortion compared to the one obtained with the other filter.

The *AutoSAD* helps in a lot of cases to determine the desired filter. Although, it is not able to distinguish the filters for all the blocks.

6.7.4.3 The QP value

During the experiments, it has been noticed that the selection of the filters is affected by the QP value. With the increase of the QP, the selection of the DCT-IF increases and the percentage of selecting the Gaussian decreases as seen in Figure 6.12.

6.7.4.4 The block sizes

Another observation is related to the dependence of the filter selection on the block sizes. As can be seen in Figure 6.13, the block size can be a parameter impacting the determination of the filter. In other words, based on the block size, the filters proposed as options to interpolate are determined.

6.7.4.5 Interpretation of the conducted analysis

Since the listed features seem to have more or less some impact on the determination of the filter, the correlation between these input features and the selected filter is measured. Although these features provide some hints for the determination of the filter, but as mentioned before and shown in Figure 6.14, no feature is discriminatory enough. Knowing that the correlation takes values ranging between two extreme values +1 and -1, corresponding to the ideal correlation, this explains the 'ones' values on the diagonal showing the perfect relationship between the feature and itself. The texture complexity, where luminance pixel values are considered, shows the highest correlation with the filter selection as seen in Figure 6.14.

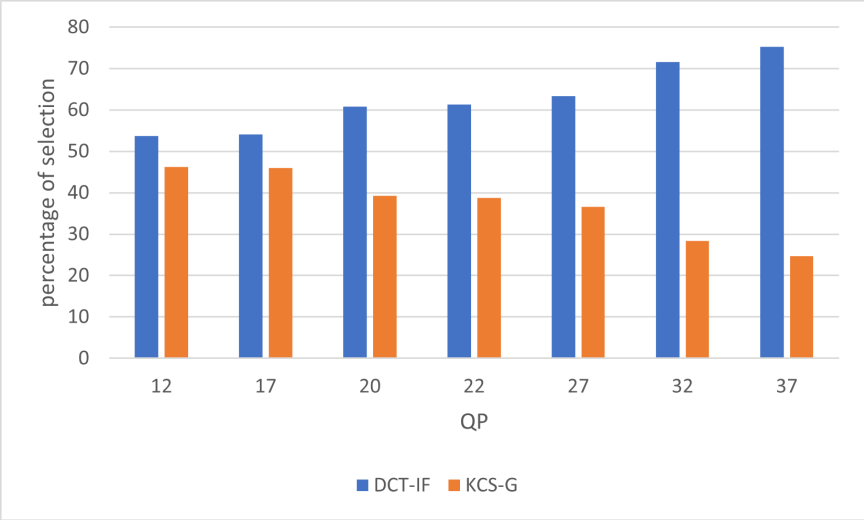


FIGURE 6.12: Distribution of filter selection according to QP value.

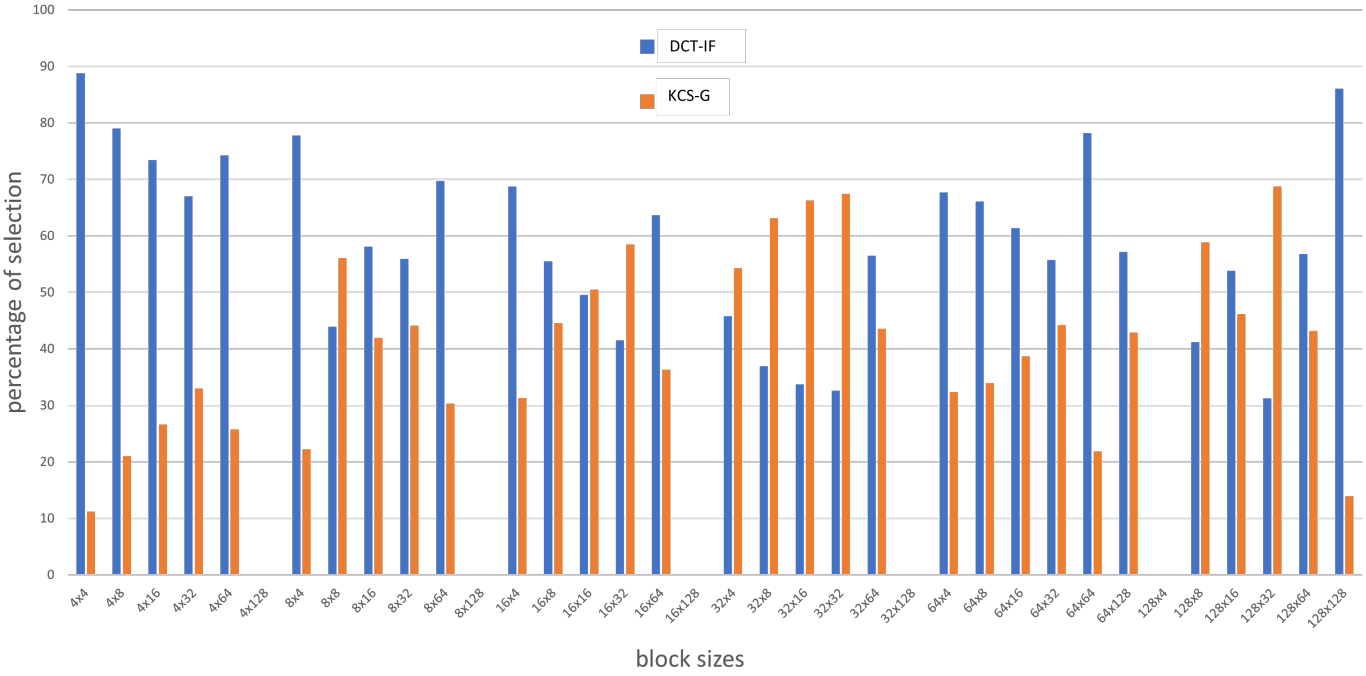


FIGURE 6.13: The block sizes play an important role in the selection of the filters put into competition.

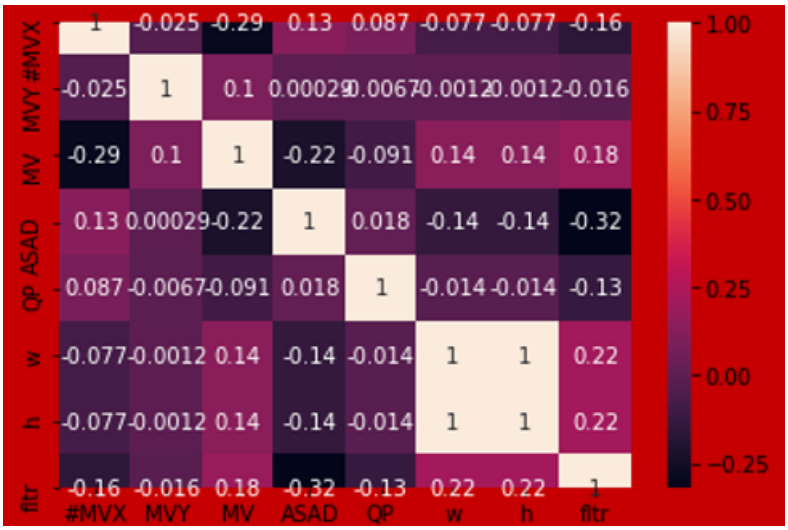


FIGURE 6.14: Measure of the correlation between different criteria and the filter choice.

Block size	Architecture	Accuracy
32x32	Modified Lenet5	87 %
32x16	Modified Lenet5	87 %
16x32	Modified Lenet5	81 %
16x16	Modified Lenet5	82 %
8x8	MLP (2 hidden layers)	77 %

TABLE 6.6: Performance of the different Neural Networks for predicting DCT-IF or KCS-G.

All these features, combined together, are given as input to a Multi-Layer Perceptron (MLP) network [191]. The target is to check whether a simple neural network can replace the CNNs employed and described in the previous section. The accuracy of predicting the right filter index using this simple MLP of 2 fully connected layers is 70%. Note that increasing the number of layers did not bring better results. For this reason, an investigation was launched in order to find other features that can be more decisive for the filter choice. The resolution of the frame (w and h in Figure 6.14) are good features compared to the other features, yet the new prediction accuracy does not exceed 73%. Comparing these results to the ones obtained in Table 6.6 with the simple convolutional network for this binary classification problem (2 filters), one can deduce the advantage of such neural networks.

Conducting such sanity check test bolsters the choice of deep learning approach in order to solve such problem.

6.8 Future works

Several propositions and future works are suggested in order to invest and benefit from the potential of this new approach, such as extending from three interpolation filters to more and separating between the vertical and horizontal interpolations.

Actually, in VVC reference software, the interpolation process is done in both horizontal and vertical directions.

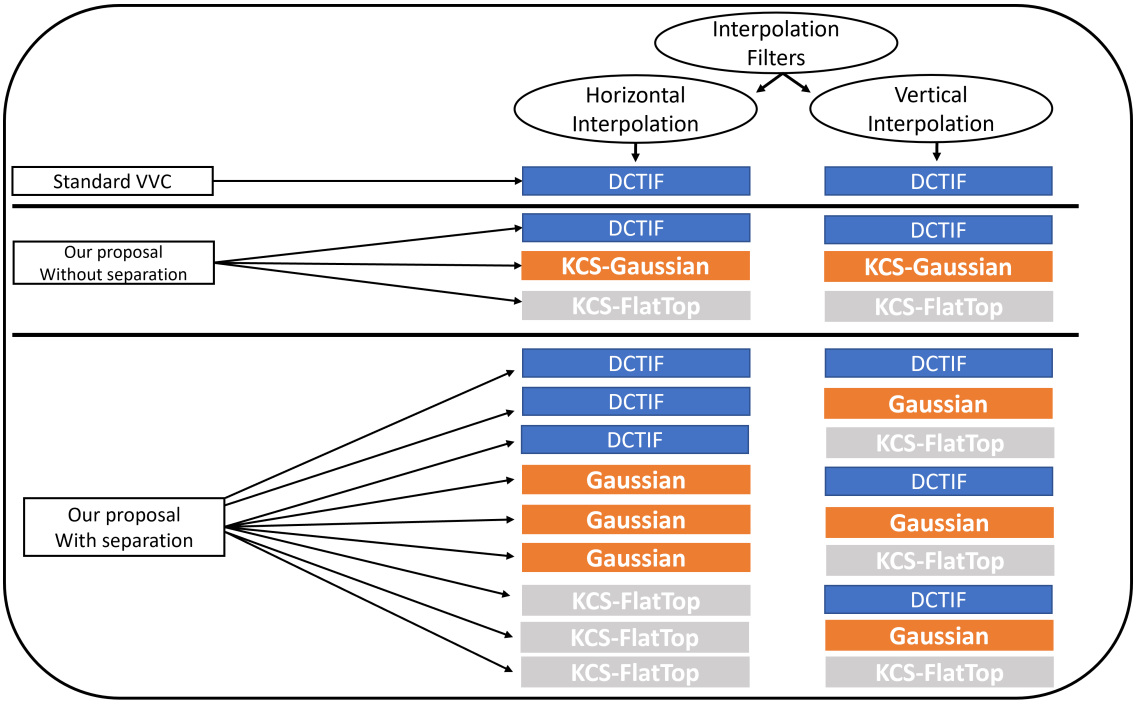


FIGURE 6.15: The different options introduced to the standard interpolation process with and without separation between horizontal and vertical interpolations.

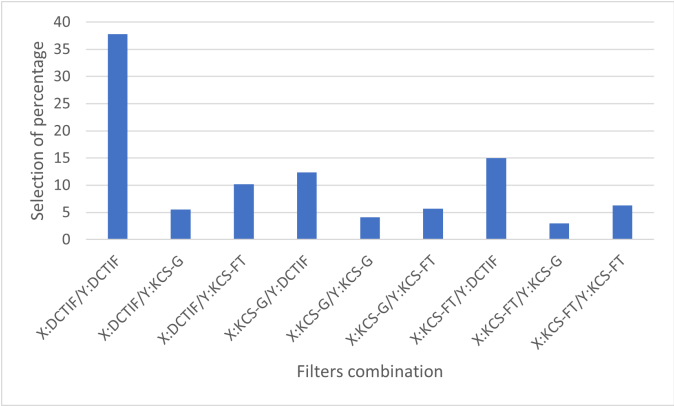


FIGURE 6.16: Percentage of selection for each of the 9 possible filter combinations.

Given that in VVC reference model one fixed DCT-interpolation filter is proposed, the horizontal and vertical interpolations were done using this same filter. However, in our proposal, the interpolation process can benefit from the presence of three or more different filters. The separation creates different combinations of filters, hence the horizontal and vertical interpolations can be executed with different filters when needed. Consequently, the interpolation process gains more flexibility and becomes more adapted to each of the two directions. Therefore, the inter prediction gets more efficient because there will be less residual information and less data to encode and transmit. After implementing and testing the oracle mode of this new potential approach described in Figure 6.15, experimental results show that the potential BD-rate gains increased. For example, on BQ-terrace, the BD-rate reduction augments from -1,59%, -1,28% and -1,41% to -2,78%, -1,58% and -2,57%, for Y,U and V respectively. This expansion in the dynamicity of the interpolation process is the source of the increasing in the gains. Nonetheless, increasing the number of combinations implicates an increasing in the signalization cost. As a result, a big part of the potential gain will be lost, as explained before. The aim in a future work is to build deep neural networks able to predict these filters combinations to recover the most possible of the potential gains.

Figure 6.15 clarifies the difference between the standard VVC and our approaches. In VVC, only one option to interpolate is available. In the proposed method, three options for interpolating are possible. With the introduction of the idea of separation, the possibilities becomes nine. In the oracle mode of this potential novel proposal, the new percentages of filters combinations selection are presented in Figure 6.16, showing a promising performance. Note that the first combination, where DCT-IF is the selected filter for both horizontal (X) and vertical (Y) interpolations, is only selected 37% of the cases. The proposed combinations take rest.

In the actual implementation, we restrict our method from being applied in Affine mode. In this mode, recent filters, different than DCT-IF, were designed, This necessitates from our side to propose other alternative filters to the one in the standard. This is another future track.

Chapter 7

Optimization of the new Adopted VVC Tools

To achieve VVC's final performance, many novel coding tools are used to yield overall bit-rate savings. However, the high number of adopted tools generated a high level of complexity which creates an issue to be considered. This chapter of the thesis begins with a deep study done on the different tools adopted in VVC. It reports a compression performance analysis, based on the coding gain evaluation of each tool for various contents. The analysis also considers the impact of the tools in terms of encoding complexity. Global performance measures are provided in different encoding configurations and picture formats. A variable performance of the tools across the sequences is noticed. Some tools provide significant coding gains on some sequences but not on others, which stimulates the concept of adapting the activation of the tools to the type of the content in a way that guarantees an equilibrium between the imposed complexity and the provided coding efficiency.

The difference between the tools described in this section and the ones presented in Section 3.2.2 is that the latter represent the core of VVC and do not have the option to be turned off. Given that the tools presented in this section include this option, we are aiming to optimize their activation in a way that reduces the complexity added by these tools while maintaining a certain level of compression efficiency.

In order to determine the optimized activation of the switchable VVC tools, it was required, as a start, to conduct a study on their performance. In the first section of this chapter, the study of these switchable VVC tools is explained in detail as stated in VTM7. The second chapter describe briefly VVenC, an optimized VVC software, that is used in the implementation of our approach. In the third section, we propose an optimization framework to determine an efficient technique of activating the new coding tools introduced in VVC. In other words, the role of this proposed framework is to find the best possible compromise between the provided compression efficiency and the complexity imposed by the VVC tools.

7.1 VVC Tools Description

Following the submission of a significant number of coding resources to the CfP, it was decided to start with a "clean slate" approach. This first version only included an advanced quadtree with multitype tree (QT+MTT) block partitioning, which was identified as a common feature among nearly all proposals, and its introduction would explicitly affect the design of all other block-based coding tools. On top of that, CfP responses, as well as new ones, were thoroughly investigated in "core experiments" for coding efficiency and implementation complexity. When a fair trade-off between coding efficiency and complexity was identified in general, additional techniques were added to the VVC architecture.

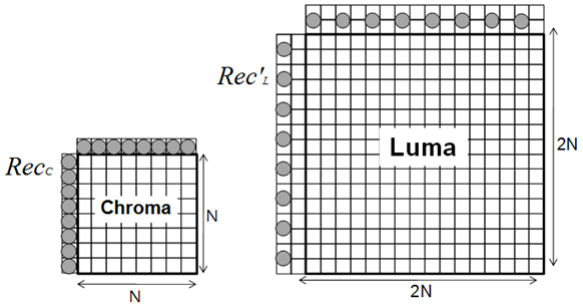


FIGURE 7.1: Locations of the samples used for the derivation of α and β [6].

VVC applies the classic block-based hybrid video coding architecture known from its predecessors. Although the same framework is applied, novel tools are included in each basic building block to further improve the compression.

7.1.1 Intra Prediction

7.1.1.1 Cross-Component Linear Model prediction (CCLM)

To reduce the prediction redundancy between Luma and Chroma components (cross-component redundancy), a cross-component linear model (CCLM) prediction mode has been introduced to VVC. By doing so, signaling for the Chroma components is further reduced. This tool consists in predicting the Chroma samples from the reconstructed luma samples of the same CU with a linear model:

$$\text{pred}_{C(i,j)} = \alpha.\text{rec}'_{L(i,j)} + \beta \tag{7.1}$$

with $\text{pred}_{C(i,j)}$ being the prediction for the CU Chroma component at pixel position (i, j) and $\text{rec}_{L(i,j)}$ the reconstructed luma samples of the same CU. The linear model parameter α and β are directly computed using 4 pixels values: the minimum and maximum Luma values $L_{\min}$ and $L_{\max}$ of the current block reference samples and their collocated Chroma values $C_{L_{\min}}$ and $C_{L_{\max}}$: Chroma values at the same positions than $L_{\min}$ and $L_{\max}$. Using the following formulas:

$$\alpha = \frac{C_{L_{\max}} - C_{L_{\min}}}{L_{\max} - L_{\min}} \quad \beta = C_{L_{\min}} - \alpha L_{\min} \tag{7.2}$$

Since α and β are computed using reference samples, they do not need to be signaled at the decoder side. They can be quickly computed. For a coding block with a square shape, the above two equations are applied directly. For a non-square coding block, the neighboring samples of the longer boundary are first subsampled to have the same number of samples as for the shorter boundary. Figure 7.1 shows the location of the left and above samples and the sample of the current block involved in the CCLM mode. Note that, since the Luma block is twice the sizes of one Chroma block, the used reference samples for the Luma is also two times bigger but down sampled to match the Chroma reference samples. Table 7.1 summarizes the general performance of CCLM in VTM7. This performance is measured by deactivating this tool from a full VVC configuration to see the impact provided by CCLM.

CCLM has been adopted in VVC with JVET-K0190 [192]. Several simplifications and improvements have been adopted. Different modes and different combinations are still under study, great chance of

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	10,83%	104%	7,44%	99%		
	Food Market	4,83%	103%	3,21%	99%		
	Campfire	23,46%	104%	14,19%	98%		
A2	Cat Robot	6,00%	100%	3,53%	99%		
	Daylight Road	3,49%	101%	2,60%	99%		
	Park Running	2,11%	96%	0,85%	98%		
B	Market Place	5,80%	101%	5,84%	99%	1,41%	100%
	Ritual Dance	4,95%	101%	3,34%	99%	0,92%	100%
	Cactus	3,74%	100%	3,53%	99%	1,82%	100%
	Basketball Drive	2,74%	100%	1,86%	99 %	0,61%	99%
	BQ-Terrace	2,07%	99%	2,67%	99%	0,69%	100%
C	Basketball Drill	8,09%	100%	3,66%	99%	1,77%	101%
	BQ-Mall	3,04%	100%	2,22%	99%	0,51%	100%
	Party Scene	4,28%	99%	2,99%	99%	0,76%	100%
	Race Horses	2,16%	98%	1,52%	99%	0,68%	99%
Overall	BDR-Y	1,90%		1,02%		0,06%	
	BDR-U	17,00%		11,85%		3,67%	
	BDR-V	18,28%		13,71%		4,14%	
Average	BDR-YUV	**5,83%**	**100%**	**3,96%**	**99%**	**1,02%**	**100%**

TABLE 7.1: Simulation results of disactivating CCLM VTM tool. (VTM 7 anchor)

evolution in future meetings Reference documents: JVET-K0190: Tests of cross-component linear model JVET-L0085/L0136: Line buffer reduction for LM Chroma JVET-L0191: Cross-component linear model simplification (with min and max values) JVET-L0338/JVET-L0340: Multi-Directional LM (MDLM) JVET-M0064: CCLM table reduction and bit range control. JVET-M0142: Modified CCLM down sampling.

7.1.1.2 Position Dependent intra Prediction Combination (PDPC)

Position Dependent Prediction Combination is an efficient tool used to improve the compression performance of several conventional intra prediction modes. This tool is more specifically to the following intra modes without signalling: planar, DC, intra angles less than or equal to horizontal, and intra angles greater than or equal to vertical and less than or equal to 80. If the current block is Bdpcm mode or MRL index is larger than 0, PDPC is not applied.

The prediction sample $\text{pred}(x, y)$ located at position (x, y) is predicted using a linear combination of reference samples according to the following expression:

$$\text{pred}(x', y') = \text{Clip}\Big(0, (1 << \text{BitDepth}) - 1, \\ \big(w_L \times R_{-1,y'} + w_T \times R_{x',-1} + (64 - w_L - w_T) \times \text{pred}(x', y') + 32\big) >> 6\Big) \quad (7.3)$$

where $R_{x,-1}, R_{-1,y}$ represent the reference samples located at the top and left boundaries of current sample (x, y), respectively.

If PDPC is applied to DC, planar, horizontal, and vertical intra modes, additional boundary filters are not needed, as required in the case of HEVC DC mode boundary filter or horizontal/vertical mode edge filters. PDPC process for DC and Planar modes is identical. For angular modes, if the current

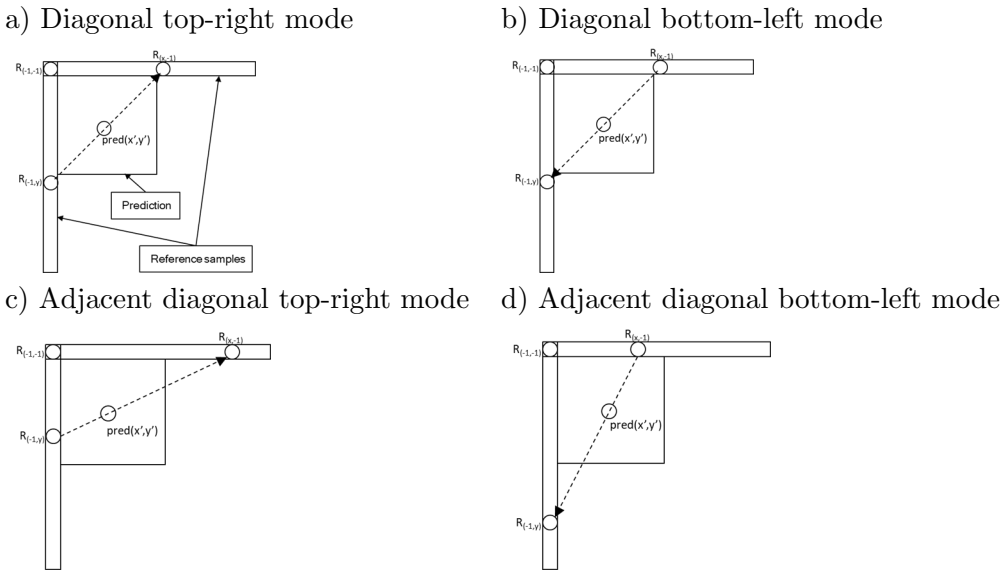


FIGURE 7.2: Definition of samples used by PDPC applied to diagonal and adjacent angular intra modes.

angular mode is HOR_IDX or VER_IDX, left or top reference samples is not used, respectively. The PDPC weights and scale factors are dependent on prediction modes and the block sizes. PDPC is applied to the block with both width and height greater than or equal to 4. Figure 7.2 illustrates the definition of reference samples ($R_{x,-1}$ and $R_{-1,y}$) for PDPC applied over various prediction modes. The prediction sample pred(x', y') is located at (x', y') within the prediction block. As an example, the coordinate x of the reference sample $R_{x,-1}$ is given by: $x = x' + y' + 1$, and the coordinate y of the reference sample $R_{-1,y}$ is similarly given by: $y = x' + y' + 1$ for the diagonal modes. For the other angular mode, the reference samples $R_{x,-1}$ and $R_{-1,y}$ could be located in fractional sample position. In this case, the sample value of the nearest integer sample location is used.

PDPC was adopted with JVET-K0063. Tests were conducted in VTM7.0 to evaluate the performance of PDPC.

7.1.1.3 Multiple Reference Line (MRL) intra prediction

The concept of Multiple Reference line (MRL) intra prediction is to use more reference lines for intra prediction. In Figure 7.3, an example of 4 reference lines is depicted, where the samples of segments A and F are not fetched from reconstructed neighbouring samples but padded with the closest samples from Segment B and E, respectively. HEVC intra-picture prediction uses the nearest reference line (i.e., reference line 0). In MRL, 2 additional lines (reference line 1 and reference line 3) are used.

The index of selected reference line (mrl_idx) is signaled and used to generate intra predictor. For reference line idx, which is greater than 0, only include additional reference line modes in MPM list and only signal mpm index without remaining mode. The reference line index is signaled before intra prediction modes, and Planar and DC modes are excluded from intra prediction modes in case a nonzero reference line index is signaled. MRL is disabled for the first line of blocks inside a CTU to prevent using extended reference samples outside the current CTU line. Among all the defined tests concerning MRL

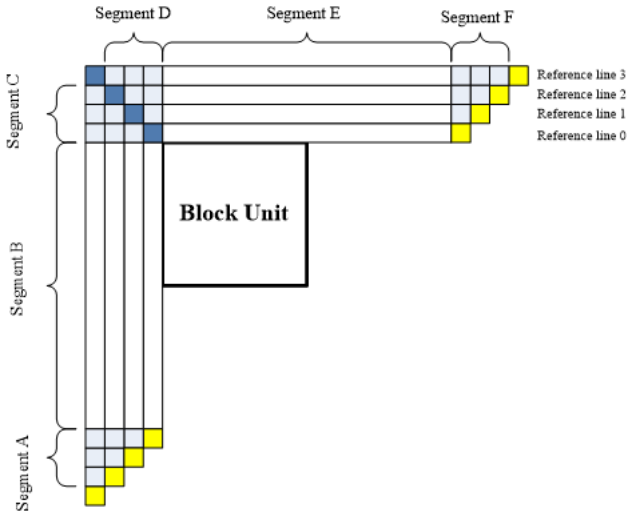


FIGURE 7.3: Example of four reference lines neighboring to a prediction block [7].

based on explicit signaling, it was concluded that, applying MRL to MPM provides the best trade-off between coding efficiency and encoder runtime. Table 7.2 represents in details the performance of MRL.

Reference documents: JVET-O0426 [193] and JVET-P0418 [194].

7.1.1.4 Intra Sub-Partitions (ISP)

The Intra Sub-Partitions (ISP) tool divides luma intra-predicted blocks vertically or horizontally into 2 or 4 sub-partitions depending on the block size. Minimum block size for ISP is 4×8 (or 8×4), but if block size is greater than this minimum, then the corresponding block is divided by 4 sub-partitions.

It has been noted that the M×128 (with $M \leq 64$) and 128×N (with $N \leq 64$) ISP blocks could generate a potential issue with the 64×64 VDPU. For example, an M×128 CU in the single tree case has an M×128 luma TB and two corresponding $M/2$×64 chroma TBs. If the CU uses ISP, then the luma TB will be divided into four M×32 TBs (only the horizontal split is possible), each of them smaller than a 64×64 block. However, in the current design of ISP chroma blocks are not divided. Therefore, both chroma components have a size greater than a 32×32 block. Analogously, a similar situation could be created with a 128×N CU using ISP. Hence, these two cases are an issue for the 64×64 decoder pipeline. For this reason, the CU sizes that can use ISP is restricted to a maximum of 64×64. Figure 7.4 shows examples of the two possibilities. All sub-partitions fulfill the condition of having at least 16 samples.

In the ISP, the dependence of the 1×N/2×N sub-block prediction on the reconstructed values of the previously decoded 1×N/2×N sub-blocks of the coding block is not permitted so that the minimum prediction width for sub-blocks becomes four samples. For example, an 8×N coding block ($N > 4$), that is coded using an ISP with a vertical split, is divided into two prediction regions each of 4×N size and four 2×N -sized transforms. In addition, a 4×N coding block that is coded using ISP with vertical slot is provided using the full 4×N block; four transform each of 1xN is used. Although the transform sizes of 1×N and 2×N are permitted, it is stated that the transform of these blocks in 4xN regions can be done in parallel. For example, when a 4xN prediction region contains four 1xN transforms, there is no transform in the horizontal direction; the transform in the vertical direction can be performed as a

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	-0,04%	98%	0,00%	100%		
	Food Market	-0,04%	97%	-0,01%	100%		
	Campfire	0,10%	98%	0,08%	100%		
A2	Cat Robot	0,16%	98%	0,10%	100%		
	Daylight Road	0,20%	98%	0,19%	100%		
	Park Running	0,03%	99%	0,01%	100%		
B	Market Place	0,01%	98%	0,04%	100%	0,02%	100%
	Ritual Dance	-0,01%	98%	-0,01%	99%	-0,05%	99%
	Cactus	0,24%	99%	0,07%	100%	-0,04%	99%
	Basketball Drive	0,31%	99%	0,21%	100 %	-0,02%	99%
	BQ-Terrace	0,97%	100%	0,51%	100%	0,05%	100%
C	Basketball Drill	1,73%	101%	0,69%	100%	0,29%	99%
	BQ-Mall	0,09%	97%	0,04%	100%	-0,12%	103%
	Party Scene	0,11%	99%	0,06%	100%	-0,08%	99%
	Race Horses	0,21%	99%	0,11%	100%	-0,02%	99%
Overall	BDR-Y	0,32%		0,16%		0,05%	
	BDR-U	0,11%		0,07%		-0,36%	
	BDR-V	0,13%		0,10%		0,07%	
Average	BDR-YUV	**0,27%**	**99%**	**0,14%**	**100%**	**0,26%**	**100%**

TABLE 7.2: Test results of VTM MRL tool "off" on various JVET CTC sequences in different configurations.

single 4xN transform in the vertical direction. Similarly, when a 4xN prediction region contains two 2xN transform blocks, the transform of the two 2xN blocks in each direction (horizontal and vertical) can be done in parallel. Thus, there is no added delay in processing these small blocks as processing intra blocks to regular 4x4 code.

For each sub-partition, reconstructed samples are obtained by adding the residual signal to the prediction signal. Here, a residual signal is generated by the processes such as entropy decoding, inverse quantization and inverse transform. Therefore, the reconstructed sample values of each sub-partition are available to generate the prediction of the next sub-partition, and each sub-partition is processed repeatedly. In addition, the first subpartition to be processed is the one that contains the top left sample of the UC, then continues down (horizontal split) or to the right (vertical split). Therefore, the reference samples used to generate the prediction signals for subpartitions are located only to the left and above the lines. All subpartitions share the same intra mode. The following are a summary of the ISP's interaction with other coding tools.

- Multiple Reference Line (MRL): if a block has an MRL index other than 0, then the ISP coding mode will be inferred to be 0 and therefore ISP mode information is not sent to the decoder.

- Entropy coding coefficient group size: the sizes of the entropy coding sub-blocks have been modified so that they have 16 samples in all possible cases, as shown in Table 7.3. Note that the new sizes only affect blocks produced by ISP in which one of the dimensions is less than 4 samples. In all other cases coefficient groups keep the 4×4 dimensions.

- CBF coding: it is assumed to have at least one of the sub-partitions with a non-zero CBF. Hence, if n is the number of sub-partitions and the first $n-1$ sub-partitions have produced a zero CBF, then the CBF of the n -th sub-partition is inferred to be 1.

Block Size	Coefficient group Size
1×N, N ≥ 16	1x16
N×1, N ≥ 16	16×1
2×N, N ≥ 8	2×8
N×2, N ≥ 8	8×2
All other possible M×N cases	4×4

TABLE 7.3: Entropy coding coefficient group size.

Intra mode	t_H	t_V
Planar Ang. 31, 32, 34, 36, 37	DST-VII	DST-VII
DC Ang. 33, 35	DCT-II	DCT-II
Ang. 2, 4, 6, . . . , 28, 30 Ang. 39, 41, 43, . . . , 63, 65	DST-VII	DCT-II
Ang. 3, 5, 7, . . . , 27, 29 Ang. 38, 40, 42, . . . , 64, 66	DCT-II	DST-VII

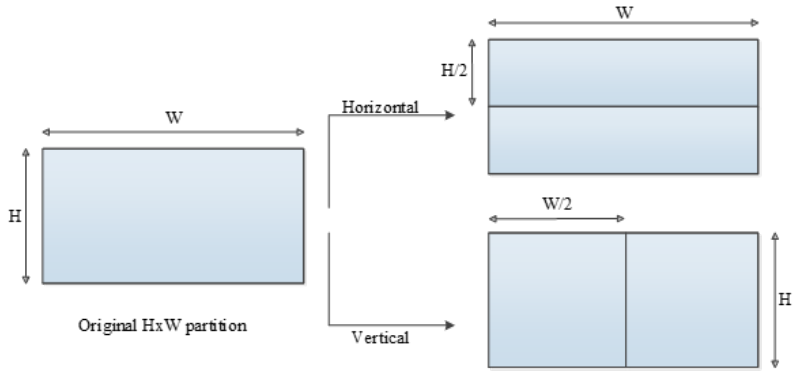
TABLE 7.4: Transform selection depends on intra mode.

- MPM usage: the MPM flag is inferred to be one in a block coded by ISP mode, and the MPM list is modified to exclude the DC mode and to prioritize horizontal intra modes for the ISP horizontal split and vertical intra modes for the vertical one.
- Transform size restriction: all ISP transforms with a length larger than 16 points uses the DCT-II.
- PDPC: when a CU uses the ISP coding mode, the PDPC filters are not applied to the resulting sub-partitions.
- MTS flag: if a CU uses the ISP coding mode, the MTS CU flag is set to 0 and it is not sent to the decoder. Therefore, the encoder does not perform RD tests for the different available transforms for each resulting sub-partition. Instead, the transform choice for ISP mode remains fix and is selected according to the intra mode, the processing order and the block size utilized. Hence, no signalling is required. For example, let t_H and t_V be the horizontal and the vertical transforms selected respectively for the $w\times$ sub-partition, where w is the width and h is the height. Then the transform is selected according to the following rules:

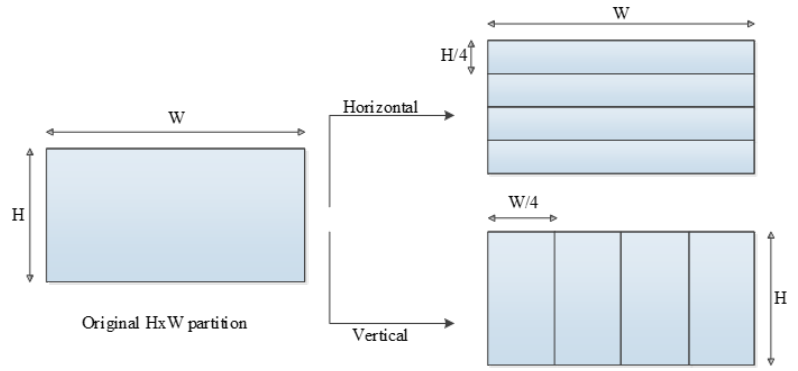
1. If $w = 1$ or $h = 1$, then there is no horizontal or vertical transform respectively;
2. If $w = 2$ or $w > 32$, t_H = DCT-II;
3. If $h = 2$ or $h > 32$, t_V = DCT-II;
4. Otherwise, the transform is selected conforming to Table 7.4.

In ISP mode, all 67 intra modes are allowed. PDPC is also applied if corresponding width and height is at least 4 samples long. In addition, the condition for intra interpolation filter selection does not exist anymore, and Cubic (DCT-IF) filter is always applied for fractional position interpolation in ISP mode.

Reference documents: JVET-O0106 [195], JVET-O0341 [196] and JVET-O0502 [197].



(A) Examples of sub-partitions for 4×8 and 8×4 CUs.



(B) Examples of sub-partitions for CUs other than 4×8, 8×4 and 4×4.

FIGURE 7.4: Sub-partition depending on the block size [8].

7.1.1.5 Matrix weighted Intra Prediction (MIP)

The Matrix weighted Intra Prediction (MIP) is an additional intra prediction tool introduced in VVC. This tool has up to 35 modes depending on the block size and generates its prediction through 3 main steps: averaging, matrix-vector multiplication and linear interpolation as shown in Figure 7.5. The first step, averaging, starts by taking the above and left boundary samples and reduces its number to 8 by averaging conforming to predefined rules (depending on the block size), as illustrated in Figure 7.5. In the case of a 4x4 block, only four samples are extracted by averaging. The obtained left and top reduced boundaries (or "red" in Figure 7.5) are then concatenated into a vector $bdry_{red}$ of size eight, except for 4x4 blocks.

The second step, matrix-vector multiplication, simply consists in multiplying the concatenated vector with a matrix A and adding an offset with a vector b to obtain the reduced/subsampled prediction signal

$$\text{pred}_{red} = A \cdot bdry_{red} + b. \tag{7.4}$$

A is a matrix that has the $\text{width}_{red} \cdot \text{height}_{red}$ rows (size of the reduced/subsampled prediction) and 8 columns (4 in 4x4 case). While, b is a vector of size $\text{width}_{red} \cdot \text{height}_{red}$. These matrices A and the offset vectors b are taken from three sets: $S0$ (18 matrices and 18 vectors for 4×4 blocks), $S1$ (10 matrices and vectors for 8×4 4×8 and 8×8 blocks) and $S2$ (6 matrices and vectors for other block sizes).

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	-0,06%	86%	0,11%	96%		
	Food Market	-0,01%	87%	0,0%	97%		
	Campfire	0,01%	86%	0,35%	93%		
A2	Cat Robot	0,33%	85%	0,25%	96%		
	Daylight Road	0,43%	85%	0,37%	95%		
	Park Running	0,07%	88%	0,03%	96%		
B	Market Place	0,10%	85%	0,13%	96%	0,10%	99%
	Ritual Dance	0,27%	85%	0,30%	95%	0,11%	99%
	Cactus	0,52%	84%	0,36%	95%	0,01%	99%
	Basketball Drive	0,66%	86%	0,45%	94 %	0,09%	98%
	BQ-Terrace	0,60%	85%	0,42%	97%	-0,12%	100%
C	Basketball Drill	1,00%	82%	0,63%	95%	-0,02%	99%
	BQ-Mall	0,95%	85%	0,50%	96%	0,05%	99%
	Party Scene	0,56%	83%	0,35%	95%	0,00%	99%
	Race Horses	0,46%	85%	0,27%	94%	0,10%	98%
Overall	BDR-Y	0,45%		0,32%		0,06%	
	BDR-U	0,24%		0,24%		-0,07%	
	BDR-V	0,19%		0,31%		-0,01%	
Average	BDR-YUV	**0,39%**	**85%**	**0,31%**	**95%**	**0,03%**	**99%**

TABLE 7.5: Test results of VTM ISP tool "off" on various JVET CTC sequences in different configurations.

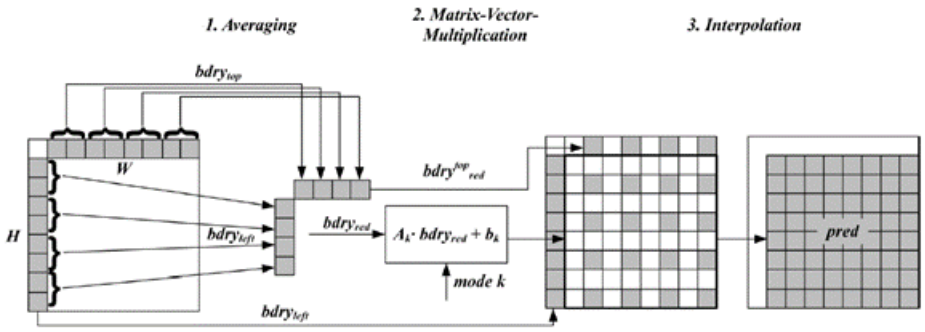


FIGURE 7.5: Matrix weighted intra prediction process [9].

The final step consists in generating the final prediction by linear interpolation of the subsampled prediction $pred_{red}$.

The number of supported MIP modes depends on block size, which is simply 2 times the number of matrix/vector minus 1, so 35, 19 or 11. Indeed, two modes share the same matrix and offset vector to reduce the memory requirement. These modes are directly coded using truncated binary code after a flag indicating if the intra prediction mode is a MIP one.

7.1.2 Inter Prediction

Some changes have been made in the Inter prediction mode. We can start by mentioning that for each inter-predicted CU, motion parameters consisting of motion vectors, reference picture indices and reference picture list usage index, and additional information needed for the new coding feature of VVC to be used for inter-predicted sample generation. The motion parameter can be signalled in an explicit or implicit manner. When a CU is coded with skip mode, the CU is associated with one PU and has

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	0,92%	90%	0,85%	95%		
	Food Market	0,85%	93%	0,57%	96%		
	Campfire	0,54%	88%	0,59%	94%		
A2	Cat Robot	0,53%	89%	0,24%	96%		
	Daylight Road	0,26%	88%	0,23%	95%		
	Park Running	0,48%	94%	0,19%	95%		
B	Market Place	0,58%	90%	0,43%	96%	0,24%	94%
	Ritual Dance	0,53%	90%	0,53%	96%	0,63%	95%
	Cactus	0,47%	88%	0,23%	95%	0,18%	93%
	Basketball Drive	0,25%	89%	0,30%	95 %	0,24%	93%
	BQ-Terrace	0,17%	87%	0,23%	96%	0,09%	97%
C	Basketball Drill	0,40%	88%	0,27%	95%	0,36%	93%
	BQ-Mall	0,44%	89%	0,29%	96%	0,18%	95%
	Party Scene	0,45%	89%	0,28%	95%	0,15%	95%
	Race Horses	0,42%	88%	0,22%	95%	0,22%	93%
Overall	BDR-Y	0,60%		0,33%		0,13%	
	BDR-U	0,14%		0,40%		0,66%	
	BDR-V	0,14%		0,52%		0,55%	
Average	BDR-YUV	**0,48%**	**89%**	**0,37%**	**95%**	**0,25%**	**94%**

TABLE 7.6: Test results of VTM MIP tool "off" on various JVET CTC sequences in different configurations.

no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current CU are obtained from neighbouring CUs, including spatial and temporal candidates, and additional schedules introduced in VVC. The merge mode can be applied to any inter-predicted CU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector, corresponding reference picture index for each reference picture list and reference picture list usage flag and other needed information are signalled explicitly per each CU. Beyond the inter coding features in HEVC, the VTM includes a number of new and refined inter prediction coding tools listed as follows:

- Affine motion inter prediction;
- Extended merge prediction;
- Block motion copy with spatial, temporal, history-based, and pairwise average merging candidates;
- Sub-block based temporal motion vector prediction (SbTMVP);
- Adaptive Motion Vector Resolution (AMVR);
- 8×8 block based motion compression for temporal motion prediction;
- High precision (1/16 pel) motion vector storage and motion compensation with 8-tap interpolation filter for luma component and 4-tap interpolation filter for chroma component;
- Triangular Partitions;
- Combined Intra and Inter Prediction (CIIP);
- Merge with MVD (MMVD);

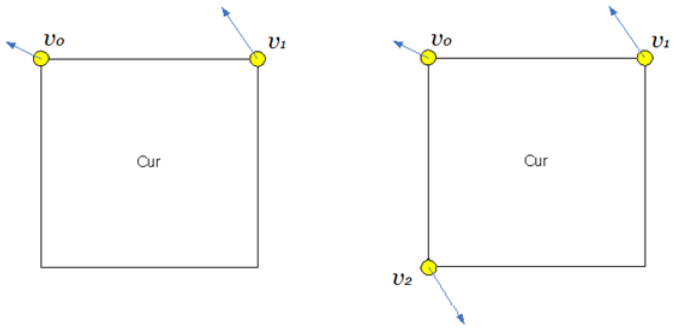


FIGURE 7.6: control point based affine motion models for 2 and 3 contorl points respectively [8].

- Symmetrical MVD coding (SMVD);
- Bi-directional Optical Flow (BDOF);
- Decoder-side Motion Vector Refinement (DMVR);
- Bi-prediction with CU-level Weight (BCW).

7.1.2.1 Affine motion compensated prediction

While HEVC predict inter picture only using block-based translation with one motion vector (MV), VVC introduces a block based affine motion model in addition to the classical translation model. Indeed, translation is not appropriate for complex motions such as zooms, rotations, etc. The affine motion model is described with two (V0 and V1) or three (V0, V1 and V2) Control Point MV (CPMV), as illustrated in Figure 7.6.

With these CPMVs, the affine motion of the current CU consists in applying a classical translation at a 4×4 sub-block level. Each of these sub-block motion vector is calculated according to the following equations. For two Control Points (CP), i.e., 4 -parameter affine motion model, motion vector at sample location (x, y) in a block is derived as:

$$\begin{cases} mvx = \frac{mv_{1x}-mv_{0x}}{w}x + \frac{mv_{1y}-mv_{0y}}{w}y + mv_{0x} \\ mvy = \frac{mv_{1y}-mv_{0y}}{w}x + \frac{mv_{1y}-mv_{0x}}{w}y + mv_{0y} \end{cases} \tag{7.5}$$

For three CP, i.e., 6 -parameter affine motion model, motion vector at sample location (x, y) in a block is derived as:

$$\begin{cases} mvx = \frac{mv_{1x}-mv_{0x}}{w}x + \frac{mv_{2x}-mv_{0x}}{w}y + mv_{0x} \\ mvy = \frac{mv_{1y}-mv_{0y}}{w}x + \frac{mv_{2y}-mv_{0y}}{w}y + mv_{0y} \end{cases} \tag{7.6}$$

(mv_{0x}, mv_{0y}) is the motion vector of the top-left control point, (mv_{1x}, mv_{1y}) is the motion vector of the top-right control point, and (mv_{2x}, mv_{2y}) is the motion vector of the bottom-left corner control point. Block-based affine transform prediction is used to simplify motion compensation prediction. In order to obtain the motion vector of each 4×4 luminance sub-block, the motion vector of the center sample of each sub-block, as shown in Figure 7.7, is calculated according to the previous equation (7.6) and rounded to the 1/16 fraction accuracy. Then a motion compensation interpolation filter is applied to

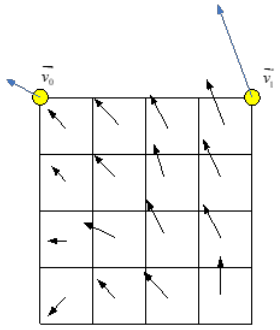


FIGURE 7.7: Affine MVF per sub-block [8].

use the obtained motion vector to generate a prediction for each sub-block. The sub-block size of the chrominance component is also set to 4×4. The 4×4 chroma sub-block is calculated as the average MV of the four corresponding 4×4 luma sub-blocks.

As done for translational motion inter prediction, there are also two affine motion inter prediction modes: affine merge mode and affine AMVP mode.

Affine merge prediction: For CUs with width and height more than or equal to 8, the AF MERGE mode can be used. The CPMVs of the present CU are generated in this mode based on the motion information of the spatial adjacent CUs. There can be up to five CPMVP candidates, with an index indicating which one is used for the current CU. The affine merge candidate list is composed of three sorts of CPVM candidates:

1. Inherited affine merge candidates that extrapolated from the CPMVs of the neighbour CUs;
2. Constructed affine merge candidates CPMVPs that are derived using the translational MVs of the neighbour CUs;
3. Zero MVs.

There are only two inherited affine possibilities in VVC, which are generated from the affine motion models of the surrounding blocks, one from the left neighboring CUs and one from the above neighboring CUs. Figure 7.8 depicts the potential blocks. The scan order for the left predictor is A0→A1, while the scan order for the upper predictor is B0→B1→B2. Only the first candidate inherited from each side is chosen. There is no pruning check between two inherited candidates. When an adjacent affine CU is discovered, the control point motion vectors of that CU are utilized to determine the CPMVP candidate in the present CU's affine merge list. As illustrated, if the neighbor left bottom block A is coded in affine mode, the motion vectors v_2, v_3 and v_4 of the top left corner, above right corner, and left bottom corner of the CU containing the block A are obtained. When block A is programmed using the 4 -parameter affine model, the current CU's two CPMVs are computed using v_2 and v_3. If block A is coded using a 6 -parameter affine model, the three CPMVs of the current CU are computed using $v2$, $v3$, and $v4$.

The term "constructed affine candidate" refers to a candidate that is built by combining the neighbor translational motion information of each control point. The motion information for the control points is generated from the spatial and temporal neighbors indicated in Figure 7.10. The k -th control point is represented by CPMVk ($k = 1, 2, 3, 4$). The B2→B3→A2 blocks are tested for CPMV1, and the MV of the first available block is utilized. The B1→B0 blocks are tested for CPMV2, while the A1→A0 blocks are checked for CPMV3. If TMVP is available, it is utilized as CPMV4. Following the completion of

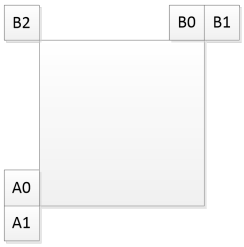


FIGURE 7.8: Locations of inherited affine motion predictors [10].

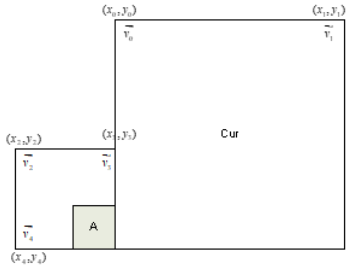


FIGURE 7.9: Control point motion vector inheritance [8].

MVs for four control points, affine merge candidates are constructed based on the motion information. The following combinations of control point MVs are used to construct in order: {CPMV1, CPMV2, CPMV3}, {CPMV1, CPMV2, CPMV4}, {CPMV1, CPMV3, CPMV4}, {CPMV2, CPMV3, CPMV4}, {CPMV1, CPMV2}, {CPMV1, CPMV3}.

The combination of three CPMVs yields a 6-parameter affine merge candidate, while the combination of two CPMVs yields a 4 -parameter affine merge candidate. If the reference indices of the control points differ, the associated combination of control point MVs is discarded to prevent motion scaling. If the list is still not full after checking inherited affine merge candidates and constructed affine merge candidates, zero MVs are added to the end of the list.

Affine AMVP prediction: Only for CUs with width and height more than or equal to 16, affine AMVP mode can be used. An affine flag is signalled in the bitstream at the CU level to indicate if affine AMVP mode is used, and another flag is signaled to indicate whether 4-parameter affine or 6-parameter

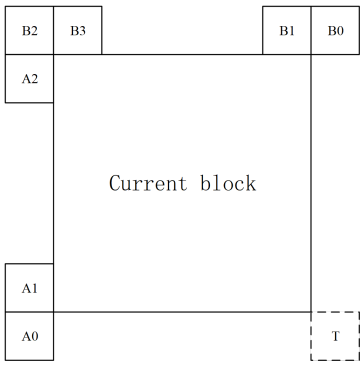


FIGURE 7.10: Locations of Candidates position for constructed affine merge mode [8].

affine is utilized. The difference between the current CU's CPMVs and their predictors CPMVPs is signaled in this mode.The affine AVMP candidate list has a size of two and is created by using the four types of CPVM candidates in the following order:

1. Inherited affine AMVP candidates that extrapolated from the CPMVs of the neighbour CUs;
2. Constructed affine AMVP candidates CPMVPs that are derived using the translational MVs of the neighbour CUs;
3. Translational MVs from neighboring CUs;
4. Zero MVs.

The checking order for inherited affine AMVP candidates is the same as for inherited affine merge candidates. The sole distinction is that only the affine CU with the same reference picture as the present block is evaluated for AVMP candidate. When an inherited affine motion predictor is added to the candidate list, no pruning is done. The constructed AMVP candidate is generated from the spatial neighbors detailed in Figure 7.10. The same checking order is utilized as in the creation of affine merge candidates. Furthermore, the reference picture index of the neighboring block is verified. The first inter coded block in the checking order with the same reference picture as in current CUs is utilized. When the current CU is coded in 4-parameter affine mode and mv_0 and mv_1 are both available, they are added to the affine AMVP list as one candidate. When the current CU is programmed in 6-parameter affine mode and all three CPMVs are available, they are added to the affine AMVP list as one candidate. Otherwise, the constructed AMVP candidate is marked as unavailable. If the affine AMVP list candidates are still fewer than two after checking inherited affine AMVP candidates and Constructed AMVP candidates, mv_0, mv_1 and mv_2 are added as translational MVs to predict all control point MVs of the current CU, when available. Finally, if the affine AMVP list is still not complete, zero MVs are utilized to fill it.

Affine AMVP prediction: The CPMVs of affine CUs are stored in a separate buffer in VVC. The stored CPMVs are only used to produce the inherited CPMVPs for the recently coded CUs in affine merge mode and affine AMVP mode. The sub-block MVs derived from CPMVs are used for motion compensation, MV derivation of merge/AMVP list of translational MVs and de-blocking. To prevent using the picture line buffer for the extra CPMVs, affine motion data inheritance from CUs above CTU is processed differently than inheritance from regular adjacent CUs. If the candidate CU for affine motion data inheritance is in the preceding CTU line, the bottom-left and bottom-right sub-block MVs in the line buffer are utilized for affine MVP derivation rather than the CPMVs. As a result, the CPMVs are simply saved in the local buffer. If the candidate CU is affine coded with six parameters, the affine model is reduced to four parameters. As illustrated in Figure 7.11 , the bottom-left and bottom-right sub-block motion vectors of a CU are employed for affine inheritance of the CUs in bottom CTUs along the top CTU boundary.

7.1.2.2 Prediction refinement with optical flow for affine mode (PROF)

When compared to pixel-based motion compensation, sub-block-based affine motion compensation can conserve memory access bandwidth and minimize calculation complexity at the expense of prediction accuracy. Prediction Refinement with Optical Flow (PROF) is utilized to enhance the sub-block-based affine motion compensated prediction without increasing the memory access bandwidth for motion compensation to obtain a finer granularity of motion compensation. After performing sub-block-based affine

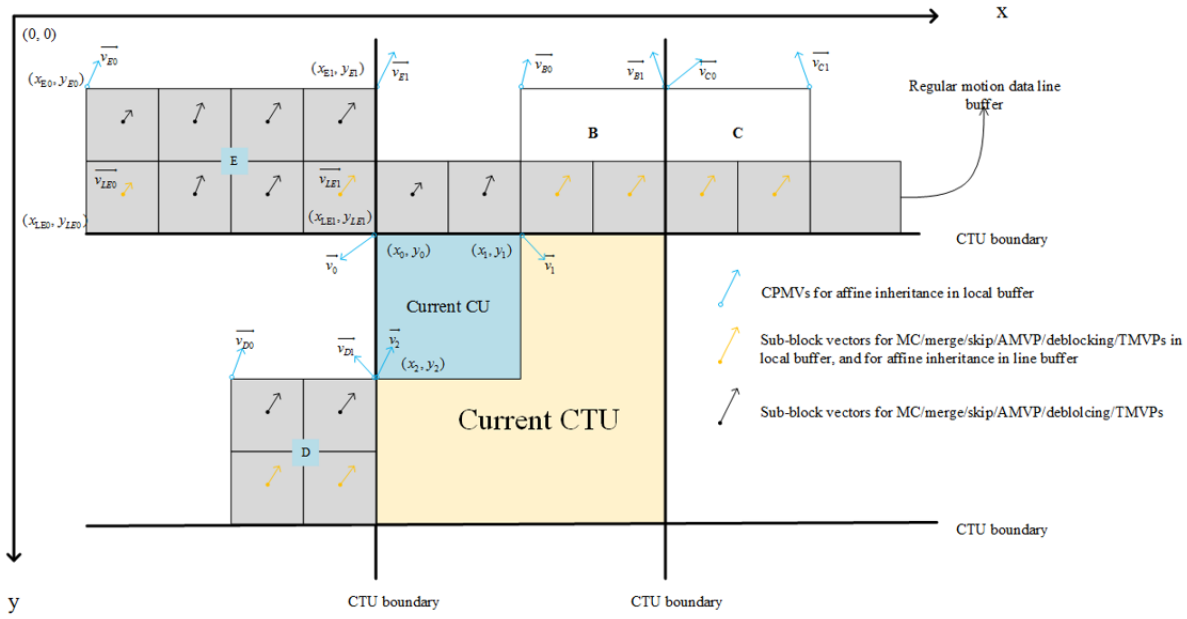


FIGURE 7.11: Illustration of motion vector usage for proposed combined method [11].

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	1,20%	79%		
	Food Market	4,23%	77%		
	Campfire	0,37%	88%		
A2	Cat Robot	7,37%	77%		
	Daylight Road	5,35%	77%		
	Park Running	3,47%	83%		
B	Market Place	4,33%	81%	3,79%	74%
	Ritual Dance	2,17%	80%	1,59%	75%
	Cactus	6,56%	83%	9,81%	73%
	Basketball Drive	1,04%	83%	1,50%	78%
	BQ-Terrace	0,24%	80%	0,47%	75%
C	Basketball Drill	0,49%	85%	0,62%	81%
	BQ-Mall	0,94%	83%	0,98%	78%
	Party Scene	2,56%	84%	4,70%	80%
	Race Horses	1,08%	84%	1,18%	84%
Overall	BDR-Y	3,01%		2,96%	
	BDR-U	2,04%		1,89%	
	BDR-V	2,00%		2,25%	
Average	BDR-YUV	**2,76%**	**82%**	**2,74%**	**78%**

TABLE 7.7: Test results of VTM AFFINE tool "off" on various JVET CTC sequences in different configurations.

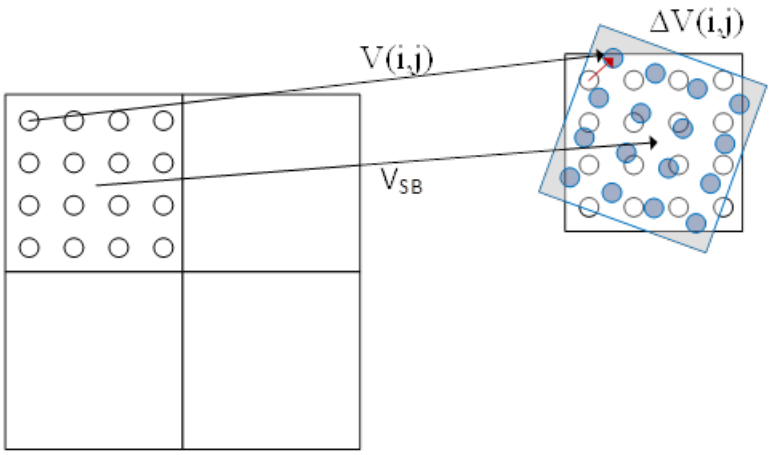


FIGURE 7.12: Sub-block MV VSB and pixel $\Delta v(i,j)$ (red arrow) [8].

motion correction, the luma prediction sample is improved in VVC by adding a difference derived from the optical flow equation. The PROF is broken down into four steps:

1. To create sub-block prediction $I(i,j)$, sub-block-based affine motion compensation is performed.

2. A 3-tap filter $[-1,0,1]$ is used to determine the spatial gradients $g_x(i,j)$ and $g_y(i,j)$ of the sub-block prediction at each sample position. The gradient calculation is identical to the BDOF gradient calculation.

$$g_x(i,j) = \big(I(i+1,j) >> shift1\big) - \big(I(i-1,j) >> shift1\big)$$
$$g_y(i,j) = \big(I(i,j+1) >> shift1\big) - \big(I(i,j-1) >> shift1\big)$$

   shift1 is used to adjust the accuracy of the gradient. This is done in the gradient computation, where the sub-block (i.e., 4×4) prediction is expanded by one sample on each side. To avoid additional memory bandwidth and interpolation computation, those expanded samples on the extended boundaries are copied from the reference picture's nearest integer pixel position.

3. The optical flow equation below is used to compute the luma prediction refinement.

$$\Delta I(i,j) = g_x(i,j) * \Delta v_x(i,j) + g_y(i,j) * \Delta v_y(i,j)$$

   where $\Delta v(i,j)$ represents the difference between the sample MV computed for sample location (i,j), denoted by $v(i,j)$, and the sub-block MV of the sub-block to which sample (i,j) belongs, as illustrated in Figure 7.12. The $\Delta v(i,j)$ is quantized in the unit of 1/32 luam sample precision. Given that the affine model parameters and sample location relative to the sub-block center remain constant from sub-block to sub-block, $\Delta v(i,j)$ may be computed for the first sub-block and utilized for subsequent sub-blocks in the same CU. Let $dx(i,j)$ and $dy(i,j)$ be the horizontal and vertical offsets from the sample position (i,j) to the sub-block center (x_{SB}, y_{SB}), respectively, then $\Delta v(x,y)$ may be calculated as follows:

$$\begin{cases} dx(i,j) = i - x_{SB} \\ dy(i,j) = j - y_{SB} \end{cases} \quad \begin{cases} \Delta v_x(i,j) = C * dx(i,j) + D * dy(i,j) \\ \Delta v_y(i,j) = E * dx(i,j) + F * dy(i,j) \end{cases}$$

To maintain the precision, the entry of the sub-block (x_{SB},y_{SB}) is computed as ((W_{SB}-1)/2, (H_{SB}-1)/2), where W_{SB} and H_{SB} are the width and height of the sub-block, respectively. For 4 -parameter affine model,

$$\begin{cases} C = F = \frac{v_{1x} - v_{0x}}{w} \\ E = -D = \frac{v_{1y} - v_{0y}}{w} \end{cases} \tag{7.7}$$

For 6 -parameter affine model,

$$\begin{cases} C = \frac{v_{1x} - v_{0x}}{w} \\ D = \frac{v_{2x} - v_{0x}}{h} \\ E = \frac{v_{1y} - v_{0y}}{w} \\ F = \frac{v_{2y} - v_{0y}}{h} \end{cases} \tag{7.8}$$

where (v_{0x},v_{0y}), (v_{1x},v_{1y}), (v_{2x},v_{2y}) are the top-left, top-right and bottom-left control point motion vectors, w and h are the width and height of the CU.

- Finally, the sub-block prediction $I(i,j)$ is refined with the luma prediction refinement $\Delta I(i,j)$. The following equation yields the final prediction I'.

$$I'(i,j) = I(i,j) + \Delta I(i,j) \tag{7.9}$$

PROF is not used for an affine coded CU in two cases: 1) all control point MVs are the same, indicating that the CU only has translational motion; 2) the affine motion parameters are larger than a predefined limit because the sub-block-based affine MC is downgraded to CU-based MC to prevent high memory access bandwidth requirements. To decrease the encoding complexity of affine motion estimation using PROF, a fast encoding approach is used. PROF is not used during the affine motion estimation stage in the following two cases: a) if this CU is not the root block and its parent block does not choose the affine mode as its best mode, PROF is not applied since the probability of the current CU choosing the affine mode as best mode is low; b) if the magnitudes of the four affine parameters (C, D, E, F) are all less than a predefined threshold and the current picture is not a low delay picture, PROF is not applied because the corresponding benefit is tiny in this situation. As a result, the affine motion estimation using PROF may be sped up.

7.1.2.3 Sub-block-based Temporal Motion Vector Prediction (SbTMVP)

VVC supports the Sub-block-based Temporal Motion Vector Prediction (SbTMVP) method. Similar to the Temporal Motion Vector Prediction (TMVP) in HEVC, SbTMVP uses the motion field within the collocated image to improve motion vector prediction and merge mode for CUs in the current picture. SbTVMP employs identical collocated picture used by TMVP. SbTMVP differs from TMVP in the following 2 main aspects:

1. TMVP predicts motion at CU level but SbTMVP predicts motion at sub-CU level.

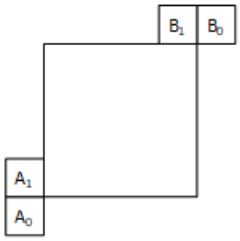


FIGURE 7.13: Spatial neighboring blocks used by SbTVMP [8].

2. Whereas TMVP fetches the temporal motion vectors from the collocated block in the collocated picture (the collocated block is the bottom-right or center block relative to the current CU), SbTMVP applies a motion shift before fetching the temporal motion information from the collocated picture, where the motion shift is obtained from the motion vector from one of the spatial neighboring blocks of the current CU.

The SbTVMP process is shown in Figures 7.13 and 7.14. SbTMVP predicts the motion vector of sub-CU in the current CU in two steps. In the first step, check the spatial neighbor A1 in Figure 7.13. If A1 has a motion vector that uses the collocated image as a reference image, select this motion vector as the applied motion shift. If such motion is not recognized, the motion shift is set to (0, 0). In the second step, apply the motion shift identified in step 1 (that is, the coordinates added to the current block) to obtain the sub-CU-level motion information (motion vectors and reference indices) of the interlaced image, as shown in Figure 7.14. In fact, the example in Figure 7.14 assumes that the motion shift is set to block A1's shift. Then, for each sub-CU, the motion information from the corresponding block in the collocated image (the smallest motion grid covering the center sample) is used to obtain the motion information for the sub-CU. Once this motion information is identified, it is used to transform the motion vectors and reference indices of the current SubCU, similar to the HEVC TMVP process, which uses the temporal motion scale to match the temporal motion vector reference image with those of the current CU.

In VVC, the sub-block merge list containing SbTVMP candidates and related merge candidates is used to indicate the sub-block based merge mode. The SbTVMP mode is activated/deactivated by the Sequence Parameter Set (SPS) flag. When enabled, the predictor of SbTMVP is added as the first entry in the sub-block-based merge candidates list, and then followed by the affine merge candidates. The size of sub-block based merge list is signalled in SPS and the maximum allowed size of the sub-block based merge list is 5 in VVC.

The size of the sub-CU used in SbTMVP is fixed to 8x8. Like the affine merge mode, the SbTMVP mode is only applicable to CUs with a width and height greater than or equal to 8. SbTMVP sub-merge coding logic candidates are the same as other merge candidates, that is, for each CU in P or B slice, an additional RD test is performed to determine whether to use the SbTMVP candidate.

The performance of this tools is also evaluated in the Table 7.8.

7.1.2.4 Adaptive Motion Vector Resolution (AMVR)

In HEVC, when *use_integer_mv_flag* is 0 in the slice header, the Motion Vector Differences (MVD) (between the motion vector and the predicted motion vector of a CU) is reported in quarter-luma-sample units. The CU-level AMVR scheme, introduced in VVC, allows to encode the MVD CU with different

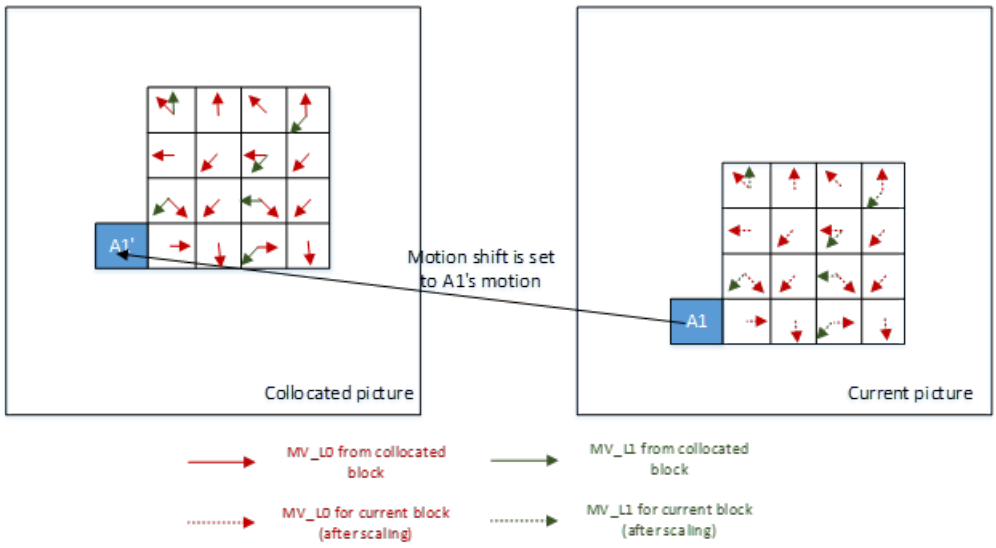


FIGURE 7.14: Deriving sub-CU motion field by applying a motion shift from spatial neighbor and scaling the motion information from the corresponding collocated sub-CUs [8].

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	1,15%	101%		
	Food Market	0,42%	100%		
	Campfire	0,03%	100%		
A2	Cat Robot	0,78%	102%		
	Daylight Road	0,29%	101%		
	Park Running	0,05%	100%		
B	Market Place	0,16%	100%	0,11%	100%
	Ritual Dance	0,10%	100%	0,07%	100%
	Cactus	0,75%	101%	1,11%	100%
	Basketball Drive	0,30%	100%	0,21%	100%
	BQ-Terrace	0,30%	101%	0,33%	101%
C	Basketball Drill	0,52%	101%	0,41%	102%
	BQ-Mall	1,04%	102%	1,10%	102%
	Party Scene	0,54%	101%	0,34%	102%
	Race Horses	0,04%	100%	0,02%	99%
Overall	BDR-Y	0,46%		0,42%	
	BDR-U	0,32%		0,32%	
	BDR-V	0,36%		0,44%	
Average	**BDR-YUV**	**0,43%**	**101%**	**0,41%**	**101%**

TABLE 7.8: Test results of VTM SbTMVP tool "off" on various JVET CTC sequences in different configurations.

precision. According to the current CU mode (normal AMVP mode or affine AMVP mode), the current CU's MVD can be adaptively selected as follows:

- Normal AMVP mode: quarter-luma-sample, half-luma-sample, integer-luma-sample or four-luma-sample.
- Affine AMVP mode: quarter-luma-sample, integer-luma-sample or 1/16 luma-sample.

If the current CU has at least one non-zero MVD component, the CU-level MVD resolution indication is conditionally signalled. If all MVD components (i.e., the horizontal and vertical MVDs of the L0 and L1 reference lists) are zero, the MVD resolution of the quarter-luma-sample is inferred. For CU with at least one non-zero MVD component, a first flag is signaled to indicate whether a quarter-luma-sample MVD precision is being used for the CU. If the first flag is 0, no further signaling is needed and quarter-luma-sample MVD precision is used for the current CU. Otherwise, a second flag is issued to indicate that half-luma-sample or another MVD accuracy (interger or four-luma sample) is being used for the normal AMVP CU. In the case of half-luma-sample, a 6-tap interpolation filter instead of the default 8-tap interpolation filter is used for the half-luma sample position. Otherwise, the third flag is set to indicate whether to use the MVD accuracy of integer-luma-sample or the MVD accuracy of four-luma-sample for the normal AMVP CU. In the case of affine AMVP CU, the second flag is used to indicate whether integer-luma-sample or 1/16 luma-sample MVD precision is used. To ensure that the reconstructed MV has the required accuracy (quarter-luma-sample, half-luma-sample, integer-luma-sample or four-luma-sample), the motion vector predictors of the CU is rounded to the previous MVD before being added together with the MVD. The motion vector predictors are rounded to zero, that is, the negative motion vector predictor is rounded to positive infinity, and the positive motion vector predicted value is rounded to negative infinity.

The encoder determines the current CU's motion vector resolution by checking the RD. To avoid running the CU-level RD check four times for each MVD resolution, VTM7 only calls the RD check of MVD precisions other than quarter-luma-sample, conditionally. First, the RD cost of quarter-luma-sample MVD precision and integer-luma sample MV precision, is computed for normal AMVP. Then, the RD cost of integer-luma-sample MVD precision is compared to that of quarter-luma-sample MVD precision to determine whether the RD cost needs to be further checked for the four-luma-sample MVD precision. If the RD value of the MVD accuracy of the quarter-luma-sample MVD is much smaller than the MVD accuracy of the integer-luma-sample, the RD check of four-luma-sample MVD precision is skipped. If the RD cost of the MVD accuracy of the integer-luma-sample is significantly higher than the best RD cost of the previously tested MVD accuracy, the MVD accuracy test of the half-luma-sample is skipped. For affine AMVP mode, if after checking rate-distortion costs of affine merge/skip mode, merge/skip mode, quarter-luma-sample MVD precision normal AMVP mode and quarter-luma-sample MVD precision affine AMVP mode, the affine inter mode has not been selected, then 1/16 luma-sample MV precision and 1-pel MV precision affine inter modes are not checked. In addition, in 1/16 luma-sample and quarter-luma-sample MV precision affine inter modes, the affine parameters obtained in quarter-luma-sample MV precision affine inter mode are used as starting search point of the search.

Similarly to former tools, this one is also evaluated and the numbers are summarized in Table 7.9.

7.1.2.5 Bi-prediction with CU-level Weight (BCW)

In HEVC, a bi-prediction signal is generated by averaging two prediction signals obtained from two different reference pictures and/or using two different motion vectors. In VVC, the bi-prediction mode

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	3,11%	84%		
	Food Market	1,41%	85%		
	Campfire	1,08%	85%		
A2	Cat Robot	1,64%	82%		
	Daylight Road	2,37%	82%		
	Park Running	1,25%	83%		
B	Market Place	1,32%	82%	0,47%	86%
	Ritual Dance	1,87%	83%	0,73%	84%
	Cactus	1,10%	86%	0,43%	86%
	Basketball Drive	2,60%	84%	1,18%	84%
	BQ-Terrace	1,24%	86%	0,96%	87%
C	Basketball Drill	1,76%	86%	0,95%	87%
	BQ-Mall	0,79%	83%	0,46%	89%
	Party Scene	0,81%	87%	0,32%	87%
	Race Horses	1,73%	82%	0,92%	81%
Overall	BDR-Y	1,42%		0,66%	
	BDR-U	2,17%		0,85%	
	BDR-V	2,27%		0,92%	
Average	BDR-YUV	**1,62%**	**84%**	**0,72%**	**85%**

TABLE 7.9: Test results of VTM AMVR tool "off" on various JVET CTC sequences in different configurations.

extends beyond the simple average to achieve a weighted average of the two prediction signals.

$$P_{bi-pred} = ((8 - w) * P_0 + w * P_1 + 4) >> 3 \tag{7.10}$$

The weighted average bi-prediction allows five weights, $w \in \{2,3,4,5,10\}$. For each bi-predictive CU, the weight w is determined using one of the following two ways: 1) for non-merge CUs, the weight index is signalled after the motion vector difference; 2) for merge CUs, the weight index is derived from neighbouring blocks based on the merge candidate index. BCW is only applicable to CUs with 256 or more luma samples (that is, the width of the CU multiplied by the height of the CU is greater than or equal to 256). All 5 weights are used for low-delay pictures. Only 3 weights ($w \in \{3,4,5\}$) are used for non-low-delay pictures.

- The encoder uses a fast search algorithm to find the weight index without significantly increasing the complexity of the encoder. These algorithms are briefly introduced below. More information can be found in the VTM software and document JVETL0646. Combined with AMVR, unequal weights are only conditionally checked for 1-pel and 4-pel motion vector precisions if the current picture is a low-delay picture.

- Combined with affine, if the affine mode is selected as the current best mode, affine ME only performs unequal weights.

- If the two reference pictures in the bidirectional prediction are the same, unequal weights are only conditionally checked.

- Unequal weights are not searched when certain conditions are met, which depends on the POC distance between current picture and its reference pictures, QP coding and the temporal level.

The BCW weight index is coded using one context coded bin followed by bypass coded bins. The first context-coded bin indicates whether the same weight is used; otherwise additional bins are signalled with a

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	0,44%	94%		
	Food Market	0,41%	95%		
	Campfire	0,46%	95%		
A2	Cat Robot	0,65%	93%		
	Daylight Road	0,25%	92%		
	Park Running	0,33%	93%		
B	Market Place	0,56%	94%	0,20%	96%
	Ritual Dance	0,34%	95%	0,01%	97%
	Cactus	0,41%	95%	0,26%	97%
	Basketball Drive	0,59%	95%	0,13%	96 %
	BQ-Terrace	0,95%	93%	0,80%	93%
C	Basketball Drill	0,11%	95%	0,23%	96%
	BQ-Mall	0,16%	95%	0,07%	97%
	Party Scene	0,23%	95%	0,15%	94%
	Race Horses	0,30%	94%	0,16%	97%
Overall	BDR-Y	0,40%		0,28%	
	BDR-U	0,42%		-0,03%	
	BDR-V	0,45%		0,11%	
Average	BDR-YUV	**0,41%**	**94%**	**0,22%**	**96%**

TABLE 7.10: Test results of VTM BCW tool "off" on various JVET CTC sequences in different configurations.

bypass coding to indicate the unequal weight being used. Weighted Prediction (WP) is a H.264/AVC and HEVC coding tool for efficiently coding fading video content. WP support has also been added to the VVC standard. Using WP, weighting parameters (weight and offset) are signalled for each reference picture in each of the reference picture lists L0 and L1. Then the weight(s) and offset(s) of the corresponding reference picture(s) are used during motion compensation. WP and BCW are suitable for different types of video content. In order to avoid the interactions between WP and BCW, which will complicate the design of the VVC decoder, the BCW weight is not transmitted when the CU uses WP. w is assumed to be 4 (i.e., the same weight is applied). The weight index is inferred from neighboring blocks based on the merge candidate index. This can be applied to regular merge mode and inherited affine merge mode. For constructed affine merge mode, affine motion information is generated from the motion information of at most 3 blocks. The BCW index of the CU using the constructed affine merge mode is simply set to the BCW index of the first control point MV. In VVC, CIIP and BCW cannot be used together for a CU. When a CU is coded in CIIP mode, the BCW index of the current CU is set to 2, e.g. the same weight.

Table 7.10 shows the simulation results of BCW with the CU size constraint that disables this tool for CU size smaller than 256 (which means when the number of Luma samples is smaller than 256).

7.1.2.6 Merge mode with MVD (MMVD)

In addition to directly using the implicitly derived motion information to generate the prediction samples of the current CU in the merge mode, the MVD has been introduced into this mode for VVC. This consists thus the Merge mode with MVD (MMVD). The MMVD flag is displayed immediately after sending the skip flag and the merge flag, to indicate whether the MMVD mode should be used for the CU. In MMVD, after a merge candidate is selected, it is further refined by the signalled MVDs information. The additional information includes a combination of a merge candidate flag, an index for indicating the magnitude of the movement, and an index for indicating the movement direction. MMVD chooses one of the first two

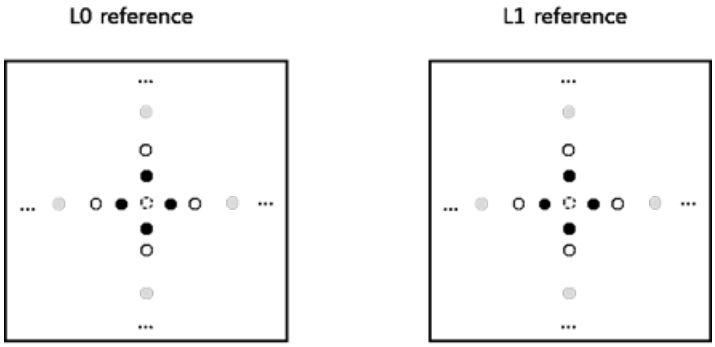


FIGURE 7.15: MMVD Search Point [8].

Distance IDX	0	1	2	3	4	5	6	7
Offset (in unit of luma sample)	1/4	1/2	1	2	4	8	16	32

TABLE 7.11: The relation of distance index and pre-defined offset

candidates in the merge list to be used as the basis for the MV. The merge candidate flag is signalled to specify which one is used.

The distance index defines information about the motion magnitude and specifies the pre-defined offset from the starting point. As shown in Figure 7.15, the offset is added to the horizontal or vertical component of the startung MV. The relation of distance index and the predefined offset are shown in Table 7.11. The direction index indicates the direction of the MVD relative to the starting point. The direction index can represent all our directions, as shown in Table 7.12. When the starting MV is an uni-prediction MV or a bi-prediction MV, and both lists point to the same side of the current picture (i.e., the two references POCs are greater than the POC of the current picture or both are smaller than the POC of the current picture), the sign in Table 7.12 specifies the MV offset sign added to the starting MV. If the initial MV is a bi-prediction MV, where the two MVs point to different sides of the current picture (the POC of the current picture is smaller than one of the reference and bigger than the other reference), the symbol in Table 7.12 indicates the sign of MV offset added to List 0 MV component of starting MV and the sign of List1 MV has the opposite value.

Table 7.13 evaluates the performance of this tool.

7.1.2.7 Symmetric MVD coding (SMVD)

In addition to the normal unidirectional prediction and bidirectional prediction mode MVD signaling, the bi-predictional MVD signaling adopts the symmetric MVD mode. In this mode, motion information including reference picture indices of both list-0 and list-1 and MVD of list-1 are not signaled but derived. The decoding process of the symmetric MVD mode is as follows:

Distance IDX	00	01	10	11
x-axis	+	-	N/A	N/A
y-axis	N/A	N/A	+	-

TABLE 7.12: Sign of MV offset specified by direction index.

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	0,56%	97%		
	Food Market	0,46%	92%		
	Campfire	0,16%	95%		
A2	Cat Robot	0,49%	92%		
	Daylight Road	0,82%	95%		
	Park Running	0,58%	94%		
B	Market Place	0,50%	87%	0,69%	96%
	Ritual Dance	0,47%	84%	0,51%	96%
	Cactus	0,25%	96%	-0,07%	95%
	Basketball Drive	0,93%	99%	0,76%	96%
	BQ-Terrace	0,82%	92%	0,51%	93%
C	Basketball Drill	0,05%	91%	-0,08%	95%
	BQ-Mall	0,62%	96%	0,56%	96%
	Party Scene	0,54%	91%	0,51%	96%
	Race Horses	0,35%	98%	0,64%	97%
Overall	BDR-Y	0,51%		0,49%	
	BDR-U	0,47%		0,22%	
	BDR-V	0,51%		0,43%	
Average	BDR-YUV	**0,51%**	**93%**	**0,45%**	**96%**

TABLE 7.13: Results of VTM MMVD tool "off" on various JVET CTC sequences in different configurations.

1. At the slice level, the variables BiDirPredFlag, RefIdxSymL0 and RefIdxSymL1 are derived as follows:
   - If $mvd\_l1\_zero\_flag$ is equal to 1, the value of BiDirPredFlag is 0.
   - If the nearest reference picture in the list is 0 and the nearest reference picture in the list-1 picture form a forward and backward pair of reference pictures or a backward and forward pair of reference pictures, the value of BiDirPredFlag is 1, and the reference pictures list-0 and list-1 are short-term reference pictures. Otherwise, BiDirPredFlag is set to 0
2. When the CU is coded in bi-prediction mode and BiDirPredFlag is 1, the symmetric mode flag is explicitly sent to indicate whether to use the symmetric mode or not.

When the symmetry mode flag is true, only mvp_l0_flag, mvp_l1_flag and MVD0 are explicitly signaled. The reference indices of list0 and list1 are respectively set equal to the pair of reference pictures. MVD1 is set equal to -MVD0. The final motion vector is shown in the following formula:

$$\begin{cases} (mvx_0, mvy_0) = (mvpx_0 + mvdx_0, mvpy_0 + mvdy_0) \\ \\ (mvx_1, mvy_1) = (mvpx_1 - mvdx_0, mvpy_1 + mvdy_0) \end{cases} \tag{7.11}$$

In the encoder, symmetrical MVD motion estimation starts from the initial MV evaluation. A set of initial MV candidates, including MV obtained from uni-prediction search, the MV obtained from bi-prediction search and the MVs from the AMVP list. The initial MV for the symmetric MVD motion search is the one with the lowest rate-distortion cost.

The performance of SMVD is summarized in Table 7.14. This tool can be applied only in Random Access configuration.

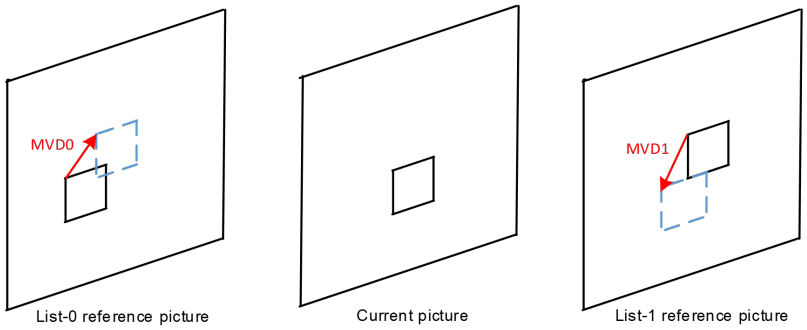


FIGURE 7.16: Illustration for symmetrical MVD mode [8].

Class	Sequence	Random Access	
		BDR-YUV	ET
A1	Tango	0,37%	95%
	Food Market	0,31%	96%
	Campfire	0,08%	93%
A2	Cat Robot	0,30%	92%
	Daylight Road	0,30%	97%
	Park Running	0,35%	93%
B	Market Place	0,20%	92%
	Ritual Dance	0,29%	91%
	Cactus	0,22%	91%
	Basketball Drive	0,42%	91%
	BQ-Terrace	0,25%	90%
C	Basketball Drill	0,22%	91%
	BQ-Mall	0,14%	93%
	Party Scene	0,26%	95%
	Race Horses	0,13%	91%
Overall	BDR-Y	0,25%	
	BDR-U	0,25%	
	BDR-V	0,26%	
Average	BDR-YUV	**0,26%**	**93%**

TABLE 7.14: Results of VTM SMVD tool "off" on various JVET CTC sequences in RA configuration.

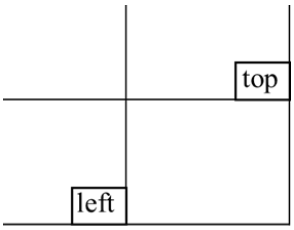


FIGURE 7.17: Top and left neighboring blocks used in CIIP weight derivation [8].

7.1.2.8 Combined Inter and Intra Prediction (CIIP)

In VVC, if a CU is coded in merge mode and it contains at least 64 luma samples (i.e., CU width times CU height is equal to or greater than 64), and if the width and the height are less than for 128 luma samples, an additional flag is signalled to indicate whether the CIIP mode is being applied to the current CU. As the name indicates, CIIP prediction combines the inter prediction signal and the intra prediction signal. The inter prediction signal in the CIIP mode Pinter is derived using the same inter prediction process applied to regular merge mode; the intra prediction signal is derived following the regular intra prediction process with the planar mode. Then a weighted average is used to combine the intra and inter prediction signals, and the weight value is calculated based on the coding mode of the top neighbor and the left neighbor blocks (shown in Figure 7.17) are as follows:

- If the top neighbor is available and coded, set isIntraTop to 1; otherwise, set isIntraTop to 0;
- If the left neighbor is available and coded, set isIntraLeft to 1;
- If (isIntraLeft + isIntraLeft) is 2, then wt is set to 3;
- If (isIntraLeft + isIntraLeft) is 1, then wt is set to 2;
- otherwise, wt is set to 1.

The CIIP prediction is formed as follows:

$$P_{CIIP} = ((4 - wt) * P_{inter} + wt * P_{intra} + 2) >> 2$$

7.1.2.9 Bi-Directional Optical Flow (BDOF)

The Bi-directional Optical Flow (BDOF) tool is included in VVC. BDOF, formerly known as BIO, has been adopted into JEM. Compared with the JEM version, the BDOF in VVC is a simpler version and requires less calculation, especially in terms of the number of multiplications and the size of the multiplier. BDOF is used to refine the bi-prediction signal of the CU at the 4×4 sub-block level. When all the following conditions are met, BDOF is applied to CU:

- CU is coded in a "true" bi-prediction mode; that is, one of the two reference pictures is prior to the current picture in display order and the other comes after;
- The POC difference from two reference pictures to the current picture are the same;

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	0,25%	96%		
	Food Market	0,09%	98%		
	Campfire	0,22%	100%		
A2	Cat Robot	0,14%	96%		
	Daylight Road	0,15%	96%		
	Park Running	0,18%	97%		
B	Market Place	0,15%	96%	0,43%	117%
	Ritual Dance	0,28%	90%	0,78%	100%
	Cactus	0,60%	100%	0,86%	98%
	Basketball Drive	0,23%	101%	0,48%	97%
	BQ-Terrace	0,04%	100%	0,13%	97%
C	Basketball Drill	0,28%	101%	0,47%	97%
	BQ-Mall	0,13%	105%	0,30%	98%
	Party Scene	0,31%	105%	0,38%	96%
	Race Horses	0,18%	101%	0,36%	98%
Overall	BDR-Y	0,28%		0,46%	
	BDR-U	0,01%		0,37%	
	BDR-V	0,01%		0,59%	
Average	BDR-YUV	**0,21%**	**99%**	**0,47%**	**100%**

TABLE 7.15: Results of VTM CIIP tool "off" on various JVET CTC sequences in different configurations.

- Both reference pictures are short-term reference pictures;
- CU is not coded in affine mode or ATMVP merge mode;
- CU has more than 64 luma samples;
- CU height and CU width are both greater than or equal to 8 luma samples;
- BCW-weight index shows the same weight;
- Current CU does not include WP;
- The current CU does not use CIIP mode.

BDOF is only applied on the luma component. BDOF mode is based on the concept of optical flow, as its name indicates. This concept assumes that the motion of an object is smooth. For each 4×4 sub-block, the motion refinement (v_x, v_y) is calculated by minimizing the difference between the prediction samples L0 and L1. Then the motion refinement is used to adjust the bi-predicted samples in the 4x4 sub-blocks. The following steps are specific to the BDOF process.

First, the horizontal and vertical gradients, $\frac{\partial I^{(k)}}{\partial x}(i,j)$ and $\frac{\partial I^{(k)}}{\partial y}(i,j)$, $k = 0, 1$, of the two prediction signals are computed by directly calculating the difference between two neighboring samples, i.e.,

$$\begin{cases} \frac{\partial I^{(k)}}{\partial x} = ((I^{(k)}(i+1,j) >> \mathit{shift1}) - (I^{(k)}(i-1,j) >> \mathit{shift1})) \\ \frac{\partial I^{(k)}}{\partial y} = ((I^{(k)}(i,j+1) >> \mathit{shift1}) - (I^{(k)}(i,j-1) >> \mathit{shift1})) \end{cases} \tag{7.12}$$

where $I^{(k)}(i,j)$ are the sample value at coordinate (i,j) of the prediction signal in list k, k=0,1, and shift1 is calculated based on the luma bit depth, bitDepth, as shift1 = max(6, bitDepth-6). Then, the auto-

and cross-correlation of the gradients, S_1, S_2, S_3, S_5 and S_6, are calculated as:

$$S_1 = \sum_{(i,j)\in\Omega} \text{Abs}(\psi_x(i,j)), \qquad S_2 = \sum_{(i,j)\in\Omega} \psi_x(i,j)\cdot \text{Sign}(\psi_y(i,j)),$$
$$S_3 = \sum_{(i,j)\in\Omega} \theta(i,j)\cdot \text{Sign}(\psi_x(i,j)), \qquad S_5 = \sum_{(i,j)\in\Omega} \text{Abs}(\psi_y(i,j)),$$
$$S_6 = \sum_{(i,j)\in\Omega} \theta(i,j)\cdot \text{Sign}(\psi_y(i,j))$$

where

$$\psi_x(i,j) = \Big(\frac{\partial I^{(1)}}{\partial x}(i,j) + \frac{\partial I^{(0)}}{\partial x}(i,j)\Big) >> n_a$$
$$\psi_y(i,j) = \Big(\frac{\partial I^{(1)}}{\partial y}(i,j) + \frac{\partial I^{(0)}}{\partial y}(i,j)\Big) >> n_a$$
$$\theta(i,j) = \big(I^{(1)}(i,j) >> n_b\big) - \big(I^{(0)}(i,j) >> n_b\big)$$

where Ω is a 6×6 window around the 4×4 sub-block, and the values of n_a and n_b are set equal to min(1, bitDepth - 11) and min(4, bitDepth – 8), respectively. The motion refinement (v_x, v_y) is then derived using the cross- and auto-correlation terms using the following:

$$v_x = S_1 > 0 ? clip3(-th'_{BIO}, th'_{BIO}, -\big((S_3 \cdot 2^{n_b-n_a} >> \lfloor \log_2 S_1 \rfloor)\big) : 0$$
$$v_y = S_5 > 0 ? clip3\Big(-th'_{BIO}, th'_{BIO}, -\big((S_6 \cdot 2^{n_b-n_a} - \big((v_x S_{2,m}) << n_{S_2} + v_x S_{2,s}\big)/2\big) >> \lfloor log_2 S_5 \rfloor\big)\Big) : 0$$

where $S_{2,m} = S_2 >> n_{S_2}$, $S_{2,s} = S_2 \& (2^{n_{S_2}} - 1)$, $th'_{BIO} = 2^{\max(5, BD-7)}$, $\lfloor \cdot \rfloor$ is the floor function, and $n_{S_2} = 12$.

Based on the motion refinement and the gradients, the following adjustment is calculated for each sample in the 4×4 sub-block:

$$b(x,y) = \text{round}\Big(\big(v_x(\frac{\partial I^{(1)}(x,y)}{\partial x} - \frac{\partial I^{(0)}(x,y)}{\partial x}) + v_y(\frac{\partial I^{(1)}(x,y)}{\partial y} - \frac{\partial I^{(0)}(x,y)}{\partial y}) + 1\big)/2\Big)$$

Finally, the BDOF samples of the CU are calculated by adjusting the bi-prediction samples as follows:

$$\text{pred}_{\text{BDOF}}(x,y) = (I^{(0)}(x,y) + I^{(1)}(x,y) + b(x,y) + O_{offset}) >> shift$$

These values are chosen so that the multipliers do not exceed 15 bits in the BDOF process, and the maximum bit-width of the intermediate parameter is kept within 32 bits. In order to obtain the gradient value, it is necessary to generate several prediction samples $I^{(k)}(i,j)$ in the list k, $(k = 0, 1)$, outside the current CU boundaries. BDOF in VVC uses extended one row/column around the CU boundary as depicted in Figure 7.18. To control the computational complexity of generating out-of-boundary prediction samples, the extended area prediction samples (white positions) are generated by

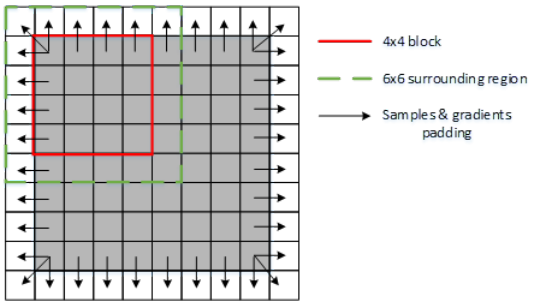


FIGURE 7.18: Extended CU region used in BDOF [12].

taking reference samples at the nearby integer positions (using the Floor() operation on the coordinates). This is done directly without interpolation, and the traditional 8-tap motion compensation interpolation filter is used to generate the prediction samples in the CU (gray position). These extended sample values are only used for gradient calculation. For the remaining steps of the BDOF process, if samples and gradient values outside the CU boundary are needed, they are padded by their nearest neighbors. If the width and/or height of the CU exceeds 16 luma samples, it is divided into sub-blocks with a width and/or height of 16 luma samples, and the sub-block boundaries are considered as a CU boundaries for the BDOF process. The maximum unit size in BDOF process is limited to 16×16. For each sub-block, the BDOF process can be skipped. If the SAD between the initial prediction samples L0 and L1 is less than the threshold, the BDOF process is not applied to the sub-block. The threshold is set to (8 * W * (H >> 1), where W is the width of the sub-block and H is its height. The SAD calculated between the initial L0 and L1 samples is reused, in order to avoid the added complexity caused by repeating the same computation. If BCW is activated for the current block, this means in that the weight is not uniform, then the bidirectional optical flow is disabled. Similarly, if the WP of the current block is activated, i.e., $luma\_weight\_lx\_flag$ for one of the two reference pictures is 1, then BDOF is also disabled. If the CU is coded in symmetrical MVD or CIIP modes, BDOF will also be disabled.
Like some other tools, BDOF can be applied only in Random Access configuration. The performance of this tool is evaluated and summarized in Table 7.16.

7.1.2.10 Decoder side Motion Vector Refinement (DMVR)

In VVC, a Bilateral-Matching (BM) based decoder side motion vector refinement is used to improve the accuracy of the MVs in merge mode. In the bi-prediction procedure, a refined MV is searched around the initial MVs in the reference picture lists $L0$ and $L1$. The distortion between the two candidate blocks in the reference picture list $L0$ and $L1$ is calculated by the BM technique. The SAD between the red blocks depending on each MV candidate around the original MV is computed, as shown in Figure 7.19. The refined MV is the MV candidate with the lowest SAD, and it is utilized to create the bi-predicted signal. The DMVR can be used in VVC for CUs that are coded with the following modes and features:

- Merge mode at the CU level with bi-prediction MV.
- With respect to the current picture, one reference picture is in the past and another is in the future.
- The distances (i.e., the POC difference) between two reference pictures and the current picture are the same.

Class	Sequence	Random Access	
		BDR-YUV	ET
A1	Tango	0,61%	93%
	Food Market	0,43%	96%
	Campfire	0,01%	98%
A2	Cat Robot	0,76%	99%
	Daylight Road	1,32%	101%
	Park Running	0,43%	97%
B	Market Place	0,57%	94%
	Ritual Dance	0,57%	90%
	Cactus	0,57%	98%
	Basketball Drive	0,61%	100%
	BQ-Terrace	0,81%	100%
C	Basketball Drill	0,49%	98%
	BQ-Mall	1,25%	105%
	Party Scene	0,83%	98%
	Race Horses	0,40%	103%
Overall	BDR-Y	0,76%	
	BDR-U	0,31%	
	BDR-V	0,27%	
Average	BDR-YUV	**0,64%**	**98%**

TABLE 7.16: Results of VTM BDOF tool "off" on various JVET CTC sequences in different configurations.

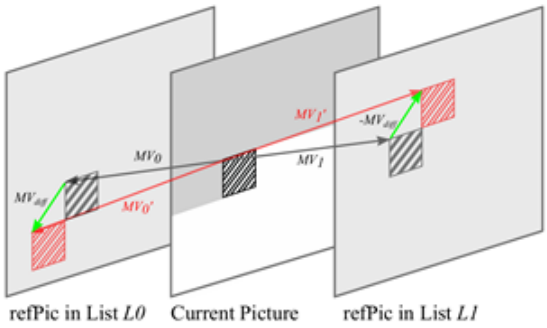


FIGURE 7.19: Decoding side motion vector refinement [13].

- WP is not enabled for the current block.
- Both reference photos are short-term reference pictures.
- CU includes more than 64 luma samples.
- CU height and CU width are more than or equal to 8 luma samples.
- BCW weight index shows equal weight.
- CIIP mode is not used for the current block.

The refined MV derived from the DMVR method is utilized to provide inter prediction samples as well as temporal motion vector prediction for future picture coding. While the original MV is used in the deblocking process, it is also used to predict spatial motion vectors for future CU coding. The following sub-clauses discuss DMVR's extra features.

The search points in DVMR are centered around the starting MV, and the MV offset follows the MV difference mirroring rule. In other words, all points verified by DMVR, indicated by a candidate MV pair $(MV0, MV1)$, must satisfy the following two equations:

$$MV0' = MV0 + MV\_offset$$
$$MV1' = MV1 - MV\_offset$$

In one of the reference pictures, MV offset depicts the refinement offset between the initial MV and the refined MV. The range of the refinement search is two integer luma samples from the starting MV. The integer sample offset search stage and the fractional sample refinement stage are included in the search.

For integer sample offset searching, a full search of 25 points is used. First, the SAD of the initial MV pair is computed. The integer sample stage of DMVR is terminated if the SAD of the initial MV pair is less than a threshold. Otherwise, SADs are computed and verified for the remaining 24 spots in raster scanning order. The point with the lowest SAD is chosen as the result of the integer sample offset searching stage. To mitigate the consequence of DMVR refinement uncertainty, it is recommended to prefer the original MV during the DMVR process. The SAD between the reference blocks referred by the initial MV candidates is decreased by 1/4 of the SAD value.

After the integer sample search, the fractional sample refinement is performed. Instead of conducting an extra search using SAD comparison, the fractional sample refinement is calculated using a parametric error surface equation to reduce computational complexity. The fractional sample refinement is called conditionally based on the results of the integer sample search stage. When the integer sample search stage is completed with the center having the least SAD in either the first or second iteration search, the fractional sample refinement is used.

In parametric error surface based sub-pixel offsets estimation, the center position cost and the costs at four neighboring positions from the center are used to fit a 2-D parabolic error surface equation of the following form

$$E(x, y) = A(x - x_{\min})^2 + B(y - y_{\min})^2 + C$$

where $(x_{\min}, y_{\min})$ is the fractional position with the lowest cost and C corresponds to the lowest cost value. The $(x_{\min}, y_{\min})$ is obtained by solving the preceding equations using the cost value of the five

Class	Sequence	Random Access	
		BDR	ET
A1	Tango	1,76%	100%
	Food Market	0,85%	99%
	Campfire	0,03%	99%
A2	Cat Robot	1,52%	101%
	Daylight Road	2,27%	101%
	Park Running	0,59%	100%
B	Market Place	1,01%	100%
	Ritual Dance	0,83%	100%
	Cactus	0,70%	100%
	Basketball Drive	0,72%	100%
	BQ-Terrace	0,39%	100%
C	Basketball Drill	0,75%	100%
	BQ-Mall	1,02%	100%
	Party Scene	0,43%	100%
	Race Horses	0,58%	100%
Overall	BDR-Y	0,83%	
	BDR-U	1,08%	
	BDR-V	1,09%	
Average	BDR-YUV	**0,90%**	**100%**

TABLE 7.17: Results of VTM DMVR tool "off" test on various JVET CTC sequences in RA configurations.

search points:

$$x_{\min} = \big(E(-1,0) - E(1,0)\big)/\big(2(E(-1,0) + E(1,0) - 2E(0,0))\big)$$
$$y_{\min} = \big(E(0,-1) - E(0,1)\big)/\big(2(E(0,-1) + E(0,1) - 2E(0,0))\big)$$

Because all cost values are positive and the smallest value is $E(0,0)$, the values of $x_{\min}$ and $y_{\min}$ are automatically restricted to be between -8 and 8. In VVC, this corresponds to a half pel offset with a 1/16th-pel MV accuracy. the sub-pixel accurate refinement delta MV is obtained by adding the computed fractional $(x_{\min}, y_{\min})$ to the integer distance refinement MV.

Bilinear-interpolation and sample padding: The MVs in VVC have a resolution of 1/16 luma samples. An 8-tap interpolation filter is used to interpolate the samples at the fractional location. Because the search points in DMVR are surrounded by the initial fractional-pel MV with an integer sample offset, the samples of those fractional positions must be interpolated for the DMVR search process. The bi-linear interpolation filter is used to create fractional samples for the searching process in DMVR in order to minimize calculation complexity. Another significant impact of utilizing a bi-linear filter is that the DVMR does not access additional reference samples with a 2-sample search range as compared to the standard motion compensation algorithm. After achieving the refined MV using the DMVR search process, the standard 8-tap interpolation filter is used to generate the final prediction. To avoid accessing more reference samples during the normal MC process, the samples that are not required for the interpolation process based on the original MV but are required for the interpolation process based on the refined MV are padded from those available samples.

Maximum DMVR processing unit: When the width and/or height of a CU exceed 16 luma samples, it is further partitioned into sub-blocks of the same width and/or height. The largest unit size for the DMVR search process is 16×16.

The DMVR is uniquely evaluated in Random Access configuration (Table 7.17) since it cannot be applied in the other configurations.

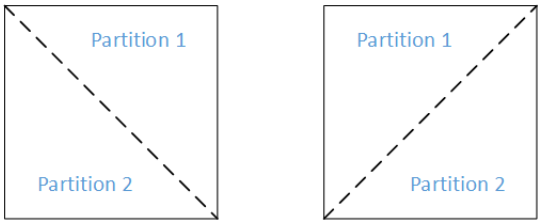


FIGURE 7.20: Triangle partition based inter prediction [14].

7.1.2.11 Triangle Partition Mode for inter prediction (TPM)

This tool has been known afterwards as Geometric Partitioning Mode (GPM) since the partitions changed from being restrictively triangular to have more generic geometrical forms.

For inter prediction, VVC has a triangular partition mode. The triangle division mode is only available for CUs 8x8 or bigger. As one type of merge mode, the triangular partition mode is signaled by a CU-level flag; other merge modes include the normal merge mode, the MMVD mode, the CIIP mode, and the sub-block merge mode.

When utilizing this method, a CU is evenly divided into two triangle-shaped partitions using either the diagonal split or the anti-diagonal split, see Figure 7.20). Each triangular partition in the CU is inter-predicted using its own motion; only uni-prediction is permitted for each partition, which means that each partition has a single motion vector and a single reference index. The uni-prediction motion constraint is used to ensure that, as with conventional bi-prediction, only two motion compensated predictions are required for each CU. The technique outlined in the next sub-section is used to calculate the uni-prediction motion for each division.

A flag dedicated for the triangle partition direction (diagonal or anti-diagonal) and two merge indices (one for each partition) are transmitted. At the slice level, the maximum TPM candidate size is explicitly signaled and defines syntax binarization for TPM merge indices. Following the prediction of each triangle partition, the sample values along the diagonal or anti-diagonal edge are adjusted using a blending mechanism with adaptive weights. This is the prediction signal for the entire CU, and the transform and quantization processes are applied to the entire CU as they are in other prediction modes.

When the signalled triangle mode is equal to 1, the cu_sbt_flag is assumed to be 0 without signalling, indicating that the triangle partition mode is not utilized in combination with SBT.

Uni-prediction candidate list construction: The merge candidate list generated by the expanded merge prediction procedure is used to create the uni-prediction candidate list. In the triangular uni-prediction candidate list, denote n as the index of the uni-prediction motion. The n-th uni-prediction motion vector of the TPM is the LX motion vector of the n-th extended merge candidate, with X equal to the parity of n. In Figure 7.21, these motion vectors are denoted by the letter "x" in the table, when the LX motion vector of the n-th extended merge candidate does not exist, the L(1-X) motion vector of the same candidate is utilized as the uni-prediction motion vector for triangle partition mode.

Blending along the triangle partition edge: After each triangle split is predicted using its own motion, the two prediction signals are blended to generate samples along the diagonal or anti-diagonal edge. In the blending process, the following weights are used:

7/8, 6/8, 5/8, 4/8, 3/8, 2/8, 1/8 for luma and 6/8, 4/8, 2/8 for chroma, as shown in Figure 7.22.

Motion field storage: The following procedure is used to produce the motion vectors of a CU coded in Triangle Partition Mode:

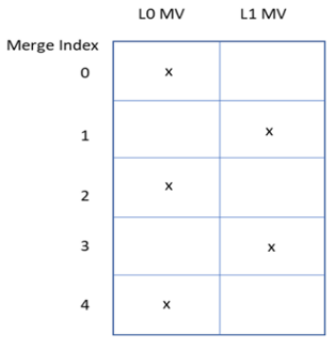


FIGURE 7.21: Uni-prediction MV selection for TPM.

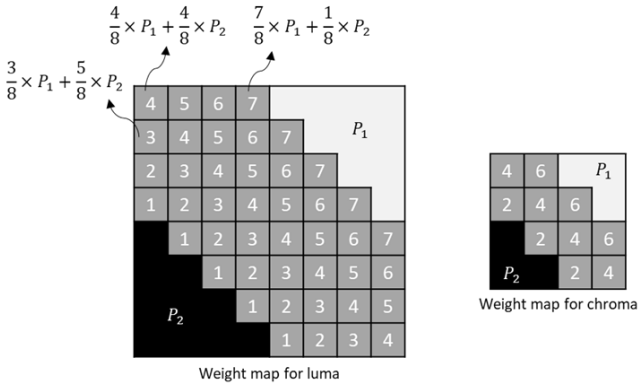


FIGURE 7.22: Weights used in the blending process [14].

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	0,49%	95%		
	Food Market	0,15%	97%		
	Campfire	0,16%	96%		
A2	Cat Robot	0,45%	97%		
	Daylight Road	0,29%	100%		
	Park Running	0,40%	97%		
B	Market Place	0,33%	91%	0,70%	96%
	Ritual Dance	0,32%	86%	0,62%	99%
	Cactus	0,41%	98%	0,99%	98%
	Basketball Drive	0,28%	97%	0,67%	96 %
	BQ-Terrace	0,22%	99%	0,38%	96%
C	Basketball Drill	0,83%	85%	1,86%	97%
	BQ-Mall	1,27%	97%	1,68%	97%
	Party Scene	0,45%	89%	0,76%	97%
	Race Horses	0,75%	98%	0,98%	97%
Overall	BDR-Y	0,38%		0,88%	
	BDR-U	0,64%		1,13%	
	BDR-V	0,68%		1,28%	
Average	BDR-YUV	**0,45%**	**95%**	**0,96%**	**97%**

TABLE 7.18: Results of VTM TPM tool "off" test on various JVET CTC sequences in different configurations.

1. If Mv1 and Mv2 come from distinct reference picture lists (one from L0 and the other from L1), they are simply added together to give the bi-prediction motion vector.

2. If Mv1 and Mv2 are both from the same list, only uni-prediction motion Mv2 is saved.

7.1.2.12 Motion field storage

Quarter-luma-sample precision is the maximum precision of explicitly signaled motion vectors in VVC. Motion vectors are derived with 1/16th-luma-sample accuracy in some inter prediction modes, such as the affine mode. Motion compensated prediction is performed with 1/16th-sample precision. All motion vectors are recorded with 1/16th-luma-sample precision in the internal motion field storage. Motion field compression is performed at 8×8 granularity for temporal motion field storage in TMVP and ATVMP, contrasting the 16×16 granularity in HEVC.

7.1.2.13 Miscellaneous inter prediction aspects

To save memory bandwidth, the inter-coded 4×4 size CU is not alowed in VVC. Only uni-directional mode is permitted for inter-coded 4×8/8×4 CU. The bi-directional motion information from merge mode is converted to uni-directional motion information by conserving just the list 0 motion information.

7.1.3 Transform and Quantization

7.1.3.1 Sub-Block Transform (SBT)

VTM introduces the concept of sub-block transform for an inter-predicted CU . In this transform mode, only sub-part of the residual blocks is coded for CU. If CU is inter-predicted with cu_cbf equal to 1, cu_sbt_flag may be flagged to indicate whether the entire residual block or a sub-part of the residual block

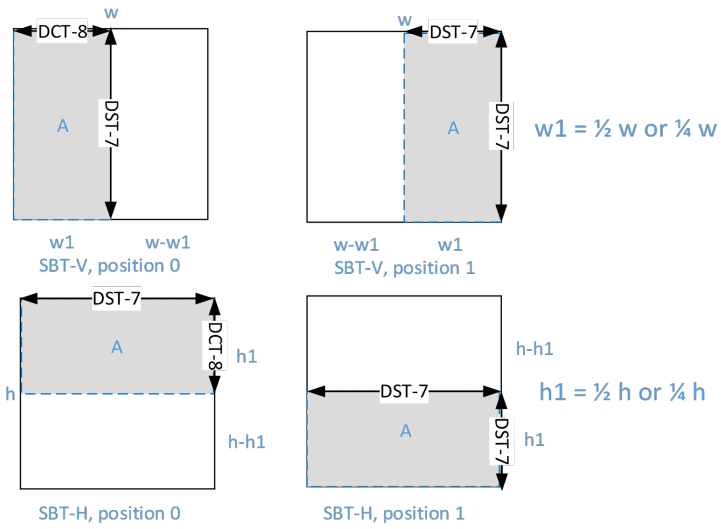


FIGURE 7.23: SBT position, type and transform type [15].

is encoded. In the former case, the inter MTS information is further parsed to determine the transform type; in the latter case, a part of the residual block is encoded using an inferred adaptive transform, and the other part of the residual block is set to zero.

When SBT is used in an inter-coded CU, the SBT type and position information are transmitted in the bitstream. There are two types of SBT and two SBT positions, as shown in Figure 7.23. For SBT-V (or SBT-H), the TU width (or height) may equal to half of the CU width (or height) or 1/4 of the CU width (or height), resulting in 2:2 split or 1:3/3:1 split. The 2:2 split is like a Binary Tree (BT) split while the 1:3/3:1 split is like an Asymmetric Binary Tree (ABT) split. In the ABT splitting, only a small part of the area contains non-zero residual. If one dimension of a CU is 8 in luma samples, the 1:3/3:1 split along that dimension is disallowed. A CU has up to 8 SBT modes.

The position-dependent transform kernel selection is used in Luma transform blocks in SBT-V and SBT-H (Chroma-TB always with DCT2). The two positions of SBT-H and SBT-V are associated with different kernel transforms as detailed in Figure 7.23. For example, the horizontal and vertical transforms for SBT-V position 0 are DCT8 and DST7, respectively. If one side of the residual TU is bigger than 32, the transform is DCT2 for both dimensions. Therefore, the sub-block transforms jointly determine the tiling of the TU, CBF, and horizontal and vertical core transform type of the residual block. A variable maxSbtSize is reported in the SPS to specify the maximum CU size that can use SBT. In the VTM7 reference software for HD and 4K sequences, the encoder sets maxSbtSize to 64; for other sequences with lower resolution, maxSbtSize is set to 32. SBT is not applicable to CUs coded in CIIP mode or TPM(also known as GPM).

7.1.3.2 Multiple Transform Selection (MTS) for core transform

A Multiple Transform Selection (MTS) technique is introduced to residual coding both inter and intra coded blocks, in addition to DCT-II, which is used in HEVC. It uses a number of DCT8/DST7 transforms. The basic functions of the selected DST-VII/DCT-VIII are shown in Table 7.21.

Class	Sequence	Random Access		Low Delay B	
		BDR	ET	BDR	ET
A1	Tango	0,16%	97%		
	Food Market	0,07%	96%		
	Campfire	0,16%	96%		
A2	Cat Robot	0,12%	95%		
	Daylight Road	0,44%	94%		
	Park Running	0,71%	91%		
B	Market Place	0,30%	95%	0,48%	91%
	Ritual Dance	0,32%	96%	0,42%	94%
	Cactus	0,26%	95%	0,45%	92%
	Basketball Drive	0,29%	95%	0,41%	92%
	BQ-Terrace	0,45%	92%	0,82%	88%
C	Basketball Drill	0,12%	97%	0,19%	95%
	BQ-Mall	0,37%	96%	0,58%	92%
	Party Scene	0,35%	95%	0,48%	91%
	Race Horses	0,35%	94%	0,33%	92%
Overall	BDR-Y	0,40%		0,65%	
	BDR-U	-0,03%		-0,19%	
	BDR-V	-0,01%		0,01%	
Average	BDR-YUV	**0,30%**	**95%**	**0,46%**	**92%**

TABLE 7.19: Results of VTM SBT tool "off" test on various JVET CTC sequences in different configurations.

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	2,28%	111%	1,34%	95%		
	Food Market	1,95%	113%	1,04%	94%		
	Campfire	0,51%	104%	1,65%	92%		
A2	Cat Robot	1,20%	110%	0,95%	95%		
	Daylight Road	0,60%	112%	0,42%	96%		
	Park Running	0,51%	102%	0,37%	91%		
B	Market Place	0,19%	112%	0,02%	96%	-0,12%	92%
	Ritual Dance	1,00%	112%	0,44%	95%	0,53%	91%
	Cactus	0,91%	110%	0,74%	94%	0,35%	90%
	Basketball Drive	1,02%	114%	0,91%	95%	0,59%	90%
	BQ-Terrace	1,05%	114%	0,92%	95%	-0,15%	93%
C	Basketball Drill	3,12%	112%	1,64%	93%	0,94%	88%
	BQ-Mall	0,50%	114%	-0,05%	95%	0,01%	91%
	Party Scene	0,27%	113%	-0,05%	94%	-0,11%	90%
	Race Horses	1,02%	112%	0,55%	94%	0,26%	89%
Overall	BDR-Y	1,16%		0,88%		0,40%	
	BDR-U	0,63%		0,02%		-0,10%	
	BDR-V	1,01%		0,49%		-0,23%	
Average	BDR-YUV	**1,08%**	**111%**	**0,73%**	**94%**	**0,25%**	**90%**

TABLE 7.20: Results of VTM LFNST tool "off" test on various JVET CTC sequences in different configurations.

Transform Type	Basis function $T_i(j), i, j = 0, 1, \ldots, N-1$
DCT-II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos(\frac{\pi \cdot i \cdot (2j+1)}{2N})$ where, $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}}, & i = 0 \\ 1, & i \neq 0 \end{cases}$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{(2N+1)}} \cdot \cos(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2})$
DST-VII	$T_i(j) = \sqrt{\frac{4}{(2N+1)}} \cdot \sin(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1})$

TABLE 7.21: Transform basis functions of DCT-II/VIII and DST-VII for N-point input.

The transform matrices are quantized more precisely than the transform matrices in HEVC in order to maintain the orthogonality of the transform matrix. After the horizontal and vertical transforms, all the coefficients should have 10-bit to preserve the intermediate values of the converted coefficients inside the 16-bit range.

Separate enabling flags for intra and inter are defined at the SPS level to handle the MTS scheme. A CU level flag is signaled when MTS is enabled at SPS to indicate if MTS is used or not. MTS is solely used for luma in this case. the MTS CU level flag is set When the following criteria are met:

- Width and height are both less than or equal to 32.
- The CBF flag equals one.

DCT2 is applied in both directions if the MTS_CU_flag is set to zero. If the MTS CU flag is set to one, two extra flags are signaled to indicate the transform type in the horizontal and vertical directions. Table 7.21 shows the transform and signaling mapping table. By eliminating the intra-mode and block-shape dependencies, the transform selection for ISP and implicit MTS has been unified. Only DST7 is applied for both horizontal and vertical transform cores in case the current block is in ISP mode or the current block is intra block and both intra and inter explicit MTS are enabled. Primary transform cores with an 8-bit precision are used for transform matrix precision. Therefore, all of the transform cores used in HEVC, including 4-point DCT-2 and DST-7, 8-point, 16-point, and 32-point DCT-2, are maintained the same. Other transform cores that use 8-bit primary transform cores include 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7, and DCT-8.

To reduce the complexity of large size DST-7 and DCT-8, high frequency transform coefficients are zeroed out for the DST-7 and DCT-8 blocks with size (width or height, or both) equal to 32. Only the coefficients within the 16×16 lower-frequency region are retained. The residual of a block can be coded using transform skip mode, like in HEVC. When the CU level MTS_CU_flag is not equal to zero, the transform skip flag is not signaled to avoid syntax coding redundancy. The transform skip block size constraint is the same as for MTS in JEM4, indicating that transform skip is applicable for a CU when both block width and height are equal to or less than 32. When LFNST or MIP is enabled for the current CU, the implicit MTS transform is set to DCT2. When MTS is enabled for inter coded blocks, the implicit MTS can still be enabled.

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/inter	
			Horizontal	Vertical
0			DCT2	
1	0	0	DST7	DST7
	0	1	DCT8	DST7
	1	0	DST7	DCT8
	1	1	DCT8	DCT8

TABLE 7.22: Transform and signalling mapping table.

Table 7.23 shows the performance of MTS in different configurations.

7.1.3.3 Low-Frequency Non-Separable Transform (LFNST)

As illustrated in Figure 7.24, LFNST, also known as reduced secondary transform, is applied between forward primary transform and quantization (at the encoder) and between de-quantization and inverse primary transform (at the decoder). Depending on the block size, LFNST uses a 4x4 non-separable transform or an 8x8 non-separable transform. For small blocks (i.e., min(width, height) < 8) and for bigger blocks (i.e., min(width, height) > 4), 4x4 LFNST is used, and 8x8 LFNST is used.

The following is a description of how to apply a non-separable transform, which is utilized in LFNST. The 4×4 input block X is used to apply 4×4 LFNST.

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix} \text{ is first represented as a vector } \vec{X}. \tag{7.13}$$

with $\vec{X} = [X_{00}, X_{01}, X_{02}, X_{03}, X_{10}, X_{11}, X_{12}, X_{13}, X_{20}, X_{21}, X_{22}, X_{23}, X_{30}, X_{31}, X_{32}, X_{33}]^T$.

The non-separable transform is calculated as $\vec{F} = \mathrm{T} \cdot \vec{X}$, where $\vec{F}$ is the transform coefficient vector and T is a 16×16 transform matrix. After that, the 16×1 coefficient vector $\vec{F}$ is reorganized into a 4×4 block using the scanning order for that block (horizontal, vertical or diagonal). The coefficients with smaller index are placed with the smaller scanning index in the 4×4 coefficient block.

Reduced non-Separable Transform: Low-Frequency Non-Separable Transform (LFNST) is applied in a single pass without multiple iterations using a direct matrix multiplication approach. However, to reduce computational complexity and memory space required to store the transform coefficients, the non-separable transform matrix size must be reduced. LFNST employs the reduced non-separable transform (or RST) technique. The main idea of the reduced non-separable transform is to map an N (N is typically equal to 64 for 8×8 NSST) dimensional vector to an R dimensional vector in another space, where N/R ($R < N$) is the reduction factor. Thus, instead of an $N \times N$ matrix, the RST matrix is transformed into an $R \times N$ matrix as follows:

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	1,21%	83%	0,52%	92%		
	Food Market	1,85%	88%	0,81%	95%		
	Campfire	0,82%	84%	0,63%	90%		
A2	Cat Robot	1,28%	85%	0,57%	94%		
	Daylight Road	1,26%	80%	0,48%	93%		
	Park Running	1,26%	84%	0,81%	89%		
B	Market Place	2,05%	80%	1,14%	88%	0,53%	96%
	Ritual Dance	1,67%	80%	1,20%	84%	1,07%	97%
	Cactus	1,44%	81%	0,86%	91%	0,31%	99%
	Basketball Drive	0,83%	85%	0,54%	94 %	0,47%	96%
	BQ-Terrace	0,76%	81%	0,69%	92%	0,24%	99%
C	Basketball Drill	0,39%	72%	0,50%	80%	0,45%	98%
	BQ-Mall	1,03%	73%	0,62%	91%	0,28%	98%
	Party Scene	0,51%	83%	0,52%	90%	0,12%	97%
	Race Horses	0,84%	77%	0,45%	96%	0,34%	97%
Overall	BDR-Y	1,19%		0,70%		0,57%	
	BDR-U	0,97%		0,59%		-0,02%	
	BDR-V	1,04%		0,72%		0,02%	
Average	BDR-YUV	**1,15%**	**81%**	**0,69%**	**91%**	**0,42%**	**97%**

TABLE 7.23: Performance results of VTM MTS tool "off" test on various JVET CTC sequences in different configurations.

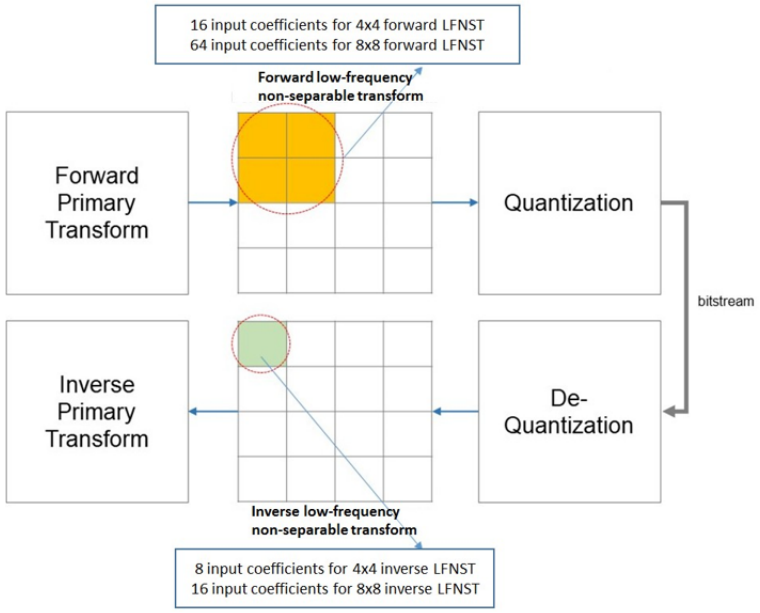


FIGURE 7.24: Low-Frequency Non-Separable Transform process [16].

IntraPredMode	Tr. set index
IntraPredMode < 0	1
$0 \leq$ IntraPredMode ≤ 1	0
$2 \leq$ IntraPredMode ≤ 12	1
$13 \leq$ IntraPredMode ≤ 23	2
$24 \leq$ IntraPredMode ≤ 44	3
$45 \leq$ IntraPredMode ≤ 55	2
$56 \leq$ IntraPredMode ≤ 80	1
$81 \leq$ IntraPredMode ≤ 83	0

TABLE 7.24: Transform selection table.

$$T_{RxN} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & ... & t_{1N} \\ t_{21} & t_{22} & t_{23} & ... & t_{2N} \\ & \vdots & & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & ... & t_{RN} \end{bmatrix}. \tag{7.14}$$

The R rows of the transform correspond to the R bases of the N dimensional space. The reduction factor for 8×8 LFNST is 4, and the 64×64 direct matrix, which is the size of a conventional 8×8 non-separable transform matrix, is reduced to 16×48 direct matrix. Hence, the 48×16 inverse RST matrix is used to generate core (primary) transform coefficients in 8×8 top-left areas on the decoder side. When using 16×48 matrices instead of 16×64 with the same transform set setup, each of which accepts 48 input data from three 4×4 blocks in a top-left 8×8 block excluding the right-bottom 4×4 block. Memory usage for storing all LFNST matrices is decreased from 10KB to 8KB with an acceptable speed loss thanks to the reduced dimension. LFNST is only applicable if all coefficients outside the first coefficient sub-group are non-significant, in order to decrease complexity. Consequently, when LFNST is used, all primary-only transform coefficients must be zero. This allows the LFNST index signaling to be conditioned on the last-significant position, avoiding the additional coefficient scanning required in the current LFNST architecture to check for significant coefficients at specific positions only. The non-separable transforms for 4x4 and 8x8 blocks are limited to 8x16 and 8x48 transforms, respectively, due to LFNST's worst-case handling (in terms of multiplications per pixel). When LFNST is used in those cases, the last-significant scan position must be less than 8 for other sizes less than 16. The suggested limitation indicates that the LFNST is now applied just once, on the top-left 4×4 region, for blocks with a $4 \times N$ and $N \times 4$ shapes and $N > 8$. The number of operations required for the primary transforms is reduced in such circumstances since all primary-only coefficients are zero when LFNST is applied. From encoder perspective, when LFNST transforms are tested, the quantization of coefficients is significantly simplified. The first 16 coefficients (in scan order) must be subjected to a rate-distortion optimized quantization, while the remaining coefficients must be zero.

LFNST transform selection: LFNST employs a total of four transform sets, each with two non-separable transform matrices (kernels). As indicated in Table 7.24, the mapping from the intra prediction mode to the transform set is pre-defined. Transform set 0 is selected for the current chroma block if one of three CCLM modes (INTRA_LT_CCLM, INTRA_T_CCLM, or INTRA_L_CCLM) is used for the current block ($81 \leq$ predModeIntra ≤ 83). The explicitly signaled LFNST index further specifies the selected non-separable secondary transform candidate for each transform set. After the transform coefficients, the index is signalled in a bit-stream once per Intra CU.

LFNST index signaling and interaction with other tools: LFNST index coding depends on

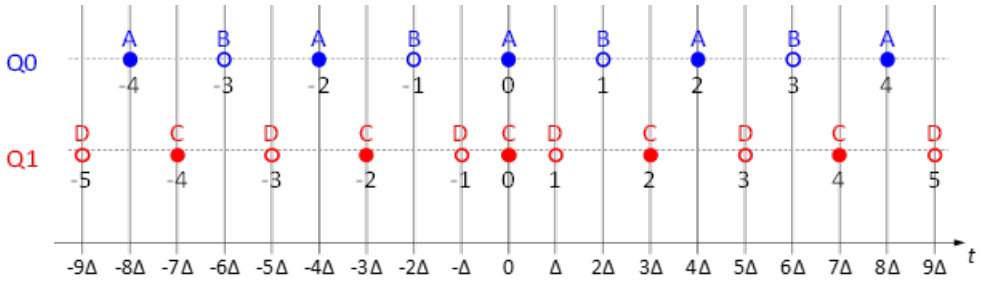


FIGURE 7.25: Illustration of the two scalar quantizers used in the proposed approach of dependent quantization [17].

the position of the last significant coefficient since it is only applicable if all coefficients outside the first coefficient sub-group are non-significant. Moreover, the LFNST index is context coded, but it does not depend on intra prediction mode, and only the first bin is context coded. LFNST is used for Luma and Chroma intra CU in both intra and inter slices. LFNST indices for Luma and Chroma are signalled independently if a dual tree is enabled. A single LFNST index is signaled and used for both Luma and Chroma for inter slice where the dual tree is deactivated.

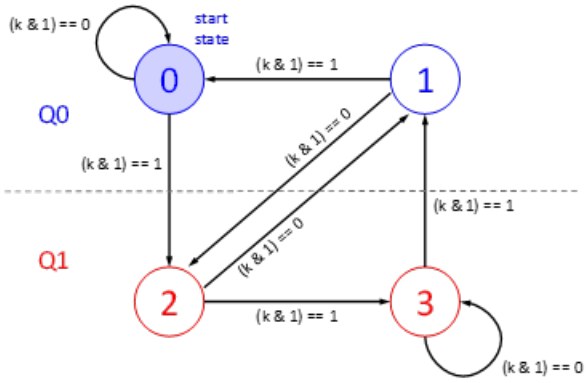
Since the performance improvement is negligible even with RST applied to every possible partition block when ISP mode is selected, LFNST is disabled and the RST index is not signalled. Furthermore, encoding complexity might be reduced by deactivating RST for ISP-predicted residual. When MIP mode is selected, LFNST is automatically deactivated, and the index is not sent.

Due to the existing maximum transform size constraint (64×64), a large CU bigger than 64×64 is implicitly split (TU tiling), an LFNST index search might increase data buffering by four times for a given number of decode pipeline steps. As a result, the greatest size that LFNST may be is limited to 64×64. It is worth noting that LFNST is only available with DCT2.

7.1.3.4 Dependent quantization

The same HEVC scalar quantization is used with a novel concept known as dependent scalar quantization. The set of admissible reconstruction values for a transform coefficient is determined by the values of the transform coefficient levels preceding the current transform coefficient level in reconstruction order. This is mainly what defines the dependent scalar quantization approach. The major effect of this method is that the admissible reconstruction vectors are packed denser in the N-dimensional vector space than with conventional independent scalar quantization, which is applied in HEVC (N represents the number of transform coefficients in a transform block). This means, the average distortion between an input vector and the nearest reconstruction vector is reduced for a certain average number of admissible reconstruction vectors per N-dimensional unit volume. The dependent scalar quantization technique is completed by (a) establishing two scalar quantizers with different reconstruction levels and (b) defining a process for switching between these quantizers.

Figure 7.25 depicts the two scalar quantizers, denoted by Q0 and Q1. A quantization step size Δ determines where the available reconstruction levels are located. The bitstream does not specify which scalar quantizer is used (Q0 or Q1). Instead, the parities of the transform coefficient levels that come before the current transform coefficient in coding/reconstruction order define the quantizer used for that transform coefficient.



current state	next state for ...	
	(k & 1) == 0	(k & 1) == 1
0	0	2
1	2	0
2	1	3
3	3	1

FIGURE 7.26: State transition and quantizer selection for the proposed dependent quantization [17].

A state machine with four states is utilized to switch between the two scalar quantizers (Q0 and Q1), as shown in Figure 7.26. The state can have one of four possible values: 0, 1, 2, or 3. It is uniquely determined by the parities of the transform coefficient levels preceding the current transform coefficient in coding/reconstruction order. The state of a transform block is set to 0 in the beginning of the inverse quantization. The transform coefficients are reconstructed in the scan order (i.e., in the same order they are entropy decoded). The state is updated after a current transform coefficient is reconstructed, as illustrated in Figure 7.26, where k represents the level of the transform coefficient. The default and user-defined scaling matrices can also be signaled. For all TB sizes, the DEFAULT mode scaling matrices are all flat, with elements equal to 16. The scaling matrices for IBC and intra coding modes are currently the same. Thus, the number of MatrixType and MatrixType_DC for USER_DEFINED matrices are updated as follows:

- **MatrixType: 30 = 2** (2 for intra&IBC/inter) × 3 (Y/Cb/Cr components) × **5** (square TB size: from 4×4 to 64×64 for luma, from 2×2 to 32×32 for chroma).
- **MatrixType_DC: 14 = 2** (2 for intra&IBC/inter × 1 for Y component) × **3** (TB size: 16×16, 32×32, 64×64) + **4** (2 for intra&IBC/inter × 2 for Cb/Cr components) × **2** (TB size: 16×16, 32×32).

The DC values for the following scaling matrices are separately coded: 16×16, 32×32, and 64×64. All elements in one scaling matrix are signaled for TBs smaller than 8×8. Only 64 elements in one 8×8 scaling matrix are signaled as a base scaling matrix if the TBs have a size more than or equal to 8×8. The 8×8 base scaling matrix is up-sampled (by duplication of elements) to the corresponding square size (i.e., 16×16, 32×32, 64×64) in order to produce square matrices of size greater than 8×8. When the high frequency coefficients for the 64-point transform are zeroed out, corresponding high frequencies of the scaling matrices are likewise zeroed out. If the TB's width or height is larger than or equal to 32, only the left or top half of the coefficients are retained, and the remaining coefficients are set to zero. Furthermore, since the bottom-right 4×4 elements are never utilized, The number of elements signalled for the 64×64 scaling matrix is decreased from 8×8 to three 4×4 sub-matrices.

DQ shows interesting performance presented in Table 7.25.

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	1,48%	106%	0,55%	111%		
	Food Market	1,26%	102%	1,25%	113%		
	Campfire	2,39%	100%	1,90%	102%		
A2	Cat Robot	1,80%	104%	1,21%	108%		
	Daylight Road	2,42%	101%	0,70%	103%		
	Park Running	2,89%	99%	2,31%	102%		
B	Market Place	1,50%	104%	1,47%	103%	2,09%	116%
	Ritual Dance	0,96%	101%	0,88%	107%	1,10%	118%
	Cactus	1,38%	98%	1,46%	106%	1,34%	113%
	Basketball Drive	0,93%	100%	0,84%	105 %	1,18%	115%
	BQ-Terrace	0,33%	95%	1,59%	100%	1,92%	112%
C	Basketball Drill	0,77%	94%	0,81%	101%	0,77%	110%
	BQ-Mall	0,85%	94%	0,85%	101%	0,99%	111%
	Party Scene	0,90%	91%	1,27%	98%	1,37%	107%
	Race Horses	1,43%	94%	1,25%	100%	1,86%	106%
Overall	BDR-Y	2,13%		1,76%		1,75%	
	BDR-U	-0,66%		-0,26%		0,56%	
	BDR-V	-0,74%		-0,51%		0,16%	
Average	BDR-YUV	**1,42%**	**99%**	**1,22%**	**104%**	**1,40%**	**112%**

TABLE 7.25: Performance results of VTM DQ tool "off" test on various JVET CTC sequences in different configurations.

7.1.3.5 Joint Coding of Chroma Residuals (JCCR)

VVC 6 supports a mode that allows the chroma residuals to be coded together. A TU-level flag tu_joint_cbcr_residual_flag indicates the activation of a joint chroma coding mode, and the selected mode is implicitly indicated by the chroma CBFs. If either or both chroma CBFs for a TU are equal to 1, the flag tu_joint_cbcr_residual_flag is set. To distinguish between the typical chroma QP offset values signaled for regular chroma residual coding mode and the joint chroma residual coding mode, chroma QP offset values for the joint chroma residual coding mode are signaled in the PPS and slice header. The chroma QP values for those blocks coded using the joint chroma residual coding mode are derived using these chroma QP offset values. When a corresponding joint chroma coding mode (mode 2 in Table 7.26) is active for a TU, this chroma QP offset is added to the applied luma-derived chroma QP during quantization and decoding of that TU. The chroma QPs for the other modes (modes 1 and 3 in Table 7.26) are calculated in the same way as for conventional Cb or Cr blocks. Table 7.26 depicts the chroma residuals (resCb and resCr) reconstruction process from transmitted transform blocks. When this mode is enabled, a single joint chroma residual block (resJointC[x][y] in Table 7.26) is signalled, and residual blocks, one for Cb (resCb) and the other for Cr (resCr) are calculated using parameters such as tu_cbf_cb, tu_cbf_cr, and CSign, which is a sign value given in the slice header. The joint chroma components are derived on the encoder side, as detailed below. The encoder generates resJointC$\{1, 2\}$ depending on the mode (as indicated in Table 7.26).

- If mode is equal to 2 (single residual with reconstruction Cb = C, Cr = CSign$*$C), the joint residual is determined according to resJointC[x][y] = (resCb[x][y] + CSign $*$ resCr[x][y])/2;
- If mode is equal to 1 (single residual with reconstruction Cb = C, Cr = (CSign $*$ C)/2), the joint residual is determined according to resJointC[x][y] = (4 $*$ resCb[x][y] + 2 $*$ CSign $*$ resCr[x][y])/5;
- Otherwise (mode is equal to 3, i.e., single residual, reconstruction Cr = C, Cb = (CSign$*$C)/2), the joint residual is determined according to resJointC[x][y] = (4$*$resCr[x][y]+2$*$CSign$*$resCb[x][y])/5.

tu_cbf_cb	tu_cbf_cr	reconstruction of Cb and Cr residuals	mode
1	**0**	resCb[x][y] = resJointC[x][y] resCr[x][y] = (CSign * resJointC[x][y]) >> 1	**1**
1	**1**	resCb[x][y] = resJointC[x][y] resCr[x][y] = CSign * resJointC[x][y]	**2**
0	**1**	resCb[x][y] = (CSign * resJointC[x][y]) >> 1 resCr[x][y] = resJointC[x][y]	**3**

TABLE 7.26: Reconstruction of chroma residuals. The value CSign is a sign value (+1 or –1), which is specified in the slice header, resJointC[.][.] is the transmitted residual.

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	**ET**	**BDR**	**ET**	**BDR**	**ET**
A1	**Tango**	0,51%	91%	0,33%	99%		
	Food Market	-0,13%	99%	-0,15%	98%		
	Campfire	-0,20%	98%	0,71%	97%		
A2	**Cat Robot**	0,89%	97%	1,11%	98%		
	Daylight Road	0,30%	98%	0,14%	99%		
	Park Running	1,71%	95%	1,04%	97%		
B	**Market Place**	-0,28%	99%	-0,39%	99%	0,24%	99%
	Ritual Dance	0,45%	98%	0,15%	99%	0,68%	98%
	Cactus	0,01%	98%	0,09%	99%	0,22%	99%
	Basketball Drive	0,59%	99%	0,38%	98 %	0,89%	98%
	BQ-Terrace	0,73%	98%	0,89%	99%	1,10%	99%
C	**Basketball Drill**	1,02%	97%	0,55%	98%	0,70%	99%
	BQ-Mall	1,03%	98%	0,54%	99%	1,08%	100%
	Party Scene	0,55%	98%	0,21%	98%	0,81%	99%
	Race Horses	0,51%	99%	-0,09%	98%	0,64%	99%
Overall	**BDR-Y**	0,62%		0,57%		0,20%	
	BDR-U	0,24%		0,04%		1,95%	
	BDR-Y	0,15%		-0,52%		2,50%	
Average	**BDR-YUV**	**0,51%**	**98%**	**0,37%**	**98%**	**0,71%**	**99%**

TABLE 7.27: Performance results of VTM JCCR tool "off" test on various JVET CTC sequences in different configurations.

Only I slices support the three joint chroma coding schemes described above. Only mode 2 is available in *P* and B slices. Hence, the syntax element tu_joint_cbcr_residual_flag is only available in *P* and B slices if both chroma CBFs are 1. In the context modeling of tu_cbf_luma and tu_cbf_cb, transform depth is eliminated.

We tested JCCR on different sequences in different configurations. The results are presented in Table 7.27.

7.1.4 In-Loop Filters

7.1.4.1 Adaptive Loop Filter (ALF)

This filter applies at CTB level with filter parameters signalled in the slice header. A flag is transmitted for luma and two others for chroma U and V to enable it at CTB level. The filter is then picked up among up to 25 filters for each 4×4 block based on the direction and activity of the local gradients. There are two filter shapes: one 7×7 diamond shape for luma and 5×5 diamond shape for chroma, as shown in Figure 7.27. The class index of the filter is derived based on the 4×4 gradient directionality D and a quantized value of its activity Â:

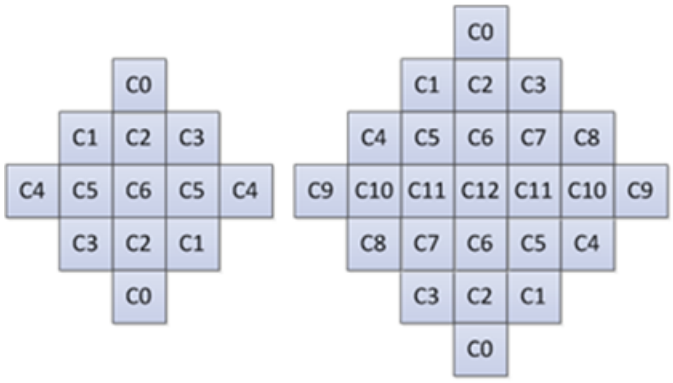


FIGURE 7.27: ALF filter shapes (chroma: 5×5 diamond, luma: 7×7 diamond) [8].

$$C = 5D + \hat{\text{A}} \tag{7.15}$$

The gradient computation is performed using a 1-D Laplacian computation using sub-sampled positions.

Before the filtering step, there is a step of geometric transformations of filter coefficients depending on gradient values. This allows applying same filter to several neighbouring 4×4 blocks. Finally, the filtering process is applied as defined below:

$$R'(i,j) = (\sum_{k=-\frac{L}{2}}^{\frac{L}{2}} \sum_{l=-\frac{L}{2}}^{\frac{L}{2}} f(k,l) \times R(i+k, j+l) + 64) >> 7 \tag{7.16}$$

Table 7.28 evaluates the performance of this tool by showing the generated BDR and encoder run-time numbers when this ALF is turned off.

7.1.4.2 Luma Mapping with Chroma Scaling (LMCS)

Before the loop filters in VVC, a coding technique called luma mapping with chroma scaling (LMCS) is included as a new processing block. LMCS consists of two primary components: 1) in-loop mapping of the luma component using adaptive piecewise linear models, and 2) luma-dependent chroma residual scaling for the chroma components. From the standpoint of the decoder, Figure 7.28 depicts the LMCS architecture. The inverse quantization, inverse transform, luma intra prediction, and addition of the luma prediction with the luma residual are among the light-blue colored blocks in Figure 7.28 that illustrate where the processing is applied in the mapped domain. The processing is applied in the original (i.e., non-mapped) domain in the unshaded blocks in Figure 7.28, which include loop filters like deblocking, ALF, and SAO, motion compensated prediction, chroma intra prediction, adding the chroma prediction with the chroma residual, and storing decoded pictures as reference pictures. Figure 7.28 shows the new LMCS functional blocks, which include luma signal forward and inverse mapping, as well as a luma-dependent chroma scaling mechanism. LMCS, like most other VVC tools, can be enabled or disabled at the sequence level using an SPS flag.

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	2,75%	95%	3,42%	99%		
	Food Market	3,00%	96%	3,33%	98%		
	Campfire	2,18%	98%	3,53%	99%		
A2	Cat Robot	2,85%	97%	6,11%	98%		
	Daylight Road	1,87%	98%	5,61%	99%		
	Park Running	3,35%	98%	6,22%	101%		
B	Market Place	2,38%	99%	3,70%	98%	3,67%	99%
	Ritual Dance	2,85%	99%	2,08%	98%	2,37%	96%
	Cactus	2,11%	99%	4,06%	99%	3,47%	98%
	Basketball Drive	2,04%	99%	5,23%	99 %	5,77%	99%
	BQ-Terrace	1,53%	99%	11,06%	98%	9,74%	98%
C	Basketball Drill	4,49%	98%	4,32%	98%	5,70%	102%
	BQ-Mall	1,60%	99%	2,30%	96%	2,46%	95%
	Party Scene	1,17%	100%	5,81%	96%	4,90%	95%
	Race Horses	2,24%	98%	3,01%	98%	3,47%	96%
Overall	BDR-Y	2,12%		4,56%		4,35%	
	BDR-U	2,94%		4,93%		5,59%	
	BDR-V	3,74%		4,94%		5,22%	
Average	BDR-YUV	**2,43%**	**98%**	**4,65%**	**98%**	**4,62%**	**97%**

TABLE 7.28: Performance results of VTM ALF tool "off" test on various JVET CTC sequences in different configurations.

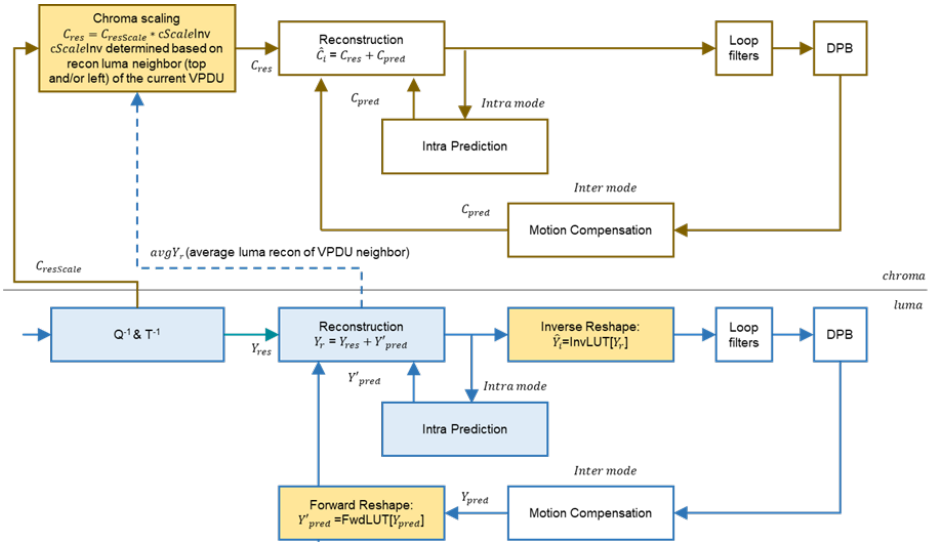


FIGURE 7.28: Luma mapping with chroma scaling architecture [18].

Luma mapping with piecewise linear model: The luma component's in-loop mapping improves compression performance by adjusting the dynamic range of the input signal by dispersing codewords throughout the dynamic range. A forward mapping function, *FwdMap*, and an inverse mapping function, *InvMap*, are used in Luma mapping. A piecewise linear model with 16 equal parts is used to signal the *FwdMap* function. The *InvMap* function is developed from the *FwdMap* function and does not need signaling.

The luma mapping model is indicated by setting aps_params_type to 1 (LMCS_APS) in the Adaptation Parameter Set (APS) syntax structure. In a coded video sequence, up to four LMCS APSs can be used. For each picture, only one LMCS APS can be utilized. The piecewise linear model is used to signal the luma mapping model. The piecewise linear model divides the dynamic range of the input signal into 16 equal parts, and the linear mapping parameters for each piece are represented using the number of codewords allocated to that piece. As an example, consider a 10-bit input. By default, each of the 16 parts will be allocated 64 codewords. The scaling factor is calculated using the signaled number of codewords, and the mapping function is adjusted accordingly for that piece. An LMCS enable flag is set at the slice level to indicate whether the LMCS process, depicted in Figure 7.28, is applied to the current slice. If it is the case, the slice header contains an aps id that identifies the APS that contains the luma mapping parameters.

Two input pivot points InputPivot[] and two output (mapped) pivot points MappedPivot[] define the i -th piece of the *FwdMap* piecewise linear model, with $i = 0 \dots 15$.

Assuming 10-bit videos, the InputPivot[] and MappedPivot[] are computed as follows:

1. OrgCW $= 64$.
2. For $i = 0 : 16$, InputPivot$[i] = i *$ OrgCW.
3. For $i = 0 : 16$, MappedPivot$[i]$ is calculated as follows:

$$\begin{aligned}&\text{MappedPivot}[0] = 0;\\&\text{for}(i = 0; i < 16; i++)\\&\quad \text{MappedPivot}[i+1] = \text{MappedPivot}[i] + \text{SignalledCW}[i];\\&\text{end for.}\end{aligned}$$

where SignalledCW$[i]$ is the signalled number of codewords for the i -th piece.

Motion compensated prediction is done in the mapped domain for an inter-coded block, as shown in Figure 7.28. In other words, the *FwdMap* function is used to transfer the luma prediction block in the original domain to the mapped domain, $\mathrm{Y}'_{\mathrm{pred}} = \mathit{FwdMap}(\mathrm{Y}_{\mathrm{pred}})$, after the motion-compensated prediction block Y pred is computed based on the reference signals in the DPB. The *FwdMap* function is not used for an intra-coded block since intra prediction is done in the mapped domain. Following the calculation of reconstructed block Y_r, the *InvMap* function is used to transform the reconstructed luma values in the mapped domain to the reconstructed luma values in the original domain ($\hat{\mathrm{Y}} = \mathit{InvMap}(\mathrm{Y}_r)$). Both intra-coded and inter-coded luma blocks are treated with the *InvMap* function.

Look-Up Tables (LUT) or on-the-fly calculation can be used to implement the luma mapping process (forward and/or inverse mapping). $\mathit{FwdMapLUT}$ and $\mathit{InvMapLUT}$ can be pre-calculated and pre-stored for usage at the tile group level if LUT is used, and forward and inverse mapping can be easily implemented as $\mathit{FwdMap}(\mathrm{Y}_{pred})$=$\mathit{FwdMapLUT}[\mathrm{Y}_{pred}]$ and $\mathit{InvMap}(\mathrm{Y}_r)$= $\mathit{InvMapLUT}[\mathrm{Y}_r]$, respectively, if LUT is used. On the other hand, on-the-fly calculation can be employed. *FwdMap*, for example, is a

forward mapping function. The sample value is right shifted by 6 bits to determine which piece a luma sample belongs to (which corresponds to 16 equal pieces). The linear model parameters for that piece are then obtained and used to compute the mapped luma value on-the-fly. Let I represent the piece index, $a1$, $a2$ represent InputPivot$[i]$ and InputPivot$[i+1]$, and $b1$, $b2$ represent MappedPivot$[i]$ and MappedPivot$[i+1]$, respectively. The following is how the *FwdMap* function is evaluated:

$$FwdMap(\mathrm{Y_{pred}}) = ((b2 - b1)/(a2 - a1)) * (\mathrm{Y_{pred}} - a1) + b1 \tag{7.17}$$

In a similar way, the *InvMap* function may be calculated on-the-fly. Because the parts in the mapped domain are not all the same size, the simplest inverse mapping procedure would need comparisons to determine which piece the current sample value belongs to. Such comparisons add to the complexity of the decoder. As a result, VVC applies the following bistream restriction to the values of the output pivot points MappedPivot$[i]$. Assume the mapped domain's range is split into 32 equal parts (for 10-bit video, this range is $[0, 1023]$). If MappedPivot$[i]$ is not a multiple of 32, MappedPivot$[i+1]$ and MappedPivot$[i]$ cannot be in the same piece of the 32 equal-sized parts, i.e., MappedPivot$[i+1]$ >> (BitDepthY $-$ 5) cannot be equivalent to MappedPivot$[i]$ >> BitDepthY (Because of this bitstream limitation, the *InvMap* function may also be used to determine the piece to which the sample value belongs by performing a simple right bit-shift by 5 bits (which equates to 32 equal-sized pieces).

Luma-dependent chroma residual scaling: The purpose of chroma residual scaling is to adjust for the interaction between the luma and chroma signals. The slice level also indicates whether chroma residual scaling is enabled or not. An extra flag is sent if luma mapping is enabled, indicating whether luma-dependent chroma residual scaling is enabled or not. Luma-dependent chroma residual scaling is deactivated when luma mapping is not utilized. Furthermore, for chroma blocks with an area of less than or equal to 4, luma-dependent chroma residual scaling is always deactivated.

The average value of top and/or left rebuilt adjacent luma samples of the current VPDU determines chroma residual scaling. The chroma residual scaling factor computed for the CU associated with the initial VPDU is applied for all chroma transform blocks in that CU if the current CU is inter 128x128, inter 128x64, or inter 64x128. The average of the rebuilt adjacent luma samples is denoted by avgYr (see Figure 7.28). The following steps are used to calculate the value of $\mathrm{C_{ScaleInv}}$:

1. Find the index Y_{Idx} of the piecewise linear model to which *avgYr* belongs based on the *InvMap* function.
2. $\mathrm{C_{ScaleInv}}$ = cScaleInv$[\mathrm{Y}_{Idx}]$, where cScaleInv[] is a 16-piece LUT pre-computed based on the value of SignalledCW$[i]$ and a offset value sginalled in APS for chroma residual scaling process.

$\mathrm{C_{ScaleInv}}$ is a constant value for the whole chroma block, unlike luma mapping, which is done on a sample-by-sample basis. The following is how $\mathrm{C_{ScaleInv}}$ applies chroma residual scaling:

$$\text{Encoder side: } \mathrm{C_{ResScale}} = \mathrm{C_{Res}} * \mathrm{C_{Scale}} = \mathrm{C_{Res}}/\mathrm{C_{ScaleInv}} \tag{7.18}$$

$$\text{Decoder side: } \mathrm{C_{Res}} = \mathrm{C_{ResScale}}/\mathrm{C_{Scale}} = \mathrm{C_{ResScale}} * \mathrm{C_{ScaleInv}} \tag{7.19}$$

Encoder-side LMCS parameter estimation: To estimate the LMCS model parameters, the VTM encoder provides a non-normative reference implementation. The reference algorithm in VTM7.0 is built differently for SDR, HDR PQ, and HDR HLG sequences since VTM anchors handle SDR, HDR PQ, and HDR HLG sequences differently. The encoder algorithm for SDR and HDR HLG sequences is

based on local luma variance and tuned for PSNR measurements. The encoder technique for HDR PQ sequences is based on luma values and tuned for wPSNR (weighted PSNR) metrics.

LMCS parameter estimation for SDR:

The VTM7.0 reference implementation for SDR works on the principle of assigning pieces with more codewords to dynamic range segments with lower than average variance and fewer codewords to dynamic range segments with greater than average variance. Smooth regions of the image will be coded with more codewords than the average in this fashion, and vice versa. VTM7.0 includes a reference method for SDR as well as user-configurable LMCS settings.

The reference method analyzes the following signals for SDR test sequences:

1. In the case of 10-bit internal coding bit-depth, statistics of the input video are gathered and evaluated. If the bit-depth of the internal coding is not 10-bit, it is first normalized to 10-bit.

2. Divide the $[0, 1023]$ dynamic range into 16 equal parts.

3. The local spatial variation of luma sample values is computed for each luma sample location in the picture using a winSize $\times$ winSize, (winSize $= \lfloor \min(\text{width}, \text{height})/240 * \rfloor 2 + 1$) neighbourhood centered on the current position. Assign the letter p to the exact piece (out of 32) to which the current luma sample value belongs. As a result, the p -th piece is linked to this local variation.

4. Calculate the average local spatial variance (bin_var) for each of the 16 components.

5. $\text{binCW}[i] = \text{round}\big(\frac{\text{totalCW}}{\text{endIdx}-\text{startIdx}+1}\big)$, where binCW$[i]$ is the number of codewords assigned to the i -th piece, totalCW is the total amount of codewords permitted, and startIdx and endIdx are the index values for the first and last valid pieces, respectively.

6. Codeword allocation is modified such that more codewords are assigned to parts with lower average local spatial variation and less codewords are given to pieces with higher average local spatial variance;

   if normVar < 1.0,

$$\text{binCW}[i] = \begin{cases} \text{binCW}[i] + delta1, & 0.8 \leq \text{normVar} < 0.9 \\ \text{binCW}[i] + delta2, & \text{normVar} < 0.8 \end{cases} \tag{7.20}$$

   else if normVar > 1.0,

$$\text{binCW}[i] = \begin{cases} \text{binCW}[i] - delta1, & 1.1 < \text{normVar} \leq 1.2 \\ \text{binCW}[i] - delta2, & \text{normVar} > 1.2 \end{cases} \tag{7.21}$$

   where normVar $=$ binVar$[i]$/meanVar, $delta1 = \text{round}(10 * \text{hist})$, $delta2 = \text{round}(20 * \text{hist})$, and binVar$[i]$ is the average local spatial variance for the luma values in the i -th piece; meanVar is the mean of the average local spatial variances across all valid pieces; and hist is the percentage of samples in the i -th piece over the total number of samples, clipped to the range of $[0, 0.4]$ to avoid aggressive codeword assignment.

7. If the total number of codewords assigned exceeds the maximum number of codewords that can be assigned. Starting with the first component, the total number of codewords is lowered by an equal amount.

8. Based on the relative histogram and average local spatial variations of the luma signal before and after reshaping, adaption decisions are made to determine LMCS slice type, high bit rate, and chroma modify adaptation parameters. Slice type adaptation enables LMCS to deal with the following slice types: intra and inter; pictures with temporal-ID 0 only; or inter alone. For QP values less than or equal to 22, high bit rate adaptation refers to changing the amount of codewords allocated to pieces. Disabling or activating chroma residue scaling is referred to as chroma adjustment adaption. The basis for all of these adaptation decisions is a series of threshold comparisons.

9. In an LMCS APS, the SignalledCW[i] values are signaled.

When using LMCS, the luma for intra (I) slices is SSE, while the luma for inter (P or B) slices is weighted SSE. For chroma mode decision, SSE is always used. Based on the codeword assignment of the k -th piece in the piecewise linear model, the weight, $w\_\text{lmcs}[k]$, is calculated as follows.

$$w\_\text{lmcs[k]} = (\text{SignalledCW[k]}/\text{OrgCW})^2 \tag{7.22}$$

Various coding configurations are given different considerations when deciding whether or not to activate LMCS at the picture level. Picture analysis is conducted for each IRAP picture under the Random Access test conditions, as mentioned above. Then, if the original picture's average local spatial variance is big, or if the mapped picture's average local spatial variance is large compared to the original, LMCS is disabled for intra. If LMCS is enabled for the IRAP picture, it is only enabled for the pictures with temporal layer ID equal to 0 for the other inter-coded pictures in the same IRAP period. If LMCS is turned off for the IRAP picture, LMCS is turned on for all inter-coded pictures.

LMCS is enabled for all pictures under the All Intra and Low Delay test conditions. For AI, all coded pictures undergo LMCS parameter estimation, and model parameters are updated in LMCS APS for all coded pictures. The LMCS parameters for LD are calculated every second interval, and the model parameters in the LMCS APS are updated at the instances of those pictures.

LMCS parameter estimation for HDR:

Two kinds of HDR sequences are provided in the JVET HDR CTC: PQ and HLG [198]. The VTM reference encoder treats these two sorts of sequences differently. The VTM reference encoder uses luma-based QP adaptation for PQ sequences and allows the QP value to vary spatially [198]. Static quantization is employed for the HLG sequences [198]. LMCS is used in a different way for these two sorts of sequences, as well. PQ uses a default LMCS mapping function calculated as detailed below to apply LMCS. The LMCS parameter estimation technique, which is similar to that used for SDR, is used for HLG.

In the HDR PQ CTC, the VTM reference encoder utilizes wPSNR instead of the traditional PSNR as an objective quality parameter [198]. To optimize the wPSNR metric, the default HDR PQ LMCS curve is generated to match the delta QP (dQP) function.

Based on the average of luma sample values, the luma-based QP adaptation calculates a local dQP value per CTU:

$$d\text{QP(Y)} = \max(-3, \min(6, 0.015 * \text{Y} - 1.5 - 6)) \tag{7.23}$$

where Y is the average luma value, $\text{Y} \in [0, \text{maxY}]$, $\text{maxY} = 1023$ for 10-bit video [199]. The wPSNR weight ($\text{W\_}SSE$) is calculated using the following dQP values:

$$\text{W\_}SSE(\text{Y}) = 2^{d\text{QP(Y)}/3} \tag{7.24}$$

The default LMCS curve is calculated based on luma sample value as follows:

1. Calculate the reshaping curve's slope: $\text{slope[Y]} = \text{sqrt}\big(\text{W\_}SSE(\text{Y})\big) = 2^{d\text{QP(Y)}/6}$.
2. Set $\text{slope[Y]} = 0$ for $\text{Y} \in [0, 64)$, or $Y \in (940, 1023]$ if the signal is in a restricted range (also known as a standard range) [199].
3. Integrate $\text{slope[Y]}, \text{F[Y} + 1] = \text{F[Y]} + \text{slope[Y]}, \text{Y} = 0 \ldots \text{maxY} - 1$ to get F[Y].
4. FwdLUT[Y] is calculated by normalizing F[Y] to $[0, \text{maxY}]$,
$$\text{FwdLUT[Y]} = \text{clip3}\big(0, \text{maxY}, \text{round}(\text{F[Y]} * \text{maxY/F[maxY]})\big). \tag{7.25}$$
5. Determine how many codewords there are for each of the 16 equal parts. As follows:

   $\text{SignalledCW}[i], i = 0 \ldots 15;$
   $\text{SignalledCW}[15] = \text{FwdLUT}[1023] - \text{FwdLUT}[960]$;
   $\text{for}(i = 14; i \geq 0; i - -)$
   $$\text{SignalledCW}[i] = \text{FwdLUT}[(i + 1) * \text{OrgCW}] - \text{FwdLUT}[i * \text{OrgCW}];$$
   end for.

When LMCS is used for rate distortion optimal mode decision at the encoder, SSE is used for luma and weighted SSE is used for chroma as the distortion measure for an intra (I) slice. Weighted SSE is utilized for both luma and chroma in an inter (P or B) slice. All slices are treated with LMCS.

Like the other tools, LMCS was also ivaluated in VTM7 on the JVET CTC sequences. The numbers are summarized in Table 7.29. Additionally, another representation of the performance of LMCS is pointed out in Figure 7.29, where graphs of the BD-rate and the complexity are plotted for the JVET CTC sequences in three different encoding configurations. These graphs proclaim the disparity in the outcome of LMCS depending on the content. This tool provides compression gains reaching more than 7% as well as narrow effect (even losses sometimes). This kind of tools consists the interest of our contribution.

7.2 Optimized Content-Adaptive Activation of VVC tools

7.2.1 The problem

As mentioned in Section 7.1 of this chapter, VVC consists of an important number of adopted tools. However, these tools may not necessarily achieve the optimal trade-off between complexity and compression efficiency. Based on a work done by the Ad-Hoc Group (AHG)-13 [200] and the results presented in Section 7.1, these tools show variable performances across different contents. Nevertheless, the study of AHG-13 covers only the effect of the deactivation of each tool individually from JVET AHG-13. Our study is going a step further by considering subsets of tools in combination. Doing so, positive and negative interactions between tools are captured and the best combination are derived for any coding performance target.

Therefore, the challenge lies in building an automated classifier able to customize the tools to be activated for each content (at a fine level, e.g., the GOP level), based on its nature and local characteristics

Class	Sequence	All Intra		Random Access		Low Delay B	
		BDR	ET	BDR	ET	BDR	ET
A1	Tango	1,39%	109%	0,90%	90%		
	Food Market	3,17%	96%	0,92%	92%		
	Campfire	0,73%	101%	1,03%	103%		
A2	Cat Robot	1,18%	100%	0,15%	90%		
	Daylight Road	2,84%	113%	0,76%	92%		
	Park Running	2,26%	84%	2,08%	97%		
B	Market Place	1,99%	102%	7,13%	101%	5,51%	95%
	Ritual Dance	-0,52%	105%	1,20%	92%	0,03%	82%
	Cactus	0,00%	100%	0,00%	98%	0,03%	83%
	Basketball Drive	-0,20%	96%	0,58%	93 %	-0,11%	92%
	BQ-Terrace	1,37%	93%	1,37%	89%	1,85%	109%
C	Basketball Drill	-0,54%	98%	1,82%	96%	0,65%	92%
	BQ-Mall	-0,55%	94%	1,08%	101%	0,06%	94%
	Party Scene	-0,40%	105%	1,93%	99%	-0,14%	90%
	Race Horses	0,11%	102%	0,25%	90%	-0,39%	103%
Overall	BDR-Y	1,30%		1,42%		1,28%	
	BDR-U	-0,71%		1,39%		-0,44%	
	BDR-V	-0,25%		0,96%		-0,59%	
Average	BDR-YUV	**0,86%**	**100%**	**1,36%**	**95%**	**0,83%**	**98%**

TABLE 7.29: Performance results of VTM LMCS tool "off" test on various JVET CTC sequences in different configurations.

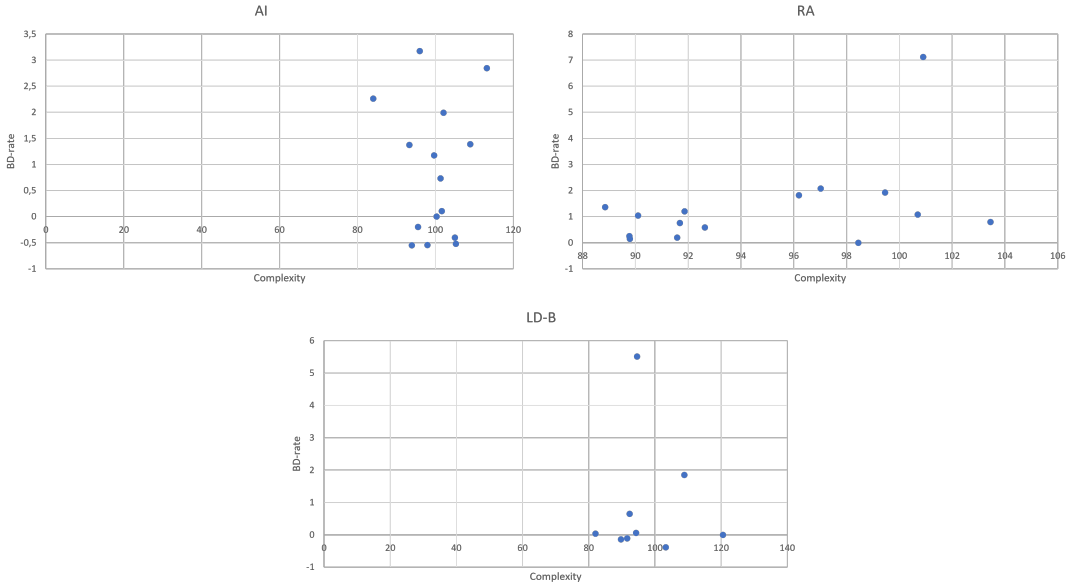


FIGURE 7.29: Graphs showing the performance obtained once the LMCS tool is disabled on the three configurations in VTM7.

Tool Category	Tool name	VTM 3	VTM 4	VTM 5	VTM 6	VTM 7	VTM 8	VTM 9	VTM 10
Intra Tools	CCLM	3.94%	4.01%	3.84%	3.57%	3.60%	3.26%	3.43%	3.43%
	MRL	0.24%	0.18%	0.17%	0.18%	0.14%	0.12%	0.15%	0.15%
	ISP		0.24%	0.12%	0.20%	0.31%	0.28%	0.28%	0.28%
	MIP			0.28%	0.32%	0.37%	0.34%	0.34%	0.34%
Inter Tools	SbTMVP	0.52%	0.43%	0.40%	0.48%	0.43%	0.43%	0,41%	0,41%
	AMVR	0.97%	1.11%	1.13%	1.60%	1.59%	1.56%	1,60%	1,60%
	BCW	0.48%	0.45%	0.46%	0,43%	0.41%	0.40%	0,41%	0,41%
	AFFINE	2.43%	2.47%	2.39%	2,82%	2.80%	2.80%	2,93%	2,93%
	GPM	0.43%	0.43%	0.40%	0,39%	0.44%	0.74%	0,77%	0,77%
	BDOF	1.02%	0.63%	0.67%	0,67%	0.66%	0.66%	0,66%	0,66%
	CIIP	0.43%	0.51%	0.32%	0,24%	0.23%	0.22%	0,20%	0,20%
	MMVD	0.81%	0.52%	0.59%	0,52%	0.51%	0.51%	0,50%	0,50%
	SMVD		0.26%	0.24%	0,27%	0.26%	0.26%	0,24%	0,24%
	DMVR		0.80%	0.87%	0,87%	0.89%	0.88%	0,90%	0,90%
	PROF				0,41%	0.39%	0.38%	0,40%	0,40%
Transform and Quantization Tools	DQ	1.41%	1.39%	1.27%	1.27%	1.28%	1.41%	1.42%	1.42%
	MTS	1.25%	0.82%	0.37%	0.68%	0.70%	0.71%	0.73%	0.73%
	SBT		0.33%	0.34%	0.31%	0.31%	0.31%	0.32%	0.32%
	LFNST			0.75%	0.60%	0.74%	0.75%	0.75%	0.75%
	JCCR			0.28%	0.35%	0.32%	0.41%	0.41%	0.40%
In-Loop filters	ALF	3.61%	3.71%	4.78%	4.65%	4.63%	7.06%	7.18%	7.18%
	LMCS		0.64%	0.61%	0.97%	1.36%	1.32%	1.32%	1.32%

TABLE 7.30: Test results of VTM tools "off" test on various VTM versions in Random Access configuration.

and the considered use case. In other words, we are seeking to adapt the activation of VVC tools according to the type of the scenario.

7.2.2 Motivation

The rationale behind the idea of adapting the activation of the tools to the type of the content is easily understandable with some examples. For instance, MRL is able to provide 0.97% of gains on Bq-Terrace but makes a loss of 0.01% on Ritual Dance (see Table 7.2). Another example is Affine, that is supposed to give 6,56% of gains on Cactus for 83% of encoding run-time (or complexity), while only 0,24% on Bq-Terrace for 80% of encoding run-time. Theses examples reveal clearly that there is a potential in activating/deactivating some tools at the sequence level or at a finer granularity like for instance the GOP level.

Regarding the defined target and based on the variability of the tools performance, demonstrated in the study in Section 7.1 including an evaluation of the BD-Rate gains provided by each tool and taking into consideration the complexity added, one can define the problem as optimizing the efficiency of the VVC tools. The efficiency of a tool here does not include only its coding efficiency but comprises also its complexity. This creates two categories of tools:

- The configurable ones whose activation should be optimized in order to compromise the considered trade-off.
- The consistent ones that should be always activated since they are efficient for all the sequences in all classes, or always deactivated since they are inefficient for all the cases.

7.2.3 Overview of the method

We can summarize our proposal in three steps. First, the performance of each tool is investigated on a considerable number of sequences to understand the convenient use case of each VVC tool. Second, a preliminary framework is built in order to define the optimization method to be followed. Finally, once this method is validated on a set of sequences, the framework is generalized to be applicable on any sequence. This last step can be performed whether using an AI technique, or not. Following these steps, the automated framework becomes capable of optimizing the activation of any tool on any content.

7.2.4 Preliminary framework

Regarding the considered problem, the performance of a tool is meant to be the trade-off between the provided compression efficiency and the complexity imposed by the coding tool. One can see the problem as a bi-objective optimization problem where the tendency is to increase as much as possible the compression efficiency while preserving the complexity at its minimal level. One strategy for solving such problems is to dissociate it into two sub-problems and let their solutions project mutually. Thus, these two problems have two ways to be solved: 1) to be alternately and iteratively solved, and 2) to be mutually solved. Theoretically, according to Von Neumann's, the solution of the alternate projection strategy ultimately converges onto the global optimal solution of the problem [201]. Consequently, the Pareto set concept [202] can be used to approximate the convex hull curve of the encoder configuration space.

As mentioned, the problem is a bi-objective optimization problem where the objective is to maximize an entity on the one hand while minimizing the other entity on the other hand. This fact creates two sub-problems to be solved. Solving these sub-problems separately can be associatively useful to two different use-cases.

- **First use-case (compression efficiency constrained use-case):** For any given (minimal threshold) compression efficiency, our method is able to order the tools in a way that the constraint activates the right set of ordered tools and deactivates the remaining ones, resulting in a minimized complexity.
- **Second use-case (complexity constrained use case):** For any fixed (maximum) complexity on a sequence, our method is able to order the right set of tools to be activated in a way that maximizes the compression efficiency.

7.2.4.1 The used metric: Efficiency Ratio

The nature of the considered problem where one entity, the compression efficiency, is supposed to be maximized and the other entity, the complexity, is supposed to be kept at its lowest levels, makes us choose the simplest metric to define the Pareto set curve. This metric can be considered as a ratio, where the numerator (to be maximized) is the compression efficiency and the denominator (to be minimized) is the complexity. A tool with a higher ratio than another tool is considered to be better. In other words, the choice of this ratio can be justified by the fact that maximizing the bit-rate reduction and minimizing the complexity, is equivalent to maximizing the ratio between the two entities. We can interpret this metric in two ways:

- tool A and tool B are supposed to provide the same value of BD-rate gain (i.e., compression efficiency). However tool A is less complex than B, thus A is more efficient than B.

Correlation (C | R)	test 1		test 2		test 3		test 4	
test 1	1	1	0,8	-0,78	0,53	-0,85	0,87	0,8
test 2	0,8	-0,78	1	1	0,8	0,95	0,83	-0,95
test 3	0,53	-0,85	0,8	0,95	1	1	0,68	-0,95
test 4	0,87	0,8	0,83	-0,95	0,68	-0,95	1	1

TABLE 7.31: Correlation between the 4 methods of tools evaluation.

- tool C and tool D are at the same run-time required to be executed (i.e., complexity). However, tool C is providing more compression gains than D, thus C is more efficient than D.

These two representations of the efficiency of a certain tool is too coherent with the defined metric. In the first representation, the denominator (complexity) of the ratio for the tool A is smaller than the denominator of tool B, hence the ratio of A will be bigger than B. Similarly, in the second representation, the numerator (compression efficiency) of the ratio for tool C is bigger than the one for tool D. Consequently, the ratio of C is higher than D, and C is supposed to be more efficient. We call this ratio **"Efficiency Ratio"**, since it evaluates the efficiency of the tools.

$$Eff.\ ratio = \frac{\frac{R_{ts}}{R_s}}{\frac{C_{ts}}{C_s}} \tag{7.26}$$

where:

- R and C represent for the two entities of the optimization problem. R stands for the BD-rate gains and C for the complexity.
- t stands for tool and s stands for sequence.
- R_{ts} is the BD-rate gains provided by the considered tool t on the considered sequence s.
- C_{ts} is the complexity imposed by the tool t on the sequence s.
- Rs is the total gains provided by all the tools on the sequence s, which makes the numerator the proportion of gains provided only by this tool to be compared to the other proportions of the other tools.
- Cs is the overall complexity imposed by all the tools on the considered sequence s. The denominator represents then the complexity proportion among all the other tools. All the components of the Efficiency Ratio are measured by VVenC (or VVC).

7.2.4.2 Tools ordering towards an optimized encoder

The efficiency of the tools should be evaluated to pick up finally the best optimized activation of these tools. Accordingly, in order to get an accurate performance of the tools, different studies have been carried out. l. The target we are aiming can be achieved by two ways: either deactivating the tools one after the other until reaching the optimized tools configuration, or the other way round, activating the tools until reaching the optimized configuration. Given these two ways, four different studies are performed to evaluate the tools performance:

Study 1: Starting by an identical study to the one of AHG-13 [200], we consider the complete configuration of VVC (All tools ON), and for each tool, we deactivate the considered tool to see the impact this tool has on the encoder. Afterwards, we measure the Efficiency Ratio of each tool with VTM as a first step, on the JVET CTC sequences [180], for the purpose of checking our metric, the Efficiency Ratio.

Study 2: This second test consists of activating for each case one single tool on top of the tools off configuration (only the core of VTM with no additional tool activated). Based on the BD-rate and complexity numbers measured for each tool, the Efficiency ratio is computed.

Study 3: In this test, starting from the basic tools off configuration, we begin to activate the tools one after the other. The tools are activated by descending order of the efficiency ratio. This order seems logical since the interest is in the most efficient tools, i.e., having the highest efficient ratio, that should be activated. The activation of the tools stops at the optimal point of the Pareto set.

Study 4: In the last test, we start from a full configuration, where all the tools are turned on. Progressively, tools are being deactivated one after the other, in an ascending order of the Efficiency Ratio (from the least efficient to the most efficient).

Study 1 and **Study 2** detect the individual performance of the considered tool, whereas **Study 3** and **Study 4** consider more the interaction between the tools. We can see these four tests as the combinations resulting from two different aspects. The first aspect is the problematic of activating (**Study 2** and **Study 3**) the tools or deactivating them (**Study 1** and **Study 4**). The second aspect is the evaluation of the tool performance in isolation (**Study 1** and **Study 2**) or in interaction with the others (**Study 3** and **Study 4**). VVC tools are evaluated on the JVET CTC high resolution sequences (Class A, B and C) in RA configuration. Run-time and BD-rate numbers are generated for each of the four studies. Afterwards, a correlation measure is performed between these numbers generated in the four studies. This is done in the intention of detecting which study (between 1 or 2) is more significant and accurate to each of the studies 3 (activating in combinations) and 4 (deactivating in combinations). These correlation measures are visualized in Table 7.31 as following:

$$Cov(\mathrm{X},\mathrm{Y}) = \frac{1}{N}\sum_{(i=1)}^{N}(\mathrm{X}_i - \bar{\mathrm{X}})\cdot(\mathrm{Y}_i - \bar{\mathrm{Y}}) \tag{7.27}$$

or

$$Cov(\mathrm{X},\mathrm{Y}) = \frac{1}{N}\sum_{(i=1)}^{N}(\mathrm{X}_i \cdot \mathrm{Y}_i) - (\bar{\mathrm{X}}\cdot\bar{\mathrm{Y}}) \tag{7.28}$$

then,

$$r(\mathrm{X},\mathrm{Y}) = \frac{Cov(\mathrm{X},\mathrm{Y})}{\sigma_{\mathrm{X}}\cdot\sigma_{\mathrm{Y}}} \tag{7.29}$$

Depending on the nature of the use case and its imposed constraint, the first aspect of activating or deactivating is determined. More precisely, when the tendency of the considered use case is to conserve the most possible of the efforts spent to achieve a certain level of compression efficiency (e.g., more than 30%), the aspect followed is to deactivate some tools. Thus, we should follow the concept of starting from a full VTM configuration and deactivate the least efficient tools, which corresponds to Study 4.

Tool	Efficiency Ratio
ALF	1,019
DQ	0,723
SbTMVP	0,448
DMVR	0,416
BDOF	0,271
AMVR	0,270
LFNST	0,251
CCLM	0,230
JCCR	0,2
CIIP	0,141
AFF	0,130
MIP	0,128
MMVD	0,098
MTS	0,079
TPM	0,071
BCW	0,061
LMCS	0,060
SBT	0,052
SMVD	0,045
PROF	0,042
ISP	0,016
MRL	0,012
CST	0,010
SAO	-0,006
IBC	-0,049

TABLE 7.32: Descending order of the tools Efficiency Ratios on Tango.

Whereas if the tendency of the used case is to maintain a certain low level of the obtained complexity (e.g., 4 times more complex than HEVC), the aspect becomes to activate tools. Therefore, we should rather start by the basic tools off configuration providing the lowest complexity and activate the most efficient tools. This corresponds to Study 3.

Once the first aspect is determined, and thanks to the correlation measure performed between the 4 studies, the evaluation method of the tools is determined to compute the efficiency ratio on the concerned video content. In Table 7.31, two correlation measures are presented between the four methods of tools evaluation. These methods of evaluation are also called studies in this section. The first correlation, C, is related to the complexity numbers and the second measure, R, is related to the bit-rate numbers generated in each study. As shown in the table, clearly, the highest correlation for the Study 3 is the one with the Study 2. Same observation for the Study 4. It is concluded that, whether the scenario imposes to activate the tools or deactivate them, the best way to evaluate the tools is to follow the method of Study 2.

Afterwards, once the Efficiency Ratios are computed, we sort them in a descending order: from the most to the less efficient (Table 7.32. The set of tools to be activated is determined for each content with respect to the corresponding use-case constraint. Mainly, the desired tendency is to conserve the most possible of the efforts spent in the standardization of VVC to achieve a certain level of compression while maintaining the level of the obtained complexity. This implies a deactivation aspect, i.e., Study 4.

To evaluate the tools in Study 2, we compare the performance of basic tools off configuration to a benchmark where the considered tool is added to this same basic configuration. This justifies the negative bd-rate numbers since the benchmark provides an additional tool that reduces the bitrate. Contrarily, in Study 4, the benchmark deactivates a tool from its reference which is a full configuration. This justifies the positive BD-R numbers of Study 4. This analysis explains the negative values in the table, where the

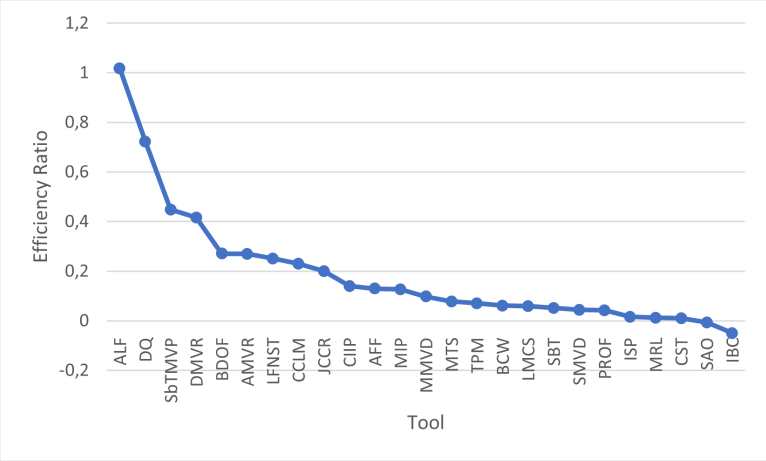


FIGURE 7.30: Efficiency ratios of the tools on tango in a curve.

two Studies are different in the aspect of activating or deactivating. The fact that the highest correlation measure in Table 7.31 is between Study 4 and Study 2 is coherent with our choice of starting from the full configuration tools and deactivating the tools. For this reason, with the aim of determining the right order of the deactivation of the tools until the optimal point, the most accurate evaluation of these tools is proven to be the Study 2, where we activate the tool alone.

An encoder having set of tools activated together defines the global performance. In this case, the interaction between the tools is revealed. Sorting these tools in the order of Study 2 leads to an increasing curve of the efficiency ratios of the global performance in Study 4. In other words, the monotonically decreasing curve of tools efficiency ratios from Study 2 (Figure 7.30 produces an increasing curve of the accumulated tools in Study 4, until a maximal performance indicated by a peak of the efficiency ratio (Figure 7.32). After this point, the curve decreases indicating a deflection of the performance. Meanwhile, the BD-rate/complexity curve, obtained after the deactivation of the tools one by one in the defined order, presents a convex curve (Figure 7.31). The target is to obtain the optimized performance which means the point where the trade off between the BD-rate and the complexity is at its maximum. This can be expressed by the point that is, in the same time, satisfying the highest compression level and the lowest complexity level. This point of the Pareto set convex curve is what we call the optimal point. All the tools activated after this point are contributing in having an optimized performance, while the tools that are deactivated before arriving to this point drain the performance. This optimal point is exactly the point at which the efficiency ratio of Study 4 reaches its maximum before decreasing. This coherence proves the reliability of the efficiency ratio in evaluating the tools. This process unfolds under no constraint, neither the BD-rate reduction nor the complexity, but provides the optimal functionality of VVC by determining the right tools to be activated.

Figure 7.33 shows two different uses-cases of the considered optimization problem. Our method determines finally the order of the tools that guarantees the optimized functioning of VVC. Depending on the use case, the first aspect (activating or deactivating the tools) is determined. Note that, the full VVC configuration (all tools on) is the last point on the right of the curve, while the 'VVC all tools off' configuration is the first point on the left. When the scenario is a compression constrained (First use-case) with a high compression constraint, the aspect is to deactivate the tools. In this case, we start to deactivate tools from the right to the left of the curve, until reaching the limit of the constraint. The

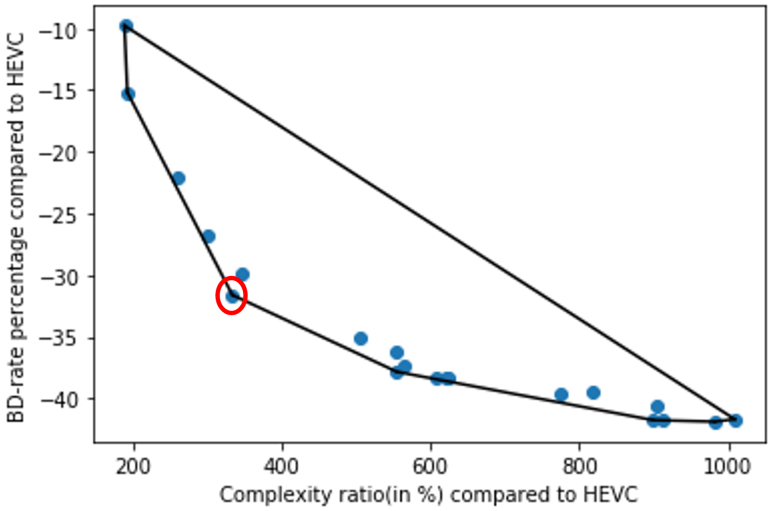


FIGURE 7.31: Convex curve obtained by the optimized order of the tools on tango sequence.

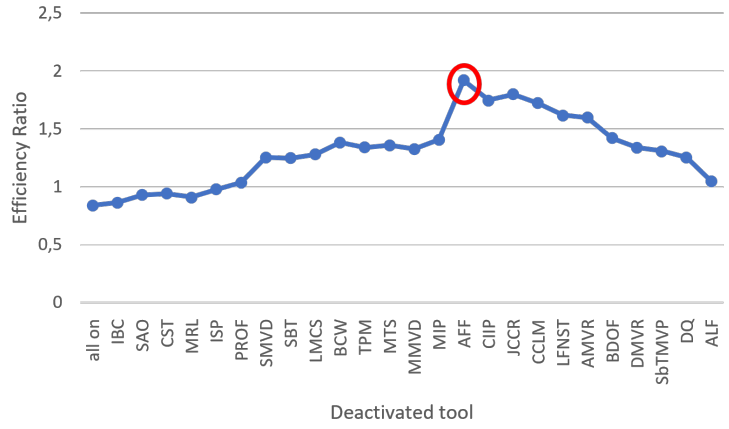


FIGURE 7.32: Deactivating the tools on tango sequence one after the other based on the order of Table 7.32.

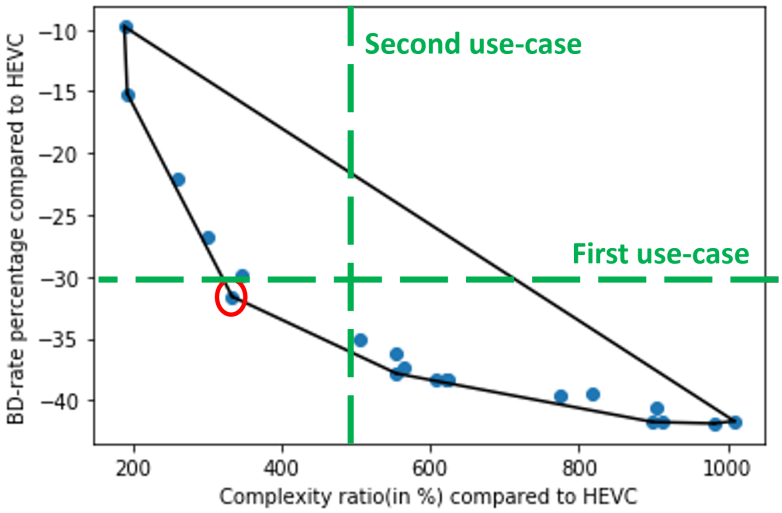


FIGURE 7.33: Our method responding to two different use-cases.

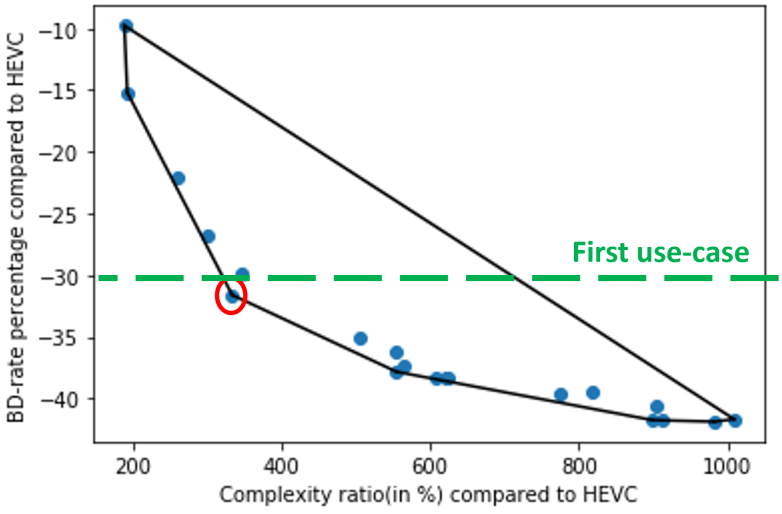


FIGURE 7.34: First use case with a constraint limited to a minimum of -30% of BD-R reductions compared to HEVC.

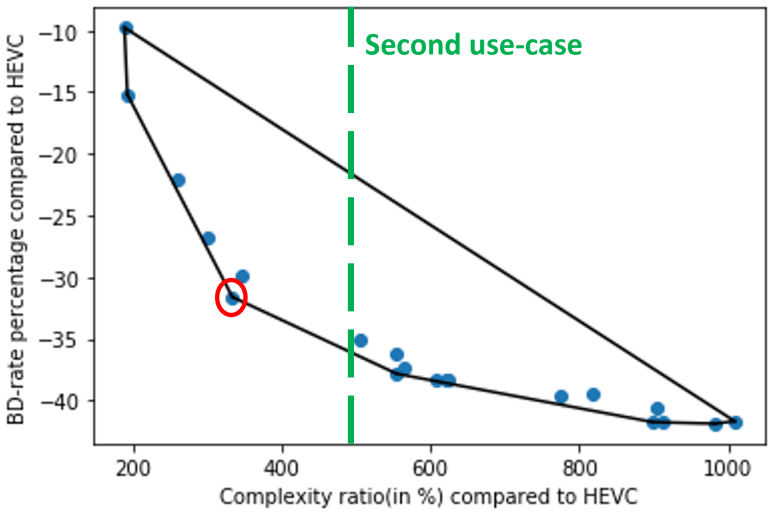


FIGURE 7.35: Second use case with a constraint limited to a maximum of 500% (5 times) of run-time complexity compared to HEVC.

convex hull guarantees an optimized performance for all its points. Consequently, the last point on the curve before reaching the limit is the optimal performance the encoder can have on this content under this constraint. Similarly, if the use case is constrained by a low complexity level (second use-case), we start from the left by activating the tools one after the other until reaching the last point on the curve before the limit of the constraint. Depending on the scenario and its constraints, we determine the aspect of deactivating from the least efficient tool to the most efficient one or the other way round, activating the tools from the most efficient to the least one. Figure 7.34 and Figure 7.35 illustrate two different use cases with their constraints. In each of these figures the optimal point respecting the corresponding constraint is highlighted with a red circle.

As mentioned, our goal is to conserve the most possible compression gains. Moreover, acceptable compression losses cannot be noticed perceptually until a certain threshold. For these reasons, we are more interested in the aspect of deactivating the tools.

To summarize, the method can be divided into two steps: 1) Training our preliminary framework on several sequences in order to provide an optimized performance (oracle mode). 2) Inferring mode or generalized framework where a classifier is built (CNN for instance) to predict our preliminary framework's outcome on any given new sequence, hence guaranteeing an optimal performance on this new input. In the oracle mode we compute the efficiency ratio to obtain the set of ordered tools; in the inferring we just predict this set of tools. The solution is to train with several constraints.

The first step (Figure 7.36) consists in measuring the efficiency ratio of the tools on the sequences. Then, the tools are sorted by descending order of Efficiency Ratio. This order leads us to determine the order of deactivating the tools on the each considered sequence, by ascending order of Efficiency ratio (from the least efficient to the most efficient). This order of deactivation generates a convex hull curve. The constraint, defined by the use case, determines the optimal point which is the last point on the curve before reaching the limit of the constraint. This optimized point corresponds to the last tool to be deactivated. In Figure 7.36, the use case considered is a compression efficiency threshold. Consequently, for any given sequence under a defined constraint, this first step determines the order of tools deactivation

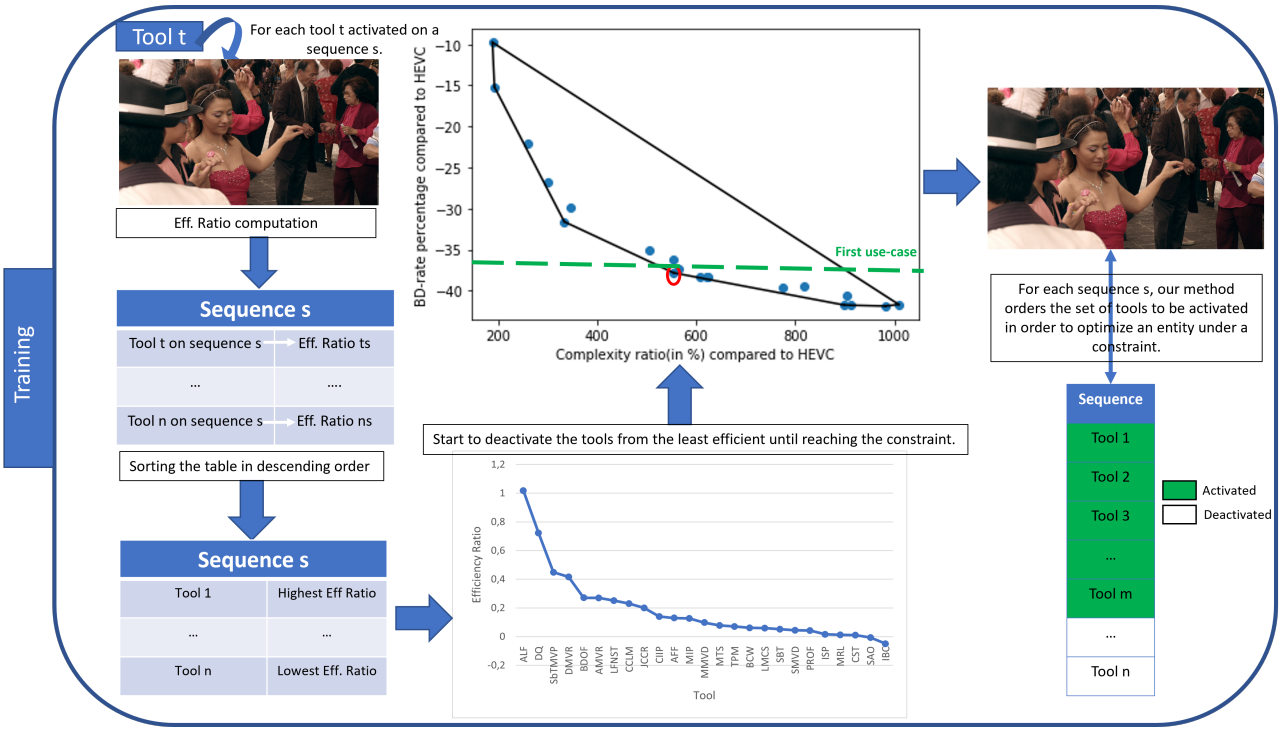


FIGURE 7.36: First step: Cheating mode or training process.

in order to obtain an optimized performance of VVC. This process is repeated on a bunch of sequence, to train our method, aiming to generalize it.

7.2.4.3 Experimental validation

Once our algorithm is designed, we evaluate it by applying it on a random sequence. We chose a Class A (4k) sequence from the JVET CTC, tango, in random access configuration. The first step is to apply Study 2 strategy, where each tool is activated alone in order to compute the efficiency ratio on tango sequence. Then, these tools are sorted in the descending order of efficiency ratio measured on the considered sequence, as shown in Table 7.32 and on the curve in Figure 7.30. From a full VTM configuration, we deactivate successively the tools one after the other, starting from the least efficient tool which is the last one in Table 7.32, i.e., IBC.

For each new deactivation, the encoder changes its behavior since each time it includes new set of remaining activated tools. This led to obtain the convex BD-rate/complexity curve in Figure 7.31 while deactivating the tools by the defined order. Note that the x-axis in this curve is the complexity ratio of the considered VVC encoder compared to HEVC, where a full VVC (all tools activated) is around 10 times (1000%) more complex than HEVC. Similarly, the y-axis shows the provided BD-rate gains compared to HEVC. We can clearly see that deactivating ISP, MRL, CST and SAO reduce remarkably the complexity level without causing noticeable losses in the compression efficiency. All the points located on the curve guarantee an optimal trade-off between the complexity and the compression efficiency. The difference between these points is a matter of priority; the more we go to the right, the more we give priority to the compression, while going to the left means giving more importance to the complexity.

		Sequences			
		Tango	**FoodMarket**	**Campfire**	**CatRobot**
Tool name	**CCLM**	✓	✓	✓	✓
	MRL	X	✓	✓	✓
	ISP	X	X	X	X
	MIP	X	✓	✓	X
	SbTMVP	✓	✓	✓	✓
	AMVR	✓	X	X	X
	BCW	X	X	X	X
	AFFINE	X	✓	X	✓
	TPM	X	X	X	✓
	BDOF	✓	✓	✓	✓
	CIIP	✓	✓	✓	✓
	MMVD	X	X	X	X
	SMVD	X	X	X	X
	DMVR	✓	✓	X	✓
	PROF	X	X	✓	✓
	DQ	✓	✓	✓	✓
	MTS	X	X	X	X
	SBT	X	X	X	X
	LFNST	✓	X	X	X
	JCCR	✓	✓	✓	✓
	ALF	✓	✓	✓	✓
	LMCS	X	X	✓	X
	CST	X	✓	✓	✓
	SAO	X	✓	✓	✓
	IBC	X	X	X	X

TABLE 7.33: Tools activation on 4 different sequences.

Figure 7.32 illustrates another representation of the changing behavior of the encoder depending on the set of tools being activated. In this graph, the x-axis indicates the tool being deactivated from the left to the right, and the y-axis indicates the efficiency ratio of the encoder with the new tools set-up. It should be noted that here, the deactivation of the tools is cumulative, which means each time we deactivate a new tool on top of the already deactivated tools. The point, where the curve reaches a maximum, designates an optimal efficiency. At this point, that we call *'the optimal activation point'*, we stop the deactivation of the tools. Lastly, the highlighted point on the convex hull of the curve in Figure 7.31 corresponds to *'the optimal activation point'* in Figure 7.32. This highlighted point represents the best compromised performance found thanks to our framework.

The same methodology is applied on other sequences, leading to a different set of activated tools on each of the tested sequences. This is due to the fact that each tool can have different performances depending on the considered sequence. Mainly, the fundamental purpose of this method is as follows: adapt the activation of the tools to the content. Table 7.33 reports the different sets of activated tools by sequence and Table 7.34 summarizes the results obtained by the adapted decisions on the concerned sequences (third column). In Table 7.33, we can see that some tools are activated for all the sequences, such as DQ, ALF, CCLM, SbTMVP, BDOF, CIIP. Contrarily, tools like MTS, SBT, ISP, MMVD, SMVD, IBC, are always turned off for all the tested sequences. The activation of all the remaining tools depends on the sequences.

Other experiments are also conducted. We present in Table 7.34 three columns representing each experiment launched with the VTM8.0 on Class A1/A2 sequences (UHD). As a first experiment, the VVC coding performance is measured with all coding tools switched on. The second column on the contrary, is the result of turning off all the switchable tools except those related to partitioning. This

	Over HEVC					
Test	**VVC All tools On**		**VVC All tools Off**		**Trade-off cfg**	
Sequences	BDR	ET	BDR	ET	BDR	ET
Tango	-41,96%	850%	-9,77%	188%	-32,00%	338%
FoodMarket	-38,29%	758%	-16,08%	159%	-32,74%	413%
Campfire	-38,76%	1470%	-14,10%	351%	-34,56%	659%
CatRobot	-46,58%	991%	-10,15%	180%	-38,04%	479%

TABLE 7.34: VVC tests on three different set-up of tools (on UHD content).

experiment provides the lowest possible complexity, at the price of a much lower bitrate saving. Actually, this demonstrates that a very limited effort (i.e., flexible partitioning) on top of HEVC can still significantly improve its performance. As expected, this experiment results in 10% to 16% bitrate saving on 4K and 8K contents. We proceed with a third experiment, presented in the third column, aiming to find the best complexity/bitrate trade-off. In other words, we want to conserve the highest possible proportion of the maximum bitrate gains provided by full VVC (all tools turned on), while keeping the complexity level as low as possible; this, without using any AI technique. This third experiment takes advantage of the tools evaluation conducted inside JVET Ad-hoc group 13. In addition to the study covering the effect of the deactivation of each tool individually, our study is going a step further by considering subsets of tools in combination. Doing so, positive and negative interactions between tools are captured and the best combination are derived for any coding performance target. Figure 7.37 and Figure 7.38 illustrate the performance in two different manners. In Figure 7.37, the performances points are grouped by tests of Table 7.34. For example, the trade-off category (in green) corresponds to the third column of the aforementioned table. Clearly, the trade-off test compromises between, a compression efficiency not far from the maximum one achieved by full VVC and a reasonable complexity not far from the one achieved by the least complex VVC encoder, with all tools off. Figure 7.38 shows the trade off made for each sequence.

As mentioned, the VVC core algorithm is able to provide up to 16% on some contents, without the activation of any additional tool, as presented in the second column of Table 7.34. On the other hand, adding new tools (e.g. dependent quantization, adaptive loop filter, affine motion and adaptive motion vector resolutions, etc.) potentially provides additional bitrate saving on top of the first experiment [200]. In this work, we show that with a subset of tools, VVC is able to assure sufficient gains with less complexity added.

As a final result, our technique provides an encoder that is two times less complex than full VVC, for a bitrate saving only 1.2 times lower. The trade-off disables some tools bringing minor gains compared to the introduced complexity. It is noted that the presented results are measured on UHD test material using the VTM-8.0 reference software. Thus, higher gains are expected to be achieved using latest reference software VTM-10.0. We can legitimately conclude that a subset of VVC tools with reasonable complexity makes 8K broadcast deployments possible, achieving high acceptable gains over HEVC with a reasonable complexity increase [203]. Since both VVC and Essential Video Coding (EVC) benefit from similar partitioning scheme and tools, one can expect that this technique can be applied on EVC or other future standard in the near future.

These experimental results validate the benefit of our framework in providing an optimized activation of VVC tools, in this first step of training. Afterwards, the second step consists in predicting for any given sequence, this order of deactivation to achieve the optimal performance. Besides the possibility of computing for each tool, its efficiency ratio, our classifier can also determine for any sequence under a

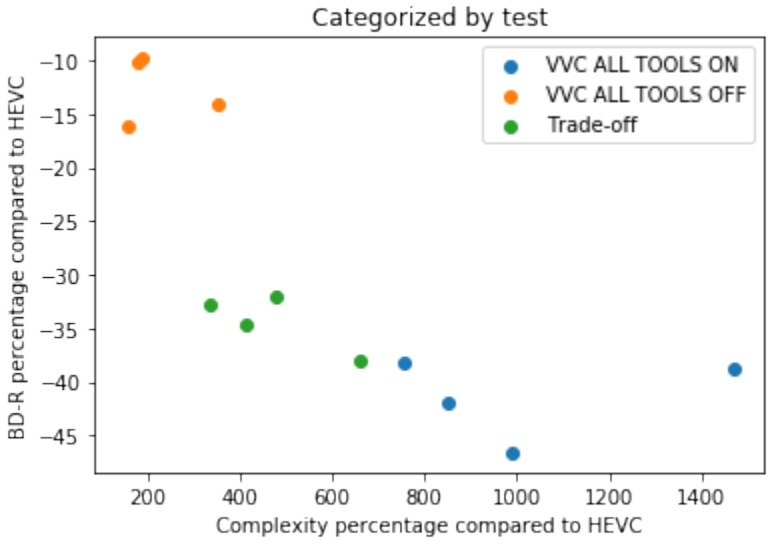


FIGURE 7.37: Comparison of the three tests performance.

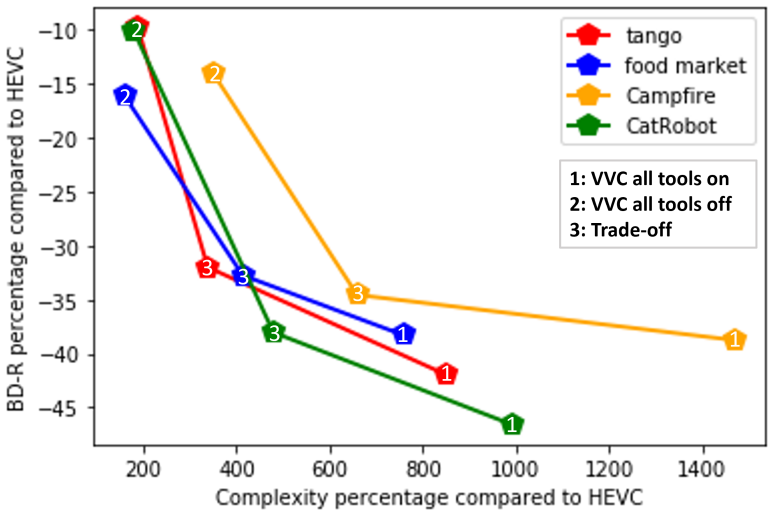


FIGURE 7.38: Comparison of the three tests performance for each sequence.

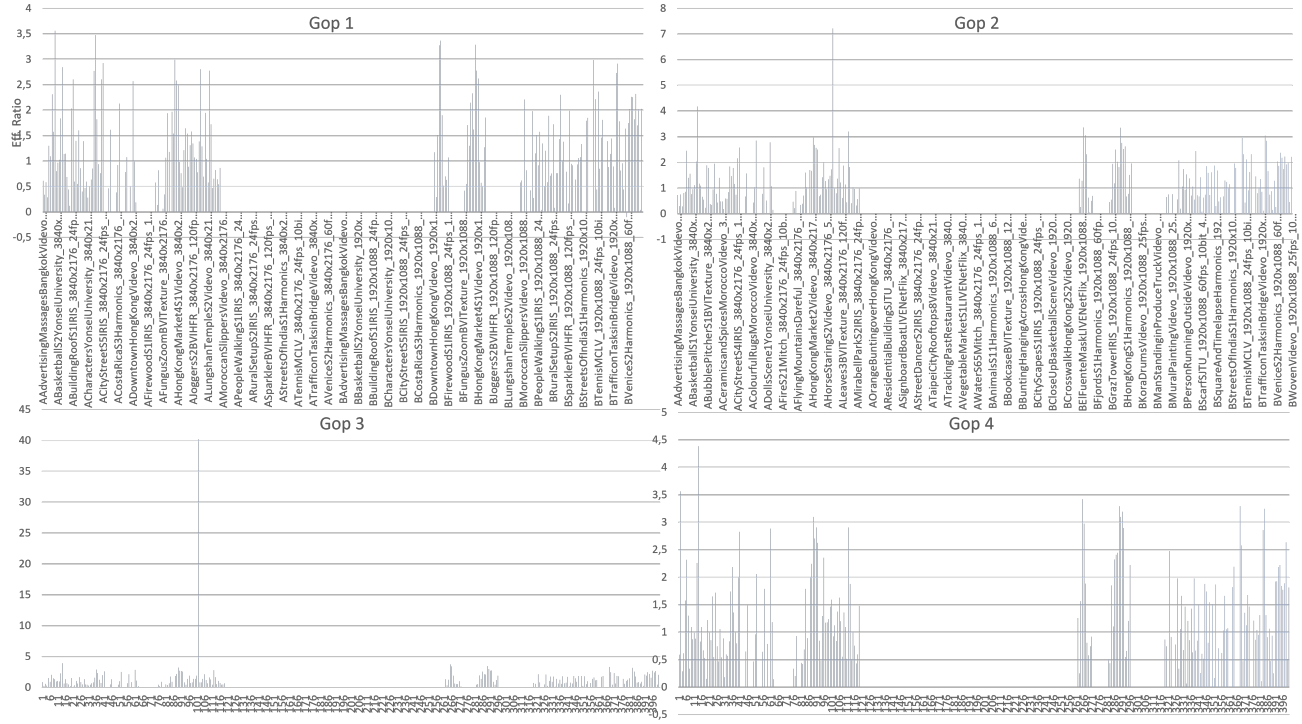


FIGURE 7.39: Variability of SMVD performance across different GOPs from variant sequences.

certain constraint, the set of tools to be deactivated providing an optimal performance.

7.2.5 Generalized framework

In order to guarantee the generalization ability of the constructed model, we tested the VVC tools with the VVenC software on an extensive and representative video database, BVI-DVC dataset [35]. BVI-DVC contains 800 sequences at various spatial resolutions from 270p to 2160p, carefully selected including diverse content. This database is chosen since it offers significantly improved training effectiveness compared to other commonly used image and video training databases. Knowing that VVC is mainly designed for high resolutions, 200 high resolution sequences (Full HD and 4K), representing variant contents and different characterisations, are taken into our experiments to evaluate the variability in the performance of the tools.

Afterwards, with the intention of applying the preliminary framework described in the previous section, the efficiency ratio is calculated to illustrate this variability in the performance of each tool across different contents from BVI-DVC. In these experiments, we prove our approach declaring that every tool can have different performance depending on the type of the sequence. Moreover, in the same sequence, the considered tool may also perform differently when there is a scene changing for instance. Accordingly, we show in these experiments the evaluation of several tools on the 200 concerned sequences from BVI-DVC. All the graphs respect the same representation: the horizontal axis for the sequences (or the id number of the sequence) and the vertical axis for the efficiency ratio. In VVenC, two configurations are proposed for the GOP size, 16 and 32; and, BVI-DVC sequences are composed of 64 frames. In the considered

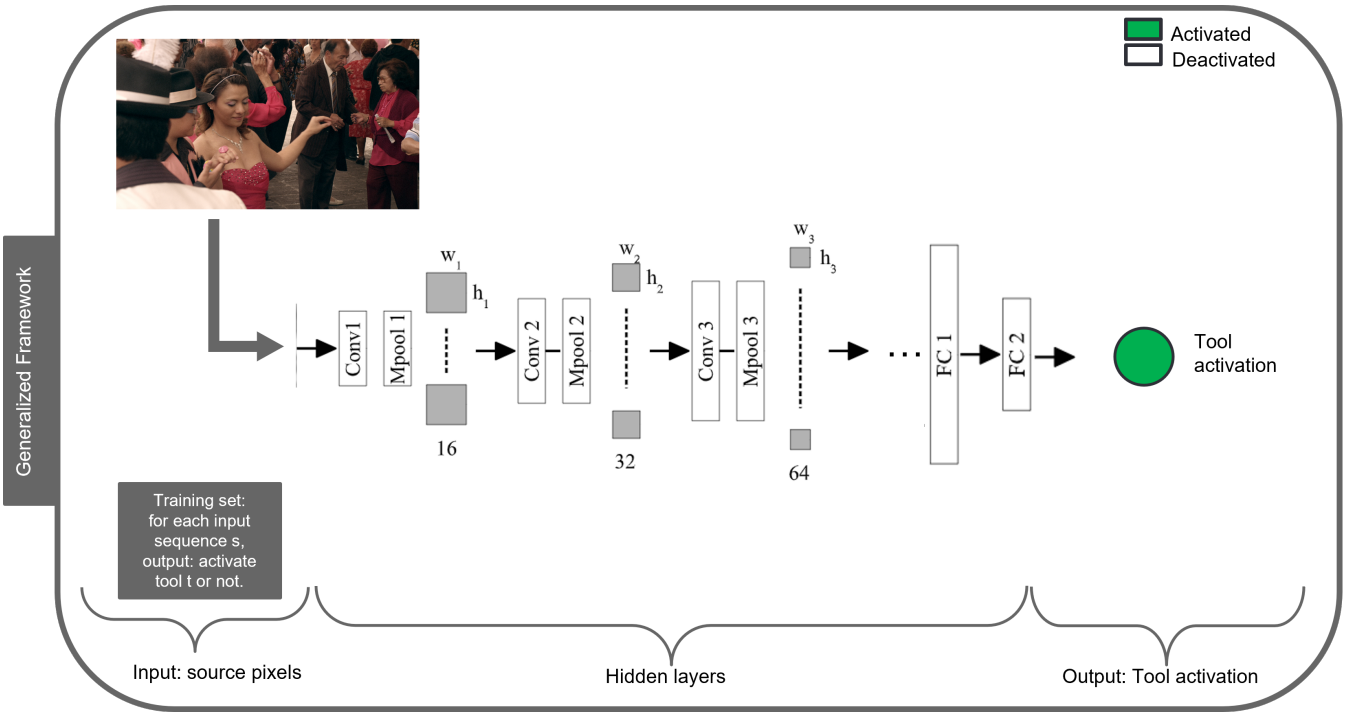


FIGURE 7.40: An example of simplified generalized framework.

framework, we test the tool using the first configuration, i.e., on 4 parts of the sequence, each of 16 pictures. In the target of generalizing our framework, we test all the new adopted tools on all the GOPs of these sequences with VVenC. The idea is to train our algorithm on these contents, so it will be able to perform well on any new video sequence.

Thereby, the oracle mode consists of a data-set determining for each sequence under any use-case constraint, the right tool configuration resulting in optimal performance. Several examples of CNN could be used to predict the right tools set, for instance Lenet5, vgg16, Mobilenet, Resnet or I3D. The inferring mode, known also as the generalized framework, can be performed in different ways. One example can be to build a classifier for each tool, deciding the activation based on if the tool is giving compression losses or not, i.e. negative efficiency ratio (Figure 7.40). Another example to build a classifier which is able to predict the set of tools to be activated for the corresponding sequence under the given constraint (Figure 7.41). A last example is a regression model, where a predictor is employed to predict the value of the efficiency ratio for each tool (Figure 7.42). Then, the tools are ordered. This order is so valuable, since it assures a convex curve of the global performance which means an optimized performance under any imposed constraint. Once the order is followed, the user can define any limit to decide when to stop the deactivation of the ordered tools.

Figure 7.39 shows the variability of the SMVD performance on different sequences, and furthermore the variability across the different GOPs of the same sequence. More precisely, one can see that SMVD shows not only a variant performance between different sequences, but also a remarkable variation within the same sequence, as can be seen for the 'AKartingIRIS_3840x2176_24fps_10bit_420' sequence. For this sequence, SMVD is presenting, more or less, a coherent performance compared to the other sequences on the GOP 1, 2 and 4. While on the third GOP, clearly SMVD boosted its performance. This fact approves

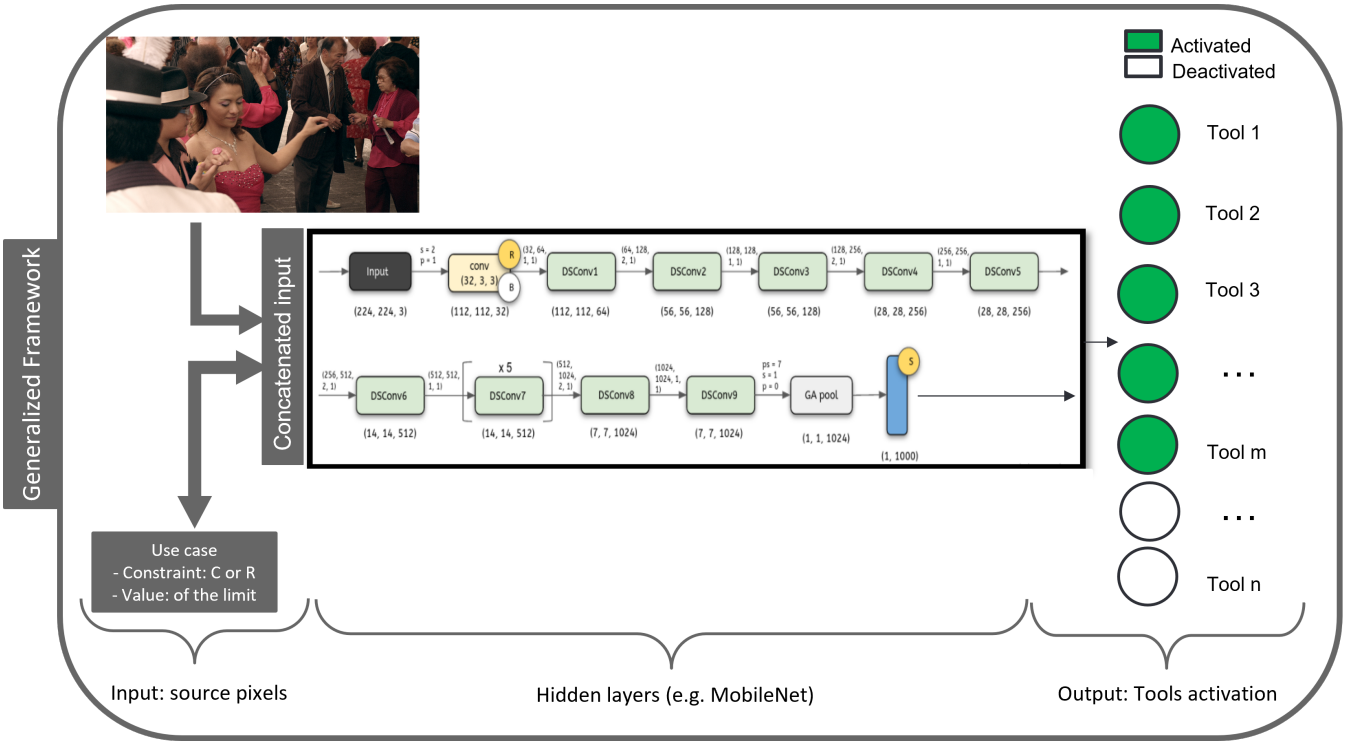


FIGURE 7.41: A second example of inference or generalized framework.

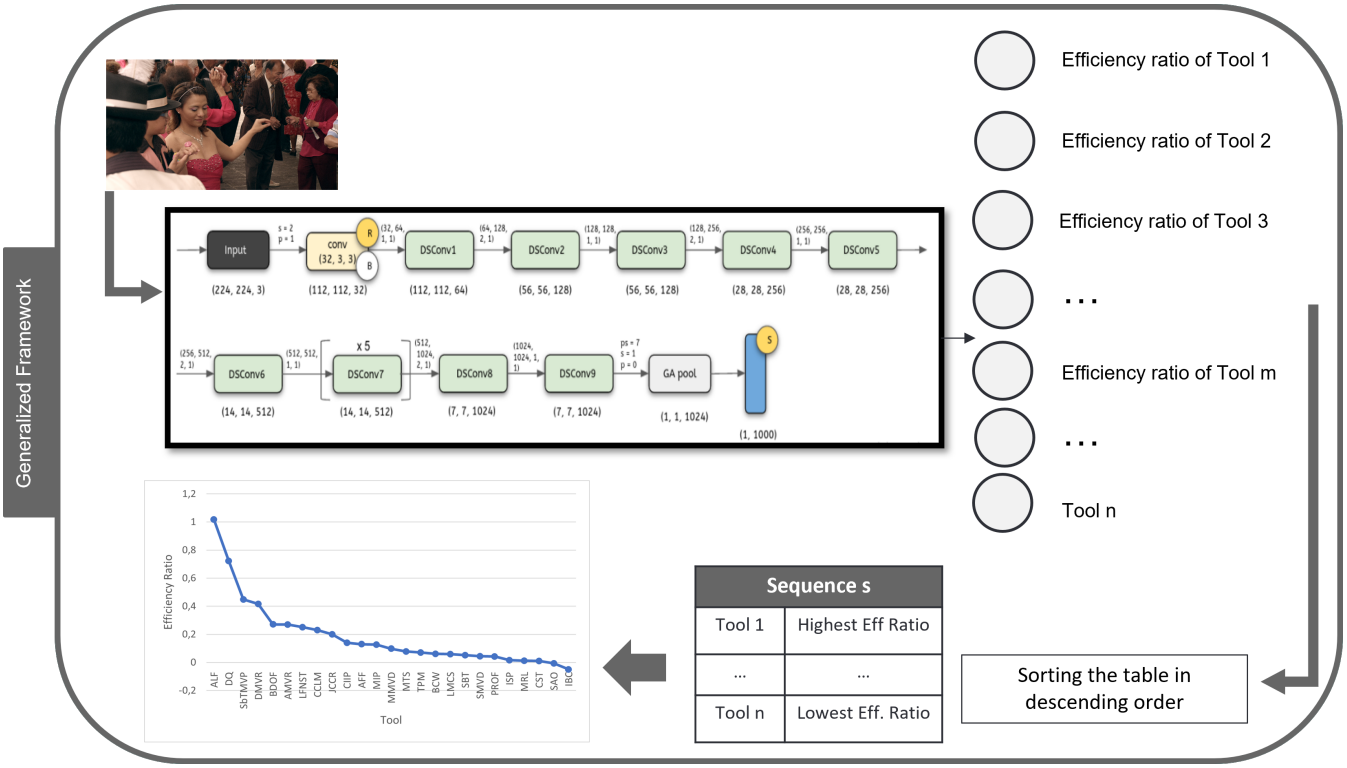


FIGURE 7.42: Third example of inferring.

FIGURE 7.43: Impact of the content nature on the efficiency of SMVD, on the 4 GOPs of 'AKartingIRIS_3840x2176_24fps_10bit_420' sequence.

the choice of activating the tool on some parts of the sequence. In addition, Figure 7.43 supports further our assumption by showing that the nature of the content has a big impact on the performance of the tool, hence some tools should be for some cases deactivated. In this figure, we can notice that the increasing of the blurriness leads to an increasing in the efficiency of SMVD. For this reason, in GOP 3, where the blurriness is glaring, SMVD reaches a very high level of efficiency.

Similarly, this kind of study is applied with all the tools on the considered sequences from the same BVI-DVC data-set. For instance, Figure 7.44 shows a performance that varies for AMVR tool across the sequences and the same remarkable high performance on the third GOP of the karting sequence. However, for JCCR tool, the performance still shows this variability but instead of having the positive peak of the efficiency ratio on the karting sequence, a noticeable negative peak is observed in Figure 7.45. Again, this reinforce the perk of such approach, since deactivating some tools on some sequences can avoid big losses, and the case of JCCR on Karting sequence is an example of these scenarios.

One more time, such analysis confirm the classification of the tools again into two categories. The first including the *consistent* tools, able to provide for the majority of the sequences (even for some tools on almost all the sequences), a good efficiency, hence a permanent activation for these tools. The second category, being the core and the interest of our proposal, includes the tools that can function well on some sequences (or portion of the sequence) but not on others. We call these tools *'the configurable tools'*. For the first category, our algorithm does not intervene, thus these tools are kept activated always. The role of our algorithm is to decide for the tools of the second category whether the activation or not, on the considered content. This second category comports tools like: SMVD, MMVD, MRLP, GEO, MIP, LMCS, LFNST, SBT, JCCR, CIIP. Other tools can be added to this category, but a sub-group of this

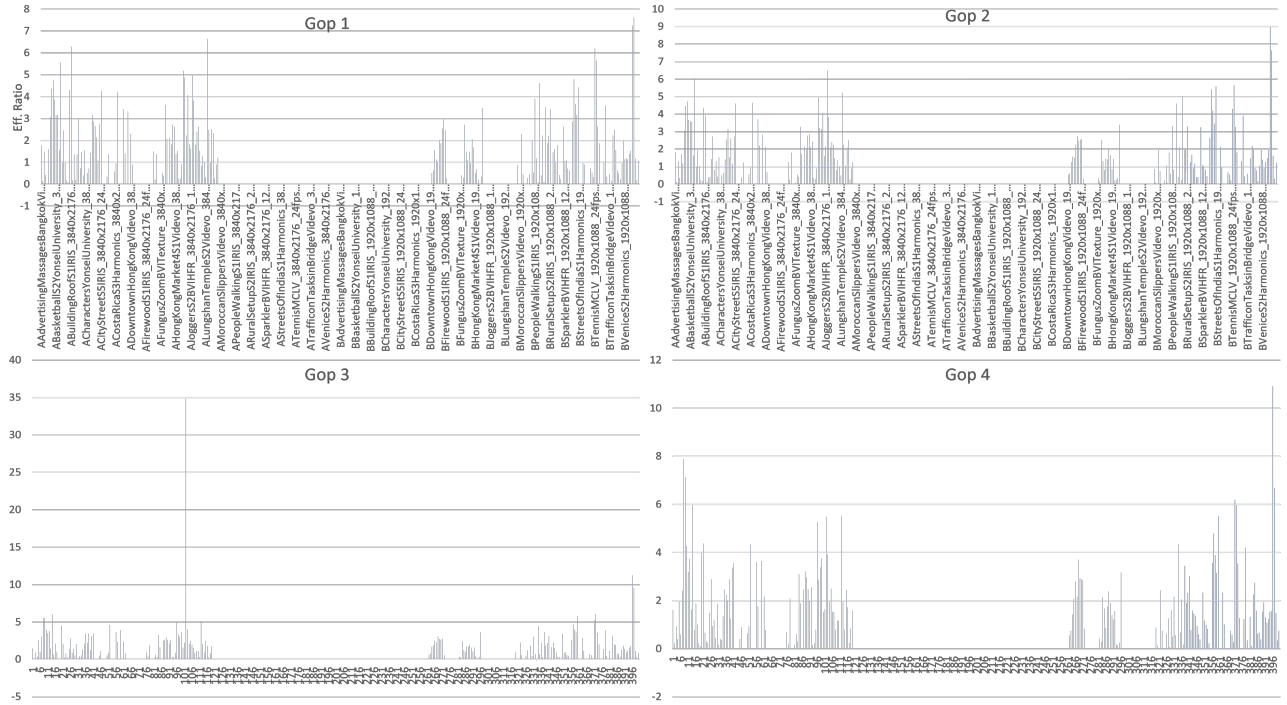


FIGURE 7.44: Variability of AMVR performance across different GOPs from variant sequences.

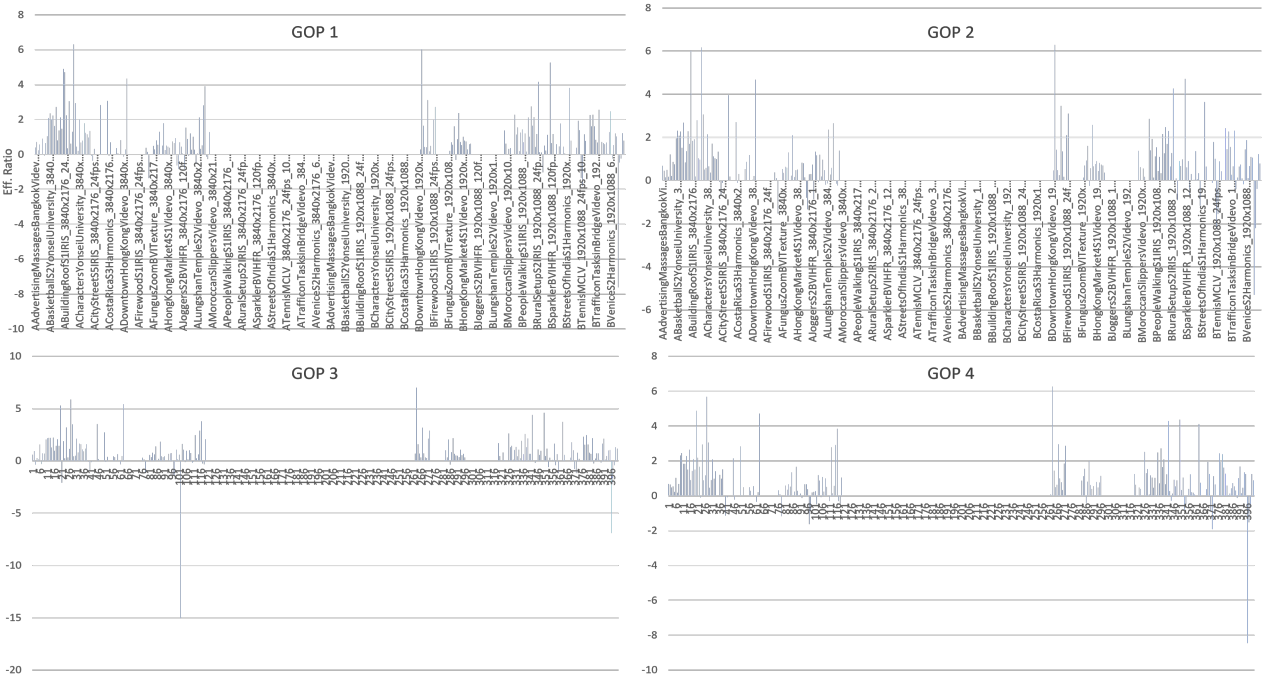


FIGURE 7.45: Variability of JCCR performance across different GOPs from variant sequences.

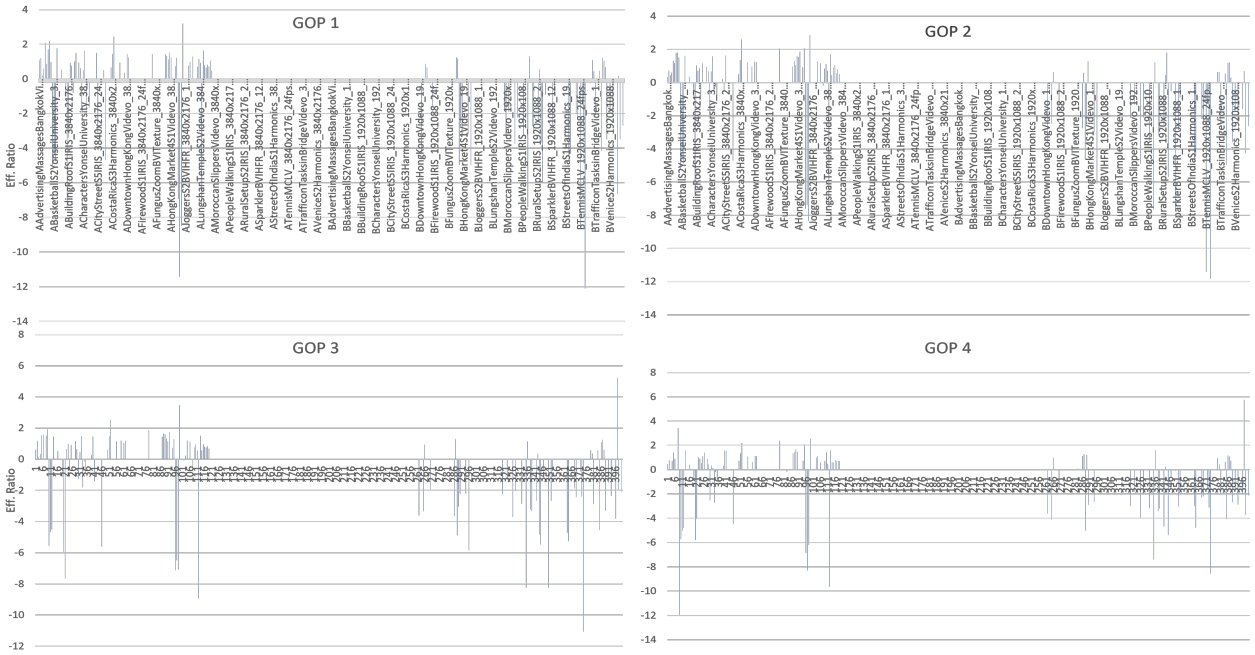


FIGURE 7.46: Variability of LMCS performance across different GOPs from variant sequences.

class is considered as a start. Whereas the first category comports tools like: DQ, ALF, ...

The role of our algorithm is to decide for all the tools whether the activation or not, on the considered content. The generalization of the framework can be executed thanks to an AI technique using artificial neural networks in order to predict the optimal set of tools for each video sequence in a specific use case. This does not dismiss its realization without any AI technique.

7.2.5.1 Simplified generalized framework

To simplify the problem, and before building a generic framework able to determine for each content the list of the tools to deactivate, we start by a simplified framework dedicated for the decision of the activation of one single tool. Based on the LMCS graphs in Figure 7.46, this tool seems to be particular and needs to be well treated. Deactivating LMCS in the convenient cases avoid considerable losses on various sequences. For this reason, we chose this tool to start with. Looking at the test launched, where LMCS is enabled on the considered sequences from BVI-DVC, one can see the problem as a supervised learning problem, where negative bars in the histograms indicating an LMCS deactivation and on the other hand the positive bars corresponding to its activation. Before going through a deep learning technique, a simpler way to accomplish the target of our framework is investigated.

Based on what has been explained about LMCS in Section 7.1.4.2, and looking at Tables 7.29 and 7.35, one can think that there is a relation between the legal range notion and the performance of LMCS. Especially that on cactus, which is a non-legal range sequence, LMCS gives 0,00% in terms of compression. For this reason, this relation is investigated. Note that, content denoted as not conforming to the broadcast legal range are the content that are treated as full range. Accordingly, legal range contents are clipped on an narrower range. For a 10-bit content, the interval of a full range is between 0 and 1023, while for

Class	Sequence	Frame nbrs	Frame Rate	Bit Depth	Legal Range	CTC
A1	Tango	294	60 fps	10	YES	YES
A1	FoodMarket	300	60 fps	10	YES	YES
A1	Drums	300	100 fps	10	YES	NO
A1	Campfire	300	30 fps	10	YES	YES
A2	CatRobot	300	60 fps	10	YES	YES
A2	DaylightRoad	300	60 fps	10	YES	YES
A2	ParkRunning	300	50 fps	10	YES	YES
A2	TrafficFlow	300	30 fps	10	YES	NO
B	MarketPlace	600	60 fps	10	YES	YES
B	RitualDance	600	60 fps	10	YES	YES
B	Cactus	600	60 fps	8	NO	YES
B	BasketBallDrive	500	50 fps	8	YES	YES
B	BQTerrace	600	60 fps	8	YES	YES

TABLE 7.35: Sequences descriptions.

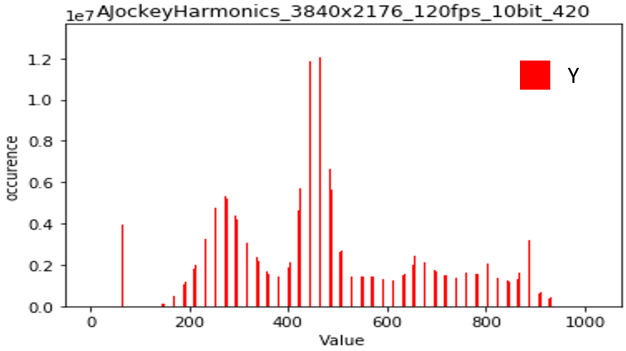
a non-full range, the interval is clipped between 64 and 940. This means, based on this first observation, LMCS is expected to work well on legal range (clipped intervals) contents contrarily to non legal range contents (cactus for instance). To validate this assumption, several examinations are executed.

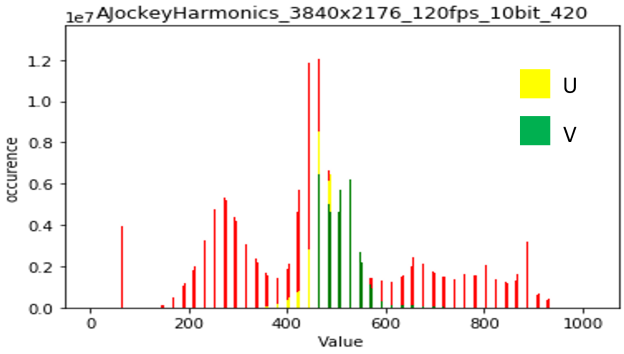
Figure 7.47 and Figure 7.48 and many other sequences affirm the stated assumption, For the *'AJockeyHarmonics'* case, the values of the Y, U and V respect the legal range on all the 4 GOPs. This is coherent with our assumption since LMCS is efficient in this legal range case, as shown in the table in Figure 7.47. Besides, in Figure 7.48, the *'BToyCalendar'* case approves the same assumption, where the Y, U and V values extend to the non-legal range on all the 4 GOPs on which LMCS is causing big compression losses. Although many sequences validate this supposition, a considerable number of sequences shows a contradictory behavior. One of these cases is depicted in Figure 7.49, where the range for *'ABangkokMarketVideo'* is a full range (or not legal). Theoretically, conforming to the stated hypothesis in this subsection, encountering a full range should imply a bad functioning of LMCS hence coding losses. Yet, this is not what can be seen in the table in Figure 7.49. LMCS exceeded 1% of coding gains on the different GOPs of this sequence. Figure 7.50 focuses on the two extremes of the non-legal range to emphasize the presence of samples on these extremes.

Decisions for activating LMCS on different sequences cannot be successfully determined based on simple approaches. Therefore, we switch to Deep Learning, a more sophisticated strategy and more capable of finding patterns that human being cannot do easily.

In Figure 7.46, we show that disabling LMCS on a bunch of sequences avoids coding losses, hence improves VVC performance on the considered sequences. In order to predict the optimal activation of the considered VVC tool at the GOP level, we employ a convolutional neural network to be appealed at the pre-encoding level. We cast the activation prediction problem as a supervised binary classification task detailing the network topology, the training set generation, the learning procedure, and finally we provide a preliminary performance assessment classification-wise.

Our convolutional neural network is tasked with predicting the optimal tool activation to optimize the usage of VVC on each GOP of a sequence. For the LMCS case, the network does not need information available both at the encoder and at the decoder, since we customize the LMCS flag for each slice of the GOP indicating the activation or not of the considered tool. This flag sent from the encoder to the decoder assures the conformity between both sides. For this reason, we assume that the network always takes in input the source (not encoded yet) pixels of the considered sequences. Concerning the network output, we cast the activation problematic as a binary classification task, where the network predicts whether to activate LMCS or not on each GOP. This corresponds typically to the case of Figure 7.40.





	GOP 1			GOP 2			GOP 3			GOP 4		
	Y	U	V	Y	U	V	Y	U	V	Y	U	V
MAX	940	612	740	940	612	744	940	616	744	940	616	744
MIN	64	300	332	64	296	312	64	100	312	64	100	312
Eff. Ratio	**3,204**			**2,871**			**3,472**			**2,567**		

FIGURE 7.47: YUV graphs from 'AJockeyHarmonics' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this seuquence.

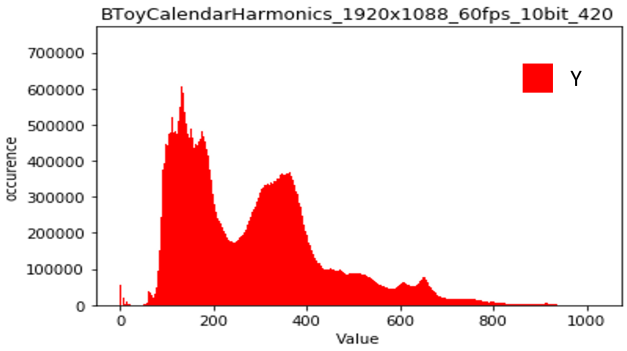
Given the above specifications and design choices, we generate a training dataset as follows. We take 200 Full HD and 4K sequences from the BVI-DVC dataset [35], then we convert them from mp4 format to yuv. Dealing with high resolutions imposes to crop the images into sub-images before passing them as input of the neural network.

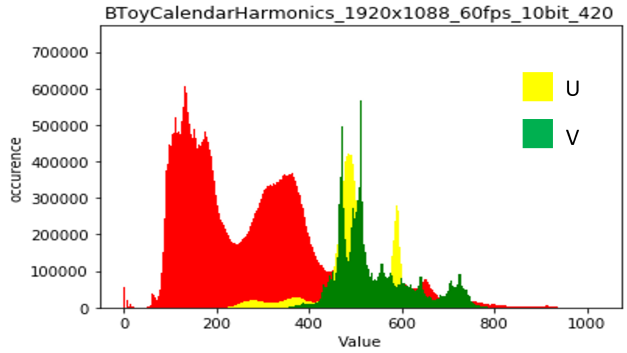
The dataset consists of hundreds of thousands patches with relative ground for training a neural network to predict the behavior of RDO algorithm of this VVC tool.

The design of the network topology is driven by some constraints. One of these constraints is the network computational complexity as well its memory footprint which shall be upper bounded, so that it can be operated in real time also on low-power decoders. One example of such network is the Mobilenet [36]. As the name applied, the MobileNet model is designed to be used in mobile applications. These CNNs are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embedding, and segmentation.

7.2.5.2 Future work

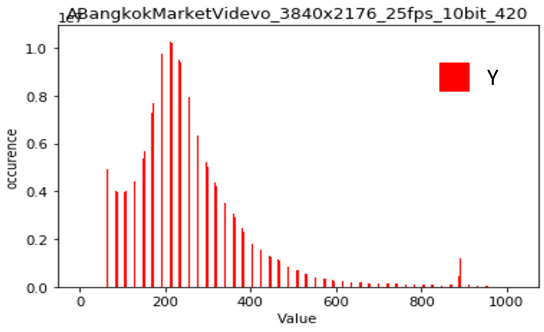
The actual work is focused on the deployment of this neural network. Once done, the next step should be the integration of this framework on top of VTM or VVenC. Consequently, any source sequence will be given as input to our classifier, to determine the right activation cases of LMCS. Afterwards, this predictor will be extended to an automated predictor able to select the suitable tools to be activated and start an optimized encoding process.





	GOP 1			GOP 2			GOP 3			GOP 4		
	Y	U	V	Y	U	V	Y	U	V	Y	U	V
MAX	1023	671	807	1023	674	808	1023	674	809	1023	674	810
MIN	0	181	334	0	181	334	0	180	334	0	180	334
Eff. Ratio	**-12,102**			**-11,822**			**-11,070**			**-8,569**		

FIGURE 7.48: YUV graphs from 'BToyCalendarHarmonics' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this seuquence.



	GOP 1			GOP 2			GOP 3			GOP 4		
	Y	U	V	Y	U	V	Y	U	V	Y	U	V
MAX	1020	700	752	1020	704	756	1020	704	756	1020	704	768
MIN	28	288	304	28	284	304	24	276	284	24	276	284
Eff. Ratio	**0,932**			**0,953**			**1,176**			**0,891**		

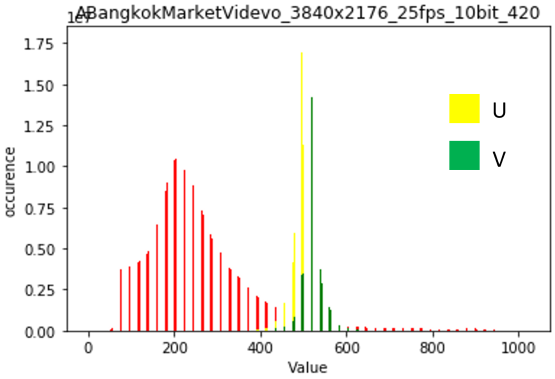


FIGURE 7.49: YUV graphs from 'ABangkokMarketVideo' sequence, and their correlation with the efficiency of LMCS on the 4 GOPs of this sequence.

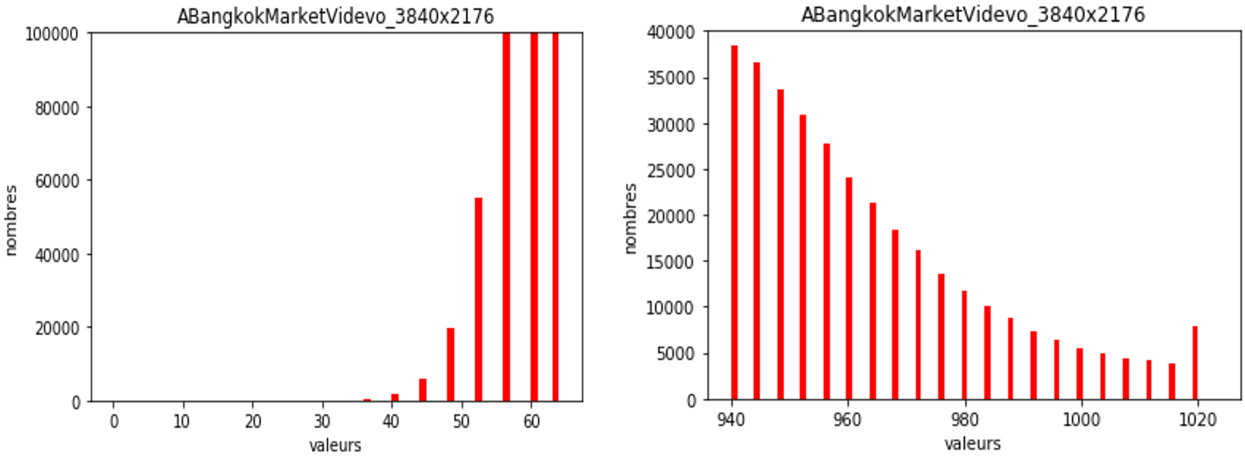


FIGURE 7.50: A zoom in focused on the two extremes of the non-legal range for 'ABangkok-MarketVideo' sequence.

Chapter 8

Conclusion

8.1 Thesis objectives

In this thesis, we have developed original coding methods to increase the coding efficiency of VVC. The methods have been structured around two research tracks.

In the first track, we have provided additional tools to the one already existent in the VVC algorithm. Two contributions compose this first track. The first contribution, DIMD complies to real-world constraints in terms of complexity, memory usage, and practicality. It is thus aimed towards standardization and potential adoption in VVC or in future standards. The second contribution C-SSIF, is more dedicated to research, in the sense that all constraints are dropped, hence giving more freedom to develop an unconventional and innovative technique to increase coding efficiency, regardless of the introduced complexity.

For the second track, the proposed method is not based on adding tools into VVC but rather on optimizing the usage of the tools already adopted. Although this contribution is still on a research stage, the target is to accelerate its integration into ATEME products (e.g.: VVC encoder of ATEME).

8.2 Summary of the contributions

8.2.1 First contribution: DIMD

8.2.1.1 Recap of the method

The first contribution of the first track proposed an implicit intra coding method, called Decoder-side Intra Mode Derivation (DIMD) with prediction fusion. As the DIMD proposal normatively change the existing video compression syntax, it is designed to be considered for next generation video coding standards. In order to demonstrate the functionality as well as the performance of the proposed DIMD, it has been implemented on top of the upcoming Versatile Video Coding (VVC) Test Model (VTM). However, it can be easily adapted to any other standard such as Essential Video Coding (EVC), or AOMedia Video 2 (AV2).

In brief, the proposed method reduces the encoding rate of a video (at the same quality) by adding a new intra coding mode. The operation referred here consists of treating the DIMD identically to other coding modes and in case this mode presents the cheapest encoding cost, it will be chosen among the modes and its choice will be signaled to the decoder. In the DIMD mode, there is no need to send the Intra prediction mode. We propose a smarter technique by building a histogram of gradients on a causal zone surrounding the current block, and by normatively deriving the best intra mode for the current block from this histogram. Another advantage of this proposal is the possibility of having more

than one intra-prediction direction thanks to this same low-complexity texture analysis. These different inferred intra-prediction modes can then be combined by means of prediction fusion, to obtain the final intra-prediction for a given block. This fusion property allows DIMD to more accurately predict areas with complex textures, which in conventional coding schemes would instead require a finer partitioning and/or transmission of high energy residuals.

8.2.1.2 Results

The experiment results of DIMD in VTM-5 shows a relevant trade-off between coding efficiency and codec complexity. More precisely, DIMD can provide up to -0.6% BDR coding efficiency gain with encoder and decoder run-time of 109% and 104%, respectively.

8.2.1.3 Conclusions

This technique has some drawbacks: first, as the derivation of the mode must be performed on decoder side, it is based on rebuilt pixels and not on the original pixels, which causes a precision loss. Furthermore, the operation adds complexity to the decoding process, which can cause problems for hardware implementations. However, as long as the complexity added to the decoder has been mastered as presented in table 5.5, and the gain in terms of signal reduction compensates for the loss of precision in the prediction, our technique has big chance to be adopted in a next video compression standard.

To conclude, this project will not be limited to the work described. Several propositions to enhance the performance of our method are planned to be added: first, instead of using the 3×3 Sobel filter to perform the gradient analysis, we can replace it with a simpler filter (3x2 or even 2x2 filters) able to provide the same outcome. Another planned future improvement, in order to further reduce the complexity, is to avoid the additional RD cost computation imposed by DIMD. One way to do that is to make the encoder decisions based on the distortion instead of adding an RD check. This solution proved its potential in the C-SSIF contribution. These improvements, inspired from the fact that DIMD was not adopted finally in the standard because of the complexity level, can have an important impact on the run-time.

Finally, it should be noted that the proposed algorithm is innovative as it has not been the subject of other published work. This offered me the chance to write my first patents declaring an invention in the video coding field, under the supervision of my mentors. Moreover, after a careful examination of the results showing interesting coding gains while maintaining a low degree of complexity (especially at the decoder), this work offers ATEME the opportunity to present this work as contributions at the JVET meetings, since it had a big chance to be included in the VVC standard. DIMD received a lot of positive feedbacks in JVET meetings and was so close from being added to the VVC standard, but this did not happen finally.

8.2.2 Second contribution: C-SSIF

8.2.2.1 Summary of the method

To summarize the status of the second contribution, as done for DIMD, we will describe in this part the technique followed, its results, its drawbacks and the potential enhancements.

The second contribution of the first track describes another proposed technique to improve the coding efficiency of VVC. The method consists in giving VVC the ability of switching between different interpolation filters with a smart way to avoid the signalization from the encoder to the decoder side,

called CNN-based Switchable Sub-pel Interpolation Filters (C-SSIF). Similarly to DIMD, C-SSIF was also implemented on top of the VTM, but can be adapted to any future standard.

C-SSIF can be divided into 2 parts. In the first part, we propose a family of filters obtained by convolving a Kaiser window with a Sinc (KCS). These filters, with stronger low pass characteristics than DCT-IF, can be beneficial for attenuating high frequency noise components in some contexts. The second part is a scheme where the decoder infers the optimal filter selected by the encoder for each coding unit, without the need to signal the filter coefficients. To do so, we trained convolutional neural networks to predict which filter minimizes the expected distortion for each coding unit only looking at the motion compensated reference in the decoding buffer. The encoder and the decoder share such network, so the filter prediction at the decoder matches that at the encoder.

8.2.2.2 Results

The proposed method achieves on average for Y,U and V respectively up to -0,47%, -0,20% and -0,34% BD-rate reduction for the CTC QP range: 22-27-32-37. The main purpose of this work is to show the interest of adapting the interpolation filter to the content, at a fine granularity, i.e., the block level. Finally, the experimental results presented proved the efficiency of switching from the concept of interpolating with a fix filter to dynamically alternating between three or more filters without signaling by using artificial neural networks to predict the best filter.

8.2.2.3 Conclusions

This technique has also its problems. For instance, the optimal filter may not be correctly predicted all the times due to some bad predictions made by the neural networks. This may cause some losses of the potential gains. Another issue is the complexity level imposed by the CNNs.

To conclude, several propositions and future works are suggested in order to invest and benefit from the potential of this new approach, such as extending from three interpolation filters to more and separating between the vertical and horizontal interpolations. Given that in VVC reference model one fixed DCT-interpolation filter is proposed, the horizontal and vertical interpolations were done using this same filter. However, in our proposal, the interpolation process can benefit from the presence of three or more different filters. The separation creates different combinations of filters, hence the horizontal and vertical interpolations can be executed with different filters when needed. Consequently, the interpolation process gains more flexibility and becomes more adapted to each of the two directions. Therefore, the inter prediction gets more efficient because there will be less residual information and less data to encode and transmit. Experimental results show that the potential BD-rate gains can significantly increase due to the act of separating the interpolation. The aim in a future work is to build deep neural networks able to predict these filters combinations to recover the most possible of the potential gains.

It is worth to mention that this second work has also not been the subject of other published work. This offered me the chance to fill another patent declaring an invention in the video coding and deep learning fields. In addition, a paper detailing the proposed techniques is being finalized for submission to a scientific journal.

8.2.3 Third contribution: Optimized Content-Adaptive Activation of VVC tools

8.2.3.1 Recap of the method

VVC reaches a significant level of bandwidth saving by accumulating several improvements and adopting hundreds of proposals. Although, this high number of adopted tools and algorithms resulted in a significant added complexity. For this reason, it is required to find the optimal trade-off between complexity and compression efficiency by determining the optimized activation of the adopted tools depending on the need.

This contribution determines the right tool configuration of VVC that results in optimal balance between complexity and compression efficiency for any given content in a given use-case (constrained complexity or compression efficiency). It consists of two successive steps, resulting in an optimization problem that the contribution is solving. The method consists in building an automated classifier able to customize the tools to be activated for each content, based on its nature and its local characteristics and the considered use case.

Briefly, the first step consists in building a framework in order to define the optimization technique to be followed. Then, this process is trained over several sequences to determine the order in which the tools should be disabled for achieving an optimized VVC performance under a defined constraint (training mode). The second step consists in using a predictor (CNN for example) that infers the outcome of the first framework (training process) on a new given video sequence, thus guaranteeing an optimized performance of VVC on this new input (inference mode). This second step can be performed whether using an AI technique, or not. Following these steps, our method forms a trained automated predictor able to optimize the activation of any tool on any content.

8.2.3.2 Experimental validations

We presented three significant tests. The first one, representing the upper bounds of the possible obtained coding gains and complexity level, is reached by enabling all the tools. The second experiment represents the lower bounds of the possible generated complexity and coding gains. This is achieved by disabling all switchable tools. Our proposal provides a compromise between the obtained compression efficiency and the complexity imposed by corresponding the right subset of VVC tools for each use case. Significant reductions of coding complexity are acquired (2 times less complex than full VVC), while maintaining a certain level of compression efficiency (only 1.2 times less efficient than full VVC).

More precisely, the proposed contribution disables the tools that bring minor gains compared to the complexity introduced. Since VVC and Essential Video Coding (EVC) benefit from a similar partitioning scheme and tools, this proposal can be applied on EVC or another future standard in the near future.

8.2.3.3 Conclusions

The disadvantage of such technique is the prediction accuracy of the right set of tools. A bad prediction can lead to a poor performance, represented for example by a noticeable loss of compression with a minor reduction in complexity. However, the work is currently focused on this task of deploying our predictor and testing it. Another challenge is to design a powerful predictor with a high prediction accuracy. Thereupon, our method will have every chance to be integrated into ATEME's products: the encoder will know for every given video content what tool to activate to maintain a trade off between its complexity and its coding efficiency.

Finally, it should be noted that this work has not, to our knowledge, been the subject of any other published work. This gave me the chance to conclude my thesis in filing two new patents on this work.

Chapter 9

Publications

9.1 Conferences papers

1. **Anthony Nasrallah**, Elie Gabriel Mora, Thomas Guionnet, Mickaël Raulet: Decoder-Side Intra Mode Derivation Based on a Histogram of Gradients in Versatile Video Coding. DCC 2019: 597;

2. **Anthony Nasrallah**, Mohsen Abdoli, Elie Gabriel Mora, Thomas Guionnet, Mickaël Raulet: Decoder-Side Intra Mode Derivation with Texture Analysis in VVC Test Model. ICIP 2019: 3153-3157.

9.2 Journal paper

Anthony Nasrallah, Attilio Fiandrotti, Mohsen Abdoli, Thomas Guionnet, Marco Cagnazzo: CNN-based Switchable Sub-pel Interpolation Filters (CSSIF) in Versatile Video Coding (VVC) (pending submission).

9.3 Webinars

T. Biatek, M. Abdoli, T. Guionnet, **A. Nasrallah** and M. Raulet (ATEME), Future MPEG Standards VVC and EVC: 8K Broadcast Enabler, IBC Conference 2020.

9.4 Contributions to the MPEG/JVET standard: H.266/VVC

1. E. Mora, **A. Nasrallah**, M. Abdoli, M. Raulet (ATEME), CE3: Decoder-side Intra Mode Derivation (tests 3.1.1, 3.1.2, 3.1.3 and 3.1.4), m45351;

2. E. Mora, **A. Nasrallah**, M. Raulet (ATEME), CE3-related: Decoder-side Intra Mode Derivation , m44184;

3. E. Mora, **A. Nasrallah**, M. Raulet (ATEME), m44200, Crosscheck for CE3-1.1.1 and CE3-1.1.2.

9.5 Patents

1. **Anthony Nasrallah** and Elie Mora, "Method for image processing: Decoder Side Intra Mode Derivation based on Histogram of Gradients", Patent date Issued: Sep 27, 2018, Patent issuer and number: eu 18306269.4 - 1209;

2. **Anthony Nasrallah**, Thomas Guionnet, Elie Mora, and Mohsen Abdoli, "Procédé de traitement d'images et appareil de mise en oeuvre associé", Patent date Issued: Jun 25, 2019, Patent issuer and number: eu 19305848.4-1209;

3. **Anthony Nasrallah** and Elie Mora, "Method for image processing and apparatus for implementing the same." U.S. Patent Application 16/584,144, filed April 2, 2020;

4. **Anthony Nasrallah**, Thomas Guionnet, Mohsen Abdoli, Marco Cagnazzo and Attilio Fiandrotti. Smart Adaptive Interpolation Filters for Video Compression, Patent date Issued: Nov 3, 2020 Patent issuer and number eu: 20306318.5-1208.

Bibliography

[1] B. Bross, J. Chen, J. R. Ohm, G. J. Sullivan, and Y.-K. Wang. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). *Proceedings of the IEEE*, page 1–31, January 2021.

[2] Sotiris Avgousti. *Mobile video tele-echography robotic platform over 4G-LTE network*. PhD thesis, 01 2018.

[3] L. Zhao, X. Zhao, S. Liu, X. Li, J. Lainema, G. Rath, F. Urban, and F. Racapé. Wide angular intra prediction for versatile video coding. In *2019 Data Compression Conference (DCC)*, pages 53–62, March 2019.

[4] Adam Wieckowski, Christian Lehmann, Benjamin Bross, Detlev Marpe, Thibaud Biatek, Mickael Raulet, and Jean Le Feuvre. A complete end-to-end open source toolchain for the versatile video coding (vvc) standard, 2021.

[5] A. Henkel, I. Zupancic, B. Bross, M. Winken, H. Schwarz, D. Marpe, and T. Wiegand. Alternative half-sample interpolation filters for versatile video coding. pages 2053–2057, 2020.

[6] Yong-Uk YOON, Do-Hyeon PARK, and Jae-Gon KIM. Enhanced derivation of model parameters for cross-component linear model (cclm) in vvc. *IEICE Transactions on Information and Systems*, E103.D(2):469–471, 2020.

[7] Y. Chang, H. Jhu, H. Jiang, L. Zhao, X. Zhao, X. Li, S. Liu, B. Bross, P. Keydel, H. Schwarz, D. Marpe, and T. Wiegand. Multiple reference line coding for most probable modes in intra prediction. In *2019 Data Compression Conference (DCC)*, pages 559–559, March 2019.

[8] Seung Hwan Kim Jianle Chen, Yan Ye. Algorithm description for Versatile Video Coding and test model 11 (VTM). *Document JVET-T2002*, teleconference, October 2020.

[9] J. Pfaff, B. Stallenberger, M. Schäfer, P. Merkle, P. Helle, T. Hinz, H. Schwarz, D. Marpe, and T. Wiegand. Affine linear weighted intra prediction. *Document JVET-N0217*, Geneva, Switzerland, March 2019.

[10] Yu Han, Han Huang, and Zhang Yan. Ce4.1.3: Affine motion compensation prediction. *Document JVET-K0337*, Ljubljana, Slovenia, July 2018.

[11] Minhua Zhou and Brian Heng. Non-ce4: A study on the affine merge mode. *Document JVET-K0052*, Ljubljana, Slovenia, July 2018.

[12] Yan Ye Xiaoyu Xiu, Yuwen He. Ce9-related: Complexity reduction and bit-width control for bi-directional optical flow (bio). *Document JVET-L0256*, Macao, October 2018.

[13] J. Chen, M. Karczewicz, Y. Huang, K. Choi, J-R. Ohm, and G. Sullivan. The joint exploration model (jem) for video compression with capability beyond hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[14] Ru-Ling Liao. Ce10.3.1.b: Triangular prediction unit mode. *Document JCTVC-L0124*, Macao, October 2018.

[15] Zhuoyi Lv, Yinji Piao, Yue Wu, Kiho Choi, and Kwang Pyo Choi. Scan region-based coefficient coding in avs3. *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–5, 2020.

[16] M. Koo, M. Salehifar, J. Lim, and S. Kim. Reduced secondary transform. *Document JVET-N0193*, Geneva, Switzerland, March 2019.

[17] H. Schwarz, N. Tung., D. Marpe, and T. Wiegand. Transform coefficient coding and dependent quantization. *Document JVET-K0071*, Ljubljana, Slovenia, July 2018.

[18] Taoran Lu, Fangjun Pu, Peng Yin, Sean McCarthy, Walt Husak, Tao Chen, Edouard Francois, Christophe Chevance, Franck Hiron, Jie Chen, Ru-Ling Liao, Yan Ye, and Jiancong Luo. Luma mapping with chroma scaling in versatile video coding. In *2020 Data Compression Conference (DCC)*, pages 193–202, 2020.

[19] Cisco Visual networking Index. Forecast and methodology, 2018-2023, white paper. *San Jose, CA, USA*, 1, 2018.

[20] M. Efoui-Hess. Climate crisis: The un- sustainable use of online video. July 2019.

[21] G. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[22] D. Liu, Z. Chen, S. Liu, and F. Wu. Deep learning-based technology in responses to the joint call for proposals on video compression with capability beyond hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[23] J. Pfaff, H. Schwarz, D. Marpe, B. Bross, S. De-Luxán-Hernández, P. Helle, C. Helmrich, T. Hinz, W. Lim, J. Ma, et al. Video compression using generalized binary partitioning, trellis coded quantization, perceptually optimized encoding, and advanced prediction and transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[24] X. Xiu, P. Hanhart, Y. He, Y. Ye, R. Vanam, T. Lu, F. Pu, and P. Yin. A unified video codec for SDR, HDR, and 360 degree video applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[25] G. J. Sullivan and T. Wiegand. Video compression - from concepts to the h.264/avc standard. *Proceedings of the IEEE*, 93(1):18–31, January 2005.

[26] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J.-R. Ohm, and G. J. Sullivan. Video quality evaluation methodology and verification testing of hevc compression performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):76–90, 2016.

[27] A. Wieckowski, G. Hege, C. Bartnik, C. Lehmann, C. Stoffers, B. Bross, and D. Marpe. Towards a live software decoder implementation for the upcoming versatile video coding (vvc) codec. *IEEE International Conference on Image Processing (ICIP)*, page 3124–3128, October 2020.

[28] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C.S. Boon. Inter frame coding with template matching spatio-temporal prediction. *Proc. SPIE Visual Communications and Image Processing (VCIP)*, 2004.

[29] S. Kamp, J. Ballé, and M. Wien. Multihypothesis prediction using decoder side motion vector derivation in inter frame video coding. *IEEE International Conference on Image Processing (ICIP)*, page 465–468, 2009.

[30] T. K. Tan, C. S. Boon, and Y. Suzuki. Intra prediction by template matching. *IEEE International Conference on Image Processing (ICIP)*, page 1693–1696, 2006.

[31] J.-M. Thiesse. *Codage Vidéo Flexible par Association d'un Décodeur Intelligent et d'un Encodeur Basé Optimisation Débit-Distorsion.* Phd thesis, Nice Sophia-Antipolis University, April 2012.

[32] Algorithm description of joint exploration test model 7 (jem 7). *JVET-G1001*, July 2017.

[33] Description of sdr, hdr and 360° video coding technology proposal by interdigital communications and dolby laboratories. *JVET-J0018*, San Diego, April 2018.

[34] Description of sdr video coding technology proposal by mediatek. *JVET-J0018*, San Diego, April 2018.

[35] D. Ma, F. Zhang, and D. R. Bull. Bvi-dvc: A training database for deep video compression. *arXiv preprint*, (2003.13552), 2020.

[36] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[37] Joint call for proposals on video compression with capability beyond hevc. *JVET-H1002*, Macau October 2017.

[38] Video codec for audiovisual services at p x 64 kbit/s. *Recommendation ITU-T H.261*, 1993.

[39] Codecs for videoconferencing using primary digital group transmission. *Recommendation ITU-T H.120*, 1993.

[40] Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s—part 2: Video. volume 1. ISO/IEC, 1993.

[41] Information technology—generic coding of moving pictures and associated audio information: Video. Number 13818-2. ITU-T and ISO/IEC JTC 1, 1995.

[42] Video coding for low bit rate communication. ITU-T, Mar. 1996.

[43] Information technology—coding of audio-visual objects—part 2: Visual. Number 14496-2. ISO/IEC JTC 1, 2001.

[44] J. Pfaff et al. Data-driven intra-prediction modes in the development of the versatile video coding standard,. In *ITU J. ICT Discoveries*, volume 3, May 2020.

[45] Information technology—digital compression and coding of continuous-tone still images—part 1: Requirements and guidelines. In *Recommendation ITU-T T.81 and ISO/IEC 10918-1*. ITU-T and ISO/IEC JTC 1, 1992.

[46] Versatile supplemental enhancement information messages for coded video bitstreams. *Recommendation ITU-T H.274 and ISO/IEC 23002-7 (VSEI)*, Jul. 2020.

[47] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan. The joint exploration model (jem) for video compression with capability beyond hevc. *IEEE Trans. Circuits Syst. Video Technol.*, 30(5):1208–1225, 2020.

[48] B. Bross et al. General video coding technology in responses to the joint call for proposals on video compression with capability beyond hevc. *IEEE Trans. Circuits Syst. Video Technol.*, 30(5):1226–1240, May 2020.

[49] H. S. Malvar, G. J. Sullivan, and S. Srinivasan. Lifting-based reversible color transformations for image compression,. *Proc. SPIE*, 7073(707307), Aug. 2008.

[50] Procedure for the allocation of itu-t defined codes for non-standard facilities. *document Recommended ITU-T T.35*, 1988.

[51] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring. Jvet common test conditions and software reference configurations for sdr video,. *document JVET-N1010*, Mar. 2019.

[52] G. Bjontegaard. Calculation of average PSNR differences between rd-curves. *VCEG-M33*, 2001.

[53] Working practices using objective metrics for evaluation of video coding efficiency experiments,. *document ITU-T HSTP-VID-WPOM and ISO/ IEC DTR 23002-8*, Mar. 2020.

[54] Vvc reference softwareversion 8.0. accessed. Feb. 2020.

[55] Hevc reference softwareversion 16.20. accessed:. Sep. 2018.

[56] Hevc screen content coding extension reference software version 16.21+scm8.8.accessed. Mar. 2020.

[57] Avc reference software version 19.0. accessed:. Mar. 2019.

[58] J.-R. 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),. volume 22, page 1669–1684, Dec. 2012.

[59] V. Baroncini and M. Wien. Vvc verification test report for uhd sdr video content. *document JVET-T2020*, 21th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Oct. 2020.

[60] Fraunhofer hhi vvenc softwarerepository. Accessed: Sep. 2020.

[61] A. Wieckowski et al. Open optimized vvc encoder (vvenc) and decoder (vvdec) implementations. *document JVET-T0099*, 21th Meeting of ITU-T/ISO/IEC Joint Video Experts Team (JVET), Oct. 2020.

[62] Stephen Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011.

[63] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference, and prediction, 2nd edition. In *Springer Series in Statistics*, 2009.

[64] Miroslav Kubat. *An Introduction to Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[65] Ethem Alpaydın. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning series. The MIT Press, 3 edition, 2014.

[66] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[67] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[68] Andreas Spanias, Ted Painter, and Venkatraman Atti. *Audio Signal Processing and Coding*. John Wiley and Sons, Dec 2005.

[69] Elena Ularu, Florina Puican, George Suciu, Alexandru Vulpe, and Gyorgy Todoran. Mobile computing and cloud maturity - introducing machine learning for erp configuration automation. *Informatica Economică*, 17:40–52, 03 2013.

[70] Josep Lluis Berral-García. A quick view on current techniques and machine learning algorithms for big data analytics. In *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2016.

[71] Shun ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196, 1993.

[72] Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip ŕZeleznỳ. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, volume 8188. Springer, 2013.

[73] V.J. Mathews and Z. Xie. A stochastic gradient adaptive filter with gradient adaptive step size. *IEEE Transactions on Signal Processing*, 41(6):2075–2087, 1993.

[74] Fetty Fitriyanti Lubis, Yusep Rosmansyah, and Suhono Harso Supangkat. Gradient descent and normal equations on cost function minimization for online predictive using linear regression with multiple variables. In *2014 International Conference on ICT For Smart Society (ICISS)*, pages 202–205, 2014.

[75] Yingyan Chen, Hongze Wang, Yi Wu, and Haowei Wang. Predicting the printability in selective laser melting with a supervised machine learning method. *Materials*, 13(22), 2020.

[76] Matthew Tivnan, Carey Rappaport, Marc Lambert, and Dominique Lesselier. A modified gradient descent reconstruction algorithm for breast cancer detection using microwave radar and digital breast tomosynthesis. In J. R. Mosig, editor, *10th European Conference on Antennas and Propagation - (EUCAP 2016)*, Procedings of 10th European Conference on Antennas and Propagation (EuCAP), Davos, Switzerland, April 2016.

[77] Guannan Qu and Na Li. Accelerated distributed nesterov gradient descent for smooth and strongly convex functions. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 209–216, 2016.

[78] Thi-Thu-Hong Phan, Emilie Poisson Caillault, and André Bigand. Comparative study on supervised learning methods for identifying phytoplankton species. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 283 – 288, Ha Long, Vietnam, July 2016.

[79] Sandra Heleno, Margarida Silveira, Magda Matias, and Pedro Pina. Assessment of supervised methods for mapping rainfall induced landslides in VHR images. In *2015 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2015, Milan, Italy, July 26-31, 2015*, pages 850–853. IEEE, 2015.

[80] P. Drotar and Z. Smeakal. Comparative study of machine learning techniques for supervised classification of biomedical data. *Acta Electrotechnica et Informatica*, 14:5–10, September 2014.

[81] F. Galton. Natural inheritance. In *Proc Royal Soc.y of London*, 1989.

[82] F. Galton. Anthropological miscellanea: "regression towards mediocrity in hereditary stature". *The Journal of the Anthropological Institute of Great Britain and Ireland*, page 246–263, 1886.

[83] F. Galton. Co-relations and their measurement. *chiefly from anthropometric data, Proc.s Royal Soc. of London*, pages 135–145, 1889.

[84] G. A. F. Seber, A. J. Lee, and R. A. Lee. Linear regression analysis. *2nd ed. New York, NY, United States: Wiley, John Sons*, 2003.

[85] André I. Khuri. Introduction to linear regression analysis, fifth edition by douglas c. montgomery, elizabeth a. peck, g. geoffrey vining. *International Statistical Review*, 81(2):318–319, 2013.

[86] H. Motulsky and A. Christopoulos. Fitting models to biological data using linear and nonlinear regression: A practical guide to curve fitting. 2004.

[87] PR Chandler, M Pachter, and M Mears. Constrained linear regression for flight control system failure identification. In *1993 American Control Conference*, pages 3141–3145. IEEE, 1993.

[88] E. Masry. Multivariate regression estimation of continuous-time processes from sampled data: Local polynomial fitting approach. *IEEE Trans. Inf. Theor.*, 45(6):1939–1953, Sept 2006.

[89] T. Banerjee, D. Wang, B. Xie, and D. P. Agrawal. Perd: Polynomial-based event region detection in wireless sensor networks. In *2007 IEEE International Conference on Communications*, pages 3307–3312, 2007.

[90] Joseph M Hilbe. *Logistic regression models*. Chapman and hall/CRC, 2009.

[91] Jeffry White. Logistic regression model effectiveness: Proportional chance criteria and proportional reduction in error. *Journal of Contemporary Research in Education*, 2:4–10, 08 2013.

[92] N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. 2000.

[93] Andreas Christmann and Ingo Steinwart. *Support Vector Machines*. 01 2008.

[94] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[95] Wei Wu, Srikantan Nagarajan, and Zhe Chen. Bayesian machine learning: Eeg\/meg signal processing measurements. *IEEE Signal Processing Magazine*, 33(1):14–36, 2015.

[96] Kian Ming Adam Chai, Hai Leong Chieu, and Hwee Tou Ng. Bayesian online classifiers for text classification and filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 97–104, 2002.

[97] R. Vedala and B. R. Kumar. An application of naive bayes classification for credit scoring in e-lending platform. *2012 International Conference on Data Science & Engineering (ICDSE)*, pages 81–84, 2012.

[98] Eun-Hye Jang, Byoung-Jun Park, Sang-Hyeob Kim, Youngji Eum, and Jin-Hun Sohn. A study on analysis of bio-signals for basic emotions classification: Recognition using machine learning algorithms. In *2014 International Conference on Information Science Applications (ICISA)*, pages 1–4, 2014.

[99] A. Sharmila and P. Geethanjali. Dwt based detection of epileptic seizure from eeg signals using naive bayes and k-nn classifiers. *IEEE Access*, 4:7716–7727, 2016.

[100] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *IEEE Trans. Inform. Theory*, IT-13:21–27, Jan. 1967.

[101] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[102] Yury Lifshits. Nearest neighbor search: Algorithmic perspective. *SIGSPATIAL Special*, 2(2):12–15, July 2010.

[103] Visar Berisha, Alan Wisler, Alfred O. Hero, and Andreas Spanias. Empirically estimable classification bounds based on a nonparametric divergence measure. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 64(3):580–591, Feb 2016.

[104] Adnan Ghaderi, Javad Frounchi, and Alireza Farnam. Machine learning-based signal processing using physiological signals for stress detection. pages 93–98, 11 2015.

[105] Memoona Khanam, Tahira Mahboob, Warda Imtiaz, Humaraia Ghafoor, and Rabeea Sehar. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications*, 119:34–39, 06 2015.

[106] M. E. Celebi and K. Aydin. Unsupervised learning algorithms. 2016.

[107] Analp Pathak Aayushi Bindal. A survey on k-means clustering and web-text mining. *International Journal of Science and Research (IJSR)*, 5(4):1049 – 1052, 04 2016.

[108] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007.

[109] Kumar Dhiraj and Santanu Rath. Gene expression analysis using clustering. *International Journal of Computer and Electrical Engineering*, pages 155–164, 01 2009.

[110] Anindya Bhattacharya and Rajat K. De. Bi-correlation clustering algorithm for determining a set of co-regulated genes. *Bioinformatics*, 25(21):2795–2801, 09 2009.

[111] Erliang Zeng, Chengyong Yang, Tao Li, and Giri Narasimhan. Clustering genes using heterogeneous data sources. *International Journal of Knowledge Discovery in Bioinformatics (IJKDB)*, 1(2):12–28, 2010.

[112] Ashwini A Mandwe and Anisa Anjum. Detection of brain tumor using k-means clustering.

[113] Sundar Chinnasamy. An analysis on the performance of k-means clustering algorithm for cardiotocogram data clustering. *International Journal on Computational Science Applications*, 2:11–20, 10 2012.

[114] Xuan Huang and Zhijun Song. Clustering analysis on e-commerce transaction based on k-means clustering. *Journal of Networks*, 9(2):443, 2014.

[115] Jayaraman J. Thiagarajan, Karthikeyan Natesan Ramamurthy, Pavan Turaga, and Andreas Spanias. Image understanding using sparse representations. *Synthesis Lectures on Image, Video, and Multimedia Processing*, 15:1–120, Apr 2014.

[116] Alberto Albiol, Luis Torres, and Edward J Delp. An unsupervised color image segmentation algorithm for face detection applications. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 2, pages 681–684. IEEE, 2001.

[117] Chih-Wen Wang and Jyh-Horng Jeng. Image compression using pca with clustering. In *2012 International Symposium on Intelligent Signal Processing and Communications Systems*, pages 458–462. IEEE, 2012.

[118] Xiangyu Bi, Sungyun Lee, James F. Ranville, Prasanna Sattigeri, Andreas Spanias, Pierre Herckes, and Paul Westerhoff. Quantitative resolution of nanoparticle sizes using single particle inductively coupled plasma mass spectrometry with the k-means clustering algorithm. *Journal of Analytical Atomic Spectrometry*, 29(9):1630–1639, Sept 2014.

[119] Sunil Rao, David Ramirez, Henry Braun, Jongmin Lee, Cihan Tepedelenlioglu, Elias Kyriakides, Devarajan Srinivasan, Jeffrey Frye, Shinji Koizumi, Yoshitaka Morimoto, and Andreas Spanias. An 18 kw solar array research facility for fault detection experiments. In Constandinos Mavromoustakis, Soulla Louca, Constantinos S. Pattichis, Julius Georgiou, Despina Michael, A. Paschalidou, Efthyvoulos Kyriacou, Vasos Vassiliou, Christos Panayiotou, Elias Kyriakides, Georgios Ellinas, George Hadjichristofi, and C. Loizou, editors, *Proceedings of the 18th Mediterranean Electrotechnical Conference*. Institute of Electrical and Electronics Engineers Inc., Jun 2016.

[120] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[121] Andreas Spanias, Jennifer Blain Christen, Trevor Thornton, Karen Anderson, Michael Goryll, Hany Arafa, Uday Shankar Shanthamallu, Erica Forzani, Heather Ross, and Wendy Barnard. Board 78 : The sensor signal and information processing reu site. 06 2018.

[122] Mohit Shah, Chaitali Chakrabarti, and Andreas Spanias. Within and cross-corpus speech emotion recognition using latent topic model-based features. *Eurasip Journal on Audio, Speech, and Music Processing*, 2015(1), 2015.

[123] Ezgi Can Ozan, Ekaterina Riabchenko, Serkan Kiranyaz, and Moncef Gabbouj. A vector quantization based k-nn approach for large-scale image classification. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2016.

[124] Diego Valsesia and Petros T Boufounos. Multispectral image compression using universal vector quantization. In *2016 IEEE Information Theory Workshop (ITW)*, pages 151–155. IEEE, 2016.

[125] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[126] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[127] Prasanna Sattigeri, Jayaraman J. Thiagarajan, Karthikeyan N. Ramamurthy, and Andreas Spanias. Implementation of a fast image coding and retrieval system using a gpu. In *2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012 - Proceedings*, 2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012 - Proceedings, pages 5–8, 2012. 2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012 ; Conference date: 12-01-2011 Through 14-01-2011.

[128] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[129] A. Courville Y. Bengio, I. J. Goodfellow. Deep learning. *Nature*, 521:436–444, 2015.

[130] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[131] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07-09 Jul 2015. PMLR.

[132] Devansh Arpit, Yingbo Zhou, Bhargava Urala Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. 03 2016.

[133] Tim Salimans and Diederik Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. 02 2016.

[134] Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[135] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[136] R. D. Dony and S. Haykin. Neural network approaches to image compression. *Proceedings of the IEEE*, 83(2):288–303, 1995.

[137] J. Jiang. Image compression with neural networks–a survey. *Signal Processing: Image Communication*, 14(9):737–760, 1999.

[138] Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. Deepcoder: A deep neural network based video compression. *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, 2017.

[139] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[140] Convolutional neural network-based fractional-pixel motion compensation. *Transactions on Circuits and Systems for Video Technology)*, 29(3):840–853, 2019.

[141] Jing Cui, Tao Zhang, Chenchen Gu, Xinfeng Zhang, and Siwei Ma. Gradient-based early termination of cu partition in vvc intra coding. In *2020 Data Compression Conference (DCC)*, pages 103–112, 2020.

[142] B. Shen and B. Heng. Decoding side intra-prediction derivation for video coding, May 2012. US Patent App. 12/945,949.

[143] X. Xiu, Y. He, and Y. Ye. Decoder-side intra mode derivation for block-based video coding. In *2016 Picture Coding Symposium (PCS)*, pages 1–5, 2016.

[144] Anis BenHajyoussef, Tahar Ezzedine, and Ammar Bouallègue. Gradient-based pre-processing for intra prediction in high efficiency video coding. *EURASIP Journal on Image and Video Processing*, 2017(1):9, 2017.

[145] A. Nasrallah, M. Abdoli, E. Mora, T. Guionnet, and M. Raulet. Decoder-side intra mode derivation with texture analysis in VVC test model (VTM). In *2019 International Conference on Image Processing (ICIP)*, pages 1–5. IEEE, 2019.

[146] T. Wedi. Motion compesation in h.264/avc. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(11):957–969, 2003.

[147] K. U. e. al. Motion compensated prediction and interpolation filter design in h.265/hevc. *IEEE journal of selected topics in signal processing*, 7(6), 2013.

[148] Cisco Visual networking Index. High efficiency video coding (hevc) text specifications. *JCTVC-I1003, San Jose, CA, USA, 2012*, 2012.

[149] S. Wittmann and T. Wedi. Separable adaptive interpolation filter for video coding. *15th IEEE International Conference on Image Processing (ICIP)*, page 2500–2503, October 2008.

[150] M. Karczewicz, Y. Ye, and P. Chen. Switched interpolation filter with offset. *Document VCEG-AI35*, July 2008.

[151] Y. Ye, G. Motta, and M. Karczewicz. Enhanced adaptive interpolation filters for video coding. *Data Compression Conference (DCC)*, page 435–444, March 2010.

[152] N. Yan, D. Liu, H. Li, and F. Wu. A convolutional neural network approach for half-pel interpolation in video coding. *International Symposium on Circuits and Systems (ISCAS)*, page 1–4, 2017.

[153] One-for-all: Grouped variation network based fractional interpolation in video coding. *IEEE Transactions on Image Processing*, 28(5):2140–2151, 2019.

[154] Learning a convolutional neural network for fractional interpolation in hevc inter coding. *Visual Communications and Image Processing (VCIP)*, page 1–4, 2017.

[155] Convolutional neural network-based invertible half-pixel interpolation filter for video coding. *International Conference on Image Processing (ICIP)*, page 201–205, 2018.

[156] "neural network based inter prediction for hevc. *Integrated computational materials engineering (ICME)*, pages 1–6, 2018.

[157] X. Zhang, W. Zhao, L. Shen, Z. Liu, and Z. Zhang. An effective cu size decision method for hevc encoders. *IEEE transactions*, 2012.

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

[159] T. L. Da Silva, L. V. Agostini, and L. A. da Silva Cruz. Fast hevc intra prediction mode decision based on edge direction information. In *Proc. European Signal Processing Conference*. IEEE, 2012.

[160] F. Sampaio, S. Bampi, M. Grellert, L. Agostini, and J. Mattos. Motion vectors merging: low complexity prediction unit decision heuristic for the inter-prediction of hevc encoders. In *Proc. IEEE International Conference on Multimedia and Expo. IEEE*. IEEE, 2012.

[161] Jens Brandenburg, Adam Wieckowski, Tobias Hinz, Anastasia Henkel, Valeri George, Ivan Zupancic, Christian Stoffers, Benjamin Bross, Heiko Schwarz, and Detlev Marpe. Towards fast and efficient vvc encoding. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2020.

[162] Gerhard Tech, Jonathan Pfaff, Heiko Schwarz, Philipp Helle, Adam Wieckowski, Detlev Marpe, and Thomas Wiegand. Cnn-based parameter selection for fast vvc intra-picture encoding. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2109–2113, 2021.

[163] Ting Fu, Hao Zhang, Fan Mu, and Huanbang Chen. Fast cu partitioning algorithm for h.266/vvc intra-frame coding. *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 55–60, 2019.

[164] Hao Yang, Liquan Shen, Xinchao Dong, Qing Ding, P. An, and G. Jiang. Low-complexity ctu partition structure decision and fast intra mode decision for versatile video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 30:1668–1682, 2020.

[165] Fen Chen, Yan Ren, Zongju Peng, Gangyi Jiang, and Xin Cui. A fast cu size decision algorithm for vvc intra prediction based on support vector machine. *Multimedia Tools and Applications*, 79(37):27923–27939, 2020.

[166] Meng Lei, Falei Luo, Xiang Zhang, Shanshe Wang, and Siwei Ma. Look-ahead prediction based coding unit size pruning for vvc intra coding. *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4120–4124, 2019.

[167] Ting Fu, Hao Zhang, Fan Mu, and Huanbang Chen. Two-stage fast multiple transform selection algorithm for vvc intra coding. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 61–66, 2019.

[168] Mário Saldanha, Gustavo Sanchez, César Marcon, and Luciano Agostini. Fast partitioning decision scheme for versatile video coding intra-frame prediction. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[169] Thomas Amestoy, Alexandre Mercat, Wassim Hamidouche, Daniel Menard, and Cyril Bergeron. Tunable vvc frame partitioning based on lightweight machine learning. *IEEE Transactions on Image Processing*, 29:1313–1328, 2020.

[170] Alexandre TISSIER, Wassim Hamidouche, J Vanne, F Galpin, and Daniel Menard. CNN ORIENTED COMPLEXITY REDUCTION OF VVC INTRA ENCODER. In *IEEE International Conference on Image Processing (ICIP 2020)*, 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, October 2020. IEEE, IEEE.

[171] Jinchao Zhao, Yihan Wang, and Qiuwen Zhang. Adaptive cu split decision based on deep learning and multifeature fusion for h.266/vvc. *Sci. Program.*, 2020:8883214:1–8883214:11, 2020.

[172] Tianyi Li, Mai Xu, Runzhi Tang, Ying Chen, and Qunliang Xing. Deepqtmt: A deep learning approach for fast qtmt-based cu partition of intra-mode vvc. *IEEE Transactions on Image Processing*, 30:5377–5390, 2021.

[173] Naima Zouidi, Fatma Belghith, Amina Kessentini, and Nouri Masmoudi. Complexity reduction of versatile video coding standard: a deep learning approach. *Journal of Electronic Imaging*, 30(2):1 – 22, 2021.

[174] Mourad Aklouf, Marc Leny, Michel Kieffer, and Frédéric Dufaux. Low complexity versatile video coding (vvc) for low bitrate applications. *8th European Workshop on Visual Information Processing (EUVIP 2019)*, Oct 2019, Rome, Italy.

[175] Jens Brandenburg, Adam Wieckowski, Tobias Hinz, and Benjamin Bross. Vvenc, fraunhofer versatile video encoder.

[176] Fraunhofer hhi vvenc v1.0.0 whitepaper. [Online]. Available:.

[177] C. R. Helmrich, I. Zupancic, J. Brandenburg, V. George, A. Wieckowski, and B. Bross. Visually optimized two-pass rate control for video coding using the low-complexity xpsnr model. *submitted to VCIP'21*.

[178] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring. Open gop resolution switching in http adaptive streaming with vvc. *35th Picture Coding Symposium (PCS)*, Bristol, US, June-July 2021.

[179] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring. Jvet common test conditions and software reference configurations for sdr video. *doc. JVET-T2010 of ITU-T/ISO/IEC Joint Video Experts Team (JVET)*, October 2020.

[180] J. Boyce, K. Suehring, X. Li, and V. Seregin. JVET common test conditions and software reference configurations. *Document JVET-J1010*, Ljubljana, Slovenia, July 2018.

[181] B. Bross, J. Chen, and S. Liu. Versatile video coding (VVC) draft 4. *Document JVET-M1001*, Geneva, Switzerland, March 2019.

[182] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[183] G. Kulupana, A. Seixas Dias, and S. Blasi. Combined-hypothesis intra-prediction with unified intra mode coding. *Document JVET-N0248*, Geneva, Switzerland, March 2019.

[184] H. Yang, H. Wang, Y. Chen, J. Liang, and L. Yu. Multi-weights intra prediction with double reference lines. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3562–3566, 2019.

[185] Y. Ye and M. Karczewicz. Improved h.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning. In *2008 15th IEEE International Conference on Image Processing*, pages 2116–2119, 2008.

[186] T. Shiodera, A. Tanizawa, A. Chujoh, and T. Yamakage. CE6 subset A: Bidirectional intra prediction. *Document JCTVC-D0108*, Daegu, South Korea, January 2011.

[187] Pierre Wickramarachi. Effects of windowing on the spectral content of a signal. *Sound and vibration*, 37(1):10–13, 2003.

[188] G. Sullivan. Color format downconversation for test sequence generation. *Document: JVT-I018*, San Diego, 2-5 September 2003.

[189] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 86(11):2278–2324, 1998.

[190] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection, 2015.

[191] Sonali B. Wankhede. Analytical study of neural network techniques: Som, mlp and classifier-a survey. *IOSR Journal of Computer Engineering*, 16:86–92, 2014.

[192] G. Laroche, J. Taquet, C. Gisquet, and P. Onno (Canon). Cross-component linear model simplification (with min and max values). *Document JVET-L0191*, LMacao, October 2018.

[193] A. K. Ramasubramonian, G. Van der Auwera, L. Pham Van, and M. Karczewicz. Non-ce3: Dc intra prediction mode alignment. *Document JVET-O0426*, Gotenburg, July 2019.

[194] B. Heng (Broadcom) T. Hellman, M. Zhou. Non-ce3: Cleanup of mrlp line storage. *Document JVET-P0418*, Geneva, October 2019.

[195] A. K. Ramasubramonian G. Van der Auwera T. Hsieh V. Seregin L. Pham Van M. Karczewicz (Qualcomm) S. De-Luxan Hernandez B. Bross T. Nguyen V. George B. Stabernack H. Schwarz D. Marpe T. Wiegand (HHI). Ce3-1.6: On 1xn and 2xn subblocks of isp. *Document JVET-O0106*, Gotenburg, July 2019.

[196] J. Lee J. Kang H. Lee S.-C. Lim H. Y. Kim (ETRI). Non-ce3: Unification of intra interpolation filter selection. *Document JVET-O0341*, Gotenburg, July 2019.

[197] S. De-Luxán-Hernández V. George G. Venugopal J. Brandenburg B. Bross T. Nguyen H. Schwarz D. Marpe T. Wiegand (HHI). Non-ce3: Proposed isp cleanup. *Document JVET-O0502*, Gotenburg, July 2019.

[198] A. Segall, E. François, W. Husak, S. Iwamura, and D. Rusanovskyy. Jvet common test conditions and evaluation procedures for hdr/wcg video. *document JVET-P2011*, 1–11 Oct. 2019.

[199] Conversion and coding practices for hdr/wcg ycbcr 4:2:0 video with pq transfer characteristics. In *Supplement ITU-T H.Sup15*, 2017.

[200] W.-J. Chien, J. Boyce, W. Chen, Y.-W. Chen, R. Chernyak, K. Choi, R. Hashimoto, Y.-W. Huang, H. Jang, R.-L. Liao, and S. Liu. Jvet ahg report: Tool procedure and testing (ahg13). *document JVET-Q0013*, 17th Meeting: Brussels, BE, 7–17 January 2020.

[201] J. V. Neumann. *Functional Operators*, volume 2. 1950.

[202] Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

[203] T. Biatek, M. Abdoli, T. Guionnet, A. Nasrallah, and M. Raulet. Future mpeg standards vvc and evc: 8k broadcast enabler. *IBC TECHNICAL PAPERS*, 2020.