HAL
open science

A compact video representation format based on spatio-temporal linear embedding and epitome

Martin Alain

To cite this version:

Martin Alain. A compact video representation format based on spatio-temporal linear embedding and epitome. Signal and Image processing. Université de Rennes, 2016. English. NNT : 2016REN1S001 . tel-01261590v2

HAL Id: tel-01261590

https://inria.hal.science/tel-01261590v2

Submitted on 7 Mar 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ANNÉE 2016

THÈSE / UNIVERSITÉ DE RENNES 1

sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

École doctorale Matisse

présentée par

Martin ALAIN

préparée au centre de recherche INRIA Rennes - Bretagne Atlantique
et à Technicolor R&I

A compact video representation format based on spatio-temporal linear embedding and epitome

Thèse soutenue à Rennes le 12 Janvier 2016

devant le jury composé de :

Luce MORIN
Professor, INSA Rennes / *Examinatrice*

Stefano TUBARO
Professor, Politecnico di Milano / *Rapporteur*

Frédéric DUFAUX
Research director, Télécom ParisTech / *Rapporteur*

William PUECH
Professor, LIRMM / *Examinateur*

Philippe GUILLOTEL
Distinguished Scientist, Technicolor / *Examinateur*

Christine GUILLEMOT
Research director, INRIA / *Directrice de thèse*

Dominique THOREAU
Senior Scientist, Technicolor / *Co-directeur de thèse*

Contents

Contents **4**

Remerciements **5**

Résumé en français **7**

Introduction **17**

1 State-of-the-art in video compression **21**
1.1 Main principles in video compression 21
1.1.1 Prediction 21
1.1.2 Transform 22
1.1.3 Quantization 22
1.1.4 Entropy coding 22
1.2 Standards 23
1.2.1 H.264/Advanced Video Coding (AVC) 23
1.2.1.1 Hierarchical syntax 23
1.2.1.2 Types of frame 24
1.2.1.3 Encoder structure 24
1.2.1.4 Intra prediction 25
1.2.1.5 Inter prediction 26
1.2.1.6 Coding mode selection 28
1.2.2 HEVC 28
1.2.2.1 Hierarchical syntax 29
1.2.2.2 Types of frame 30
1.2.2.3 Encoder structure 30
1.2.2.4 Intra prediction 31
1.2.2.5 Inter prediction 32
1.2.2.6 Coding mode selection 32
1.2.3 SHVC 32

2 Inter prediction methods based on linear embedding for video compression **35**
2.1 Background: Prediction based on Template Matching 37

2.2 Inter prediction based on LLE . . . 38
2.2.1 LLE-based predictor computation . . . 38
2.2.2 Template-based K-NN search for LLE-based prediction . . . 39
2.2.3 Block-based K-NN search for LLE-based prediction . . . 40
2.2.4 Optimized template-based K-NN search for LLE-based prediction 41
2.3 Simulations and results . . . 41
2.3.1 Integration in the MPEG-4 AVC/H.264 coding scheme . . . 41
2.3.2 Experimental conditions . . . 44
2.3.3 RD performance analysis & elements of complexity . . . 45
2.3.3.1 Impact of the parameter K . . . 45
2.3.3.2 Impact of the reference frames number . . . 47
2.3.3.3 Impact of the search window size . . . 49
2.3.4 Combining Intra and Inter LLE-based prediction methods . . . 50
2.4 Conclusion and perspectives . . . 54

3 Epitome-based quantization noise removal 57
3.1 Background . . . 58
3.1.1 Epitome generation . . . 58
3.1.1.1 Generative models . . . 58
3.1.1.2 Factoring similar content within an image . . . 59
3.1.1.3 Epitome dedicated to image coding . . . 62
3.1.2 De-noising . . . 64
3.1.2.1 Optimal Wiener filter . . . 65
3.1.2.2 Transform domain thresholding . . . 66
3.1.2.3 Non Local Mean . . . 66
3.1.2.4 BM3D . . . 67
3.2 Clustering-based methods for fast epitome generation . . . 71
3.2.1 List-based method for self-similarities search . . . 72
3.2.2 Threshold-based clustering for self-similarities search . . . 73
3.2.3 Trade-off between complexity and quality . . . 74
3.2.4 Simulations and results . . . 74
3.2.5 Conclusion . . . 77
3.3 Epitome-based quantization noise removal . . . 79
3.3.1 Overview of the proposed method . . . 79
3.3.2 Epitome generation . . . 80
3.3.3 Epitome encoding . . . 82
3.3.4 Epitome-based de-noising . . . 83
3.3.4.1 Epitome-based Local Linear Combination . . . 83
3.3.4.2 Epitome-based BM3D . . . 85
3.3.4.3 Epitome-based Local Linear Mapping . . . 88
3.3.5 Simulation and results . . . 89
3.3.5.1 De-noising with source epitome . . . 89
3.3.5.2 RD performances in a single layer coding scheme . . . 93
3.3.5.3 RD performances in a SNR scalable coding scheme . . . 96

3.3.5.4 RD performances in a spatial scalable coding scheme . . 96
3.4 Conclusions and perspectives 97

4 Clustering-based quantization noise removal 101
4.1 Background 102
4.1.1 Un-supervised learning 102
4.1.1.1 K-means algorithm 102
4.1.1.2 Gaussian Mixture Model (GMM) 102
4.1.1.3 Spectral clustering 104
4.1.2 Supervised learning 104
4.1.2.1 Support Vector Machine 105
4.1.2.2 Decision Tree 108
4.2 Clustering-based linear mapping learning for quantization noise removal 111
4.2.1 Main idea and preliminary results with additive noise 111
4.2.2 Proposed scheme for quantization noise removal 112
4.2.3 Clustering 113
4.2.3.1 Choice of a clustering algorithm 113
4.2.3.2 Choice of a number of clusters K 114
4.2.4 Linear mappings encoding 116
4.2.5 Two steps scheme with first pass blind de-noising 118
4.2.6 Simulations and results 120
4.2.6.1 RD performances using the K-means clustering 120
4.2.6.2 RD performances using binary tree RDO for K selection 121
4.2.6.3 RD performances with first pass blind de-noising 122
4.3 Optimal clustering for de-noising based on linear mappings 124
4.3.1 K-means limitations for linear mappings learning 124
4.3.2 Optimal clustering for linear mapping learning 126
4.3.3 Potential RD performances using optimal clustering 132
4.3.4 Conclusion 133
4.4 Application to super-resolution for scalable video compression 134
4.4.1 Preliminary super-resolution results 134
4.4.2 Proposed scheme for salable compression 134
4.4.2.1 Super-resolution of the base layer 135
4.4.2.2 Super-resolution in a scalable scheme 136
4.4.3 Clustering 138
4.4.4 Linear mappings encoding 138
4.4.5 Simulations and results 138
4.5 From clustering to classification 140
4.5.1 Preliminary results: classification performances 142
4.5.1.1 Choice of the features 143
4.5.1.2 Dimensionality reduction using PCA 145
4.5.2 Application in the compression context 147
4.5.2.1 Transmission of a classifier model 147
4.5.2.2 Learning on a subset of data 149

4.5.2.3 RD performances . . . 150
4.5.3 Conclusion . . . 152
4.6 Conclusion and perspectives . . . 152

Conclusion **155**

Author's publications **159**

A Appendix of chapter 2 **161**

B Appendix of chapter 3 **162**
B.1 Comparison of the epitomes obtained with the different self-similarities search methods . . . 162
B.2 Epitomes for epitome-based de-noising . . . 167

C Appendix of chapter 4 **169**
C.1 De-noising . . . 169
C.1.1 Results on AWGN and SDN . . . 169
C.1.2 RD curves for optimal clustering . . . 173
C.2 Super-resolution . . . 174
C.3 Classification . . . 175
C.3.1 Test images . . . 175
C.3.2 Classification performances . . . 176
C.3.3 Classifications performances in a compression context . . . 178
C.3.4 RD performances . . . 180
C.3.4.1 De-noising . . . 180
C.3.4.2 Super-resolution . . . 181

Glossary **183**

Bibliography **194**
Contents

Remerciements

Tout d'abord, je tiens vivement à remercier Christine Guillemot, Dominique Thoreau et Philippe Guillotel, pour m'avoir encadré pendant ces trois années. Vos conseils et encouragements ont été précieux, et les nombreuses conversations que nous avons pu avoir ont nourri les travaux qui ont abouti à ce manuscrit. Je garderai un excellent souvenir de ces trois années à vos côtés, durant lesquelles j'ai beaucoup appris.

Je remercie également les membres du jury pour l'intérêt porté à mes travaux, ainsi que pour les commentaires et conseils que vous m'avez apportés. Merci donc aux Pr. Stefano Tubaro et Dr. Frédéric Dufaux d'avoir accepté d'être rapporteurs de cette thèse, au Pr. William Puech d'avoir accepté d'être examinateur, et enfin au Pr. Luce Morin d'avoir accepté de présider ce jury.

Merci aux collègues et amis de l'INRIA comme de Technicolor pour m'avoir accueilli et pour les bons moments passés ensembles, au travail comme à l'extérieur, vous êtes des machines. Merci donc, dans le désordre à : Mehmet, Fabien D., Fabien R., Dominique, Erik, Merwan, Matis, Ronan B., Tania, Darya, Fanny, Mikael, Alan, Ronan L.B., Marco, Raul, Simon, Steve, Antoine, Julio, Elif, Tom, Robin, Guillaume, Jérémy, Reuben, Thomas, Gilles, Jean, Mathieu, Thierry. Merci au grand gourou Harouna. Un grand merci également à Huguette Béchu pour son aide. Cette liste est non exhaustive, merci donc à ceux que j'ai oubliés et qui auraient du être cités ici. Merci aux mardis et marchés fous.

Merci à l'atelier "Douce Ambiance", notamment Julien, pour les pauses déjeuners musicales en compagnie de Django.

Un grand merci également aux amis rencontrés à Bordeaux pour les weekends plaisirieux : Nico, Antoine, David, Yohan, Manu, Amouda, Julie, Richard, FX, Didier, Clément, Maxime, Claire, Lucie, Michèle, et j'en oublie.

Merci à tous mes amis Rennais qui me soutenaient déjà avant que je ne commence cette thèse. Là encore il y a du monde à remercier et cette liste n'est sûrement pas exhaustive : merci donc à Raton, Adèle, Charles, Seb, Maryse, Evariste, Morgane, Lucie, Karen, Didine, Max, Gwen, Malet, Micka, Antoine, Christian, Nico, Alvin. Merci à Seb, Charles, Raton et Alvin pour les weekends musicaux avec ce groupe qui n'a toujours pas de nom.

Enfin je voudrai terminer par un grand merci à mes parents, mon frère, et toute ma famille, vous m'avez toujours soutenu et c'est aussi grâce à vous que je suis arrivé jusqu'ici.

Résumé en français

Introduction

Nous sommes de plus en plus demandeurs de contenus vidéos, tout en étant de plus en plus exigeants en termes de qualité de service, notamment concernant la qualité visuelle et la rapidité d'accès. Des services comme Youtube ou plus récemment Netflix sont emblématiques de cette évolution. En dépit de la loi de Moore et des puissances de calcul ainsi que des capacités de stockage croissantes qui en découlent, la compression des contenus vidéos reste cruciale pour assurer la qualité de service désirée. De plus, pour répondre à la demande d'amélioration constante de qualité visuelle de la part d'un public de plus en plus averti, de nouveaux formats voient le jour. On peut notamment citer la vidéo à haute résolution (4K - 8K), à gamme de couleur ou dynamique étendues, ou encore à fréquence d'images augmentée. Ces nouveaux formats représentent un volume de données considérable, qui doit être nécessairement comprimé. Dans un livre blanc publié en 2014, Cisco a prédit qu'en 2018 le trafic vidéo représenterait 79 % du trafic sur Internet. De plus, il est estimé que 80 % à 90 % du trafic global sera du contenu vidéo sous formes diverses (télévision, Internet, vidéo à la demande, vidéo sur IP, *peer to peer*). L'efficacité des services de compression est donc un enjeu essentiel, et le sera d'autant plus dans un futur proche. De ce point de vue, le standard MPEG HEVC est aujourd'hui un des schémas de compression les plus efficaces, et remplacera au fur et à mesure son prédécesseur H.264.

Les principes fondamentaux des standards modernes de compression vidéo sont la réduction des redondances spatiales et temporelles du signal en utilisant des outils de prédiction, l'utilisation d'une transformée afin de diminuer d'avantage les corrélations du signal, une quantification afin de réduire l'information non perceptible, et enfin un codage entropique pour prendre en compte les redondances statistiques du signal. Dans le cadre de la compression sans perte, la quantification n'est pas effectuée afin de pouvoir reconstruire parfaitement le signal d'origine. Ce type de compression est notamment utile dans le cadre de l'imagerie médicale, où la fidélité du signal est capitale pour établir un diagnostic. Toutefois, les travaux de cette thèse s'inscrivent dans le cadre de la compression avec perte, qui est utilisée en particulier dans les applications grand public mentionnées ci-dessus. Dans ce cas, les pertes viennent de l'étape de quantification, et le paramètre de quantification sert alors à réguler l'équilibre entre qualité visuelle et bande-passante.

Pour faire face aux multiples récepteurs existants (télévision, ordinateur, tablette,

smartphone), qui requièrent de transmettre le même contenu vidéo avec différents niveaux de qualité visuelle, de définition spatiale, de fréquence d'images, ou de profondeur de bits, des schémas de compression dits scalables sont utilisés. Le but de ces schémas de compression est d'optimiser les performances de compression en réduisant les redondances entre les différentes versions du contenu vidéo. Ces différentes versions ne sont donc pas encodées séparément : une première version du contenu est d'abord encodée et considérée comme couche de base. Une autre version, de meilleure qualité visuelle, et/ou définition spatio-temporelle, et/ou profondeur de bit, est alors encodée comme couche d'amélioration, en s'appuyant sur la couche de base décodée. Plusieurs couches d'améliorations peuvent ainsi être encodées de manière récursive afin de prendre en compte les différentes versions du contenu vidéo.

Contributions et organisation du manuscrit

Le premier chapitre présente l'état de l'art en compression vidéo, notamment les standards de compression vidéo MPEG. Les chapitres suivants contiennent les contributions de la thèse, avec en début de chapitre un état de l'art des méthodes spécifiques étudiées dans le chapitre.

Chapitre 1 : état de l'art en compression vidéo

Nous présentons dans ce chapitre les principes de bases utilisés en compression vidéo: la prédiction, la transformation, la quantification, et le codage entropique. Nous décrivons ensuite l'application de ces principes dans les standards de compressions vidéo.

L'étape de prédiction a pour but de réduire les redondances spatiales et temporelles d'un signal vidéo. Les deux types de prédiction correspondant sont nommés la prédiction Intra et la prédiction Inter respectivement. Le procédé de prédiction consiste à estimer un bloc de pixels à partir de pixels disponibles dans les parties déjà décodées de la vidéo. Dans le cas de la prédiction Intra, le bloc de pixel est prédit à partir des pixels spatialement voisins, tandis que pour la prédiction Inter, il est prédit à partir des images précédemment décodées de la séquence. Dans les schémas de compression récents tels que H.264 [1] ou HEVC [2], la prédiction Intra consiste à propager les valeurs des pixels sur la gauche et au-dessus du bloc courant suivant une direction prédéfinie. Dans H.264, 8 modes directionnels sont disponible, 33 dans HEVC. Un mode DC est également défini, qui consiste à prendre la moyenne des pixels voisins comme prédicteur du bloc courant. La prédiction Inter s'appuie sur l'estimation/compensation de mouvement, réalisée en général avec un algorithme de type *block matching* (BM). A la suite de la prédiction, on obtient un résidu de prédiction, qui correspond à la différence entre le bloc original et le bloc prédit. Ce résidu est utilisé dans l'étape suivante de transformation.

L'étape de transformation permet de réduire les corrélations d'un signal en distribuant l'énergie de ce signal sur un nombre réduit de coefficients, qui correspondent à des projections sur des fonctions de base orthogonales. La transformation peut être appliquée directement sur les pixels d'un bloc, comme dans JPEG [3], ou sur le résidu de prédiction, comme dans H.264 et HEVC. Les principales transformées utilisées en

compression d'image ou de vidéo incluent la transformée en cosinus discrète (DCT) dans les standards JPEG, H.264 et HEVC, la transformée en sinus discrète (DST) dans HEVC, par ailleurs les transformées en ondelette discrète (DWT) sont appliquées dans SPIHT [4] et JPEG2000 [5].

L'étape de quantification consiste à réduire la précision du signal, en convertissant l'ensemble de taille n des valeurs du signal d'entrée en un nouvel ensemble fini de valeurs, de taille $m < n$. Plusieurs types de quantification existent, telles que la quantification uniforme ou non uniforme, la quantification scalaire, ou la quantification vectorielle. Dans un schéma de compression, la quantification est appliquée sur les coefficients de la transformée obtenus à l'étape précédente. La quantification est irréversible, on parle alors de compression avec perte. Le pas de quantification permet de réguler le compromis débit/qualité d'image. Une forte quantification permet de grandement réduire le débit, au détriment de la qualité. Au contraire, une quantification fine permet de maintenir une bonne qualité d'image, avec un débit plus important.

Enfin, l'étape de codage entropique exploite les statistiques du signal à coder pour en réduire les redondances. Un codeur entropique utilise un code à longueurs variables (VLC) pour construire des mots de code de différentes tailles, de telle sorte que les mots de code les plus courts sont assignés aux symboles les plus fréquents, tandis que les mots de code longs sont attribués aux symboles les moins fréquents. Parmi les codeurs entropiques les plus connus, on peut citer le codage de Huffman et le codage arithmétique. Dans un schéma de compression, le codage entropique est appliqué aux coefficients de la transformée quantifiés. Les schémas de compression récents tels que H.264 et HEVC utilisent le codeur CABAC (pour *Context Adaptive Binary Arithmetic Coding*) pour encoder à la fois les coefficient de la transformée quantifiés et des éléments de syntaxe.

Les standards de compression s'appuient sur un découpage des images de la séquence vidéo en blocs de pixels, et les principes décrits ci-dessus s'appliquent au niveau de ces blocs. Dans le standard H.264, les images sont d'abord découpées en *macrobloc*, de taille 16×16, qui peuvent être sous-divisés en blocs de tailles inférieures. Dans le standard HEVC, un principe similaire mais optimisé est mis en place, où les images sont d'abord découpées en CTU (*coding tree unit*), qui peuvent atteindre une taille de 64×64. La CTU peut être récursivement divisée en CUs (*coding unit*) de tailles inférieures, dont le partitionnement forme un arbre quaternaire.

Notons enfin que la sélection d'un mode de codage (Inter ou Intra, choix du meilleur mode directionnel Intra) s'effectue également au niveau des blocs, habituellement sur la base d'un critère d'optimisation débit/distorsion, noté RDO (pour *Rate Distortion Optimization*).

Chapitre 2 : prédiction inter images basée LLE

Le chapitre commence par une description des méthodes de prédiction basées *template matching* (TM). Ces méthodes peuvent être utilisées comme alternative au BM pour l'estimation/compensation de mouvement. Le *template*, ou gabarit, est défini comme l'ensemble des pixels voisins reconstruits à gauche et au-dessus du bloc courant à prédire.

On peut alors chercher un gabarit similaire parmi les zones de la vidéo déjà décodées. Le bloc associé à ce plus proche gabarit est alors utilisé comme prédicteur du bloc courant. Cette méthode peut être avantageusement reproduite au décodeur, ce qui signifie qu'aucune information de mouvement n'a besoin d'être transmise entre l'encodeur et le décodeur. La technique TM a ainsi pu être utilisée efficacement pour la prédiction Inter [6] comme Intra [7]. Les performances de la méthode TM ont également été améliorées en cherchant plusieurs gabarits similaires au gabarit courant, et en combinant les blocs adjacents pour obtenir le prédicteur. Initialement, un simple moyennage était utilisé [8][9], mais il a été montré que l'utilisation d'un moyennage pondéré, par exemple basé sur des représentations parcimonieuses [10][11], améliorent d'autant plus les performances.

Dans ce chapitre, nous proposons d'utiliser un moyennage pondéré dont les poids sont calculés en utilisant une approximation LLE (pour *Locally Linear Embedding*). L'efficacité de la technique LLE a été démontrée pour la prédiction Intra [12][13], et nous l'étendons ici à la prédiction Inter. L'idée principale consiste dans un premier temps à obtenir K gabarits similaire au gabarit courant. La LLE permet alors d'obtenir une combinaison linéaire des K gabarits qui approxime au sens des moindres carrés le gabarit courant, avec pour contrainte que la somme des poids soit égale à 1. La combinaison linéaire obtenue peut alors être appliquée au K blocs adjacents aux K gabarits pour obtenir le prédicteur du bloc courant.

Nous proposons plusieurs stratégies pour obtenir les K gabarits similaires au gabarit courant. La plus directe consiste à chercher les K gabarits les plus proches du gabarit courant au sens de la distance Euclidienne dans une fenêtre de recherche. Cette méthode est notée TM-LLE. L'hypothèse sous-jacente est que les blocs et gabarits sont fortement corrélés, ce qui n'est pas toujours vérifié en pratique. Toutefois cette méthode à l'avantage, comme le TM, d'être reproductible au décodeur, et ne nécessite pas de transmettre d'information sur le mouvement. Une variante de TM-LLE est présentée, notée ITM-LLE, dont le but est de renforcer les corrélations entre blocs et gabarits, sans avoir à communiquer d'information de mouvement. On cherche d'abord *le* plus proche gabarit du gabarit courant. Le patch composé de ce gabarit et de son bloc adjacent est alors utilisé pour trouver les $K-1$ patchs les plus proches dans la fenêtre de recherche. Ces $K-1$ patchs contiennent alors les $K-1$ gabarits restants nécessaires pour calculer l'approximation LLE.

Dans un second temps, nous proposons une stratégie, notée BM-LLE, qui cherche également à renforcer les corrélations entre blocs et gabarits, mais cette fois en s'appuyant sur une information de mouvement, qui est transmise au décodeur. La première étape consiste à effectuer un algorithme de type BM. Dans un second temps, le patch composé du bloc obtenu par BM et de son gabarit adjacent est alors utilisé pour trouver les $K-1$ patchs les plus proches dans la fenêtre de recherche.

Enfin, une dernière stratégie est décrite, qui est une optimisation de la méthode ITM-LLE, et est notée oITM-LLE. La méthode consiste à obtenir L prédicteurs en répétant L fois la méthode ITM-LLE. La meilleure itération en un sens débit/distorsion est retenue, et l'index de cette itération est transmis au décodeur.

Les différentes méthodes ont été implémentées dans H.264, mais restent valides dans

HEVC. Les résultats des différentes expérimentations montrent que la méthode oITM-LLE permet d'obtenir les meilleures performances de codage, au prix d'une complexité importante. Les méthodes TM-LLE et ITM-LLE permettent un meilleur compromis entre performances de codage et complexité. La complexité de BM-LLE est raisonnable mais ses performances débit/distorsion ne sont pas meilleures que l'état de l'art. Malgré l'importante complexité de oITM-LLE, son application à la prédiction Intra, combinée à une méthode moins complexe comme TM-LLE appliquée à la prédiction Inter permet d'obtenir les meilleurs performances de codage pour une complexité convenable.

Nous sommes également convaincus que la complexité des méthodes pourrait être réduite, en utilisant par exemple des méthodes de recherche des plus proches voisins rapides [14][15][16][17], une estimation rapide des poids [18], ou encore en parallélisant l'implémentation, par exemple sur GPU [19].

Chapitre 3 : techniques de réduction du bruit de quantification basées sur des épitomes

Le chapitre commence par une description des méthodes existantes pour la génération d'épitome. Un épitome est défini comme une représentation condensée d'une image ou vidéo, contenant l'essence de ses propriétés de texture. Plusieurs types de représentations épitomiques existent [20][21], toutefois nous basons ce travail sur un type d'épitome adapté au contexte de la compression [22][23]. Pour créer ce type d'épitome, la première étape consiste à rechercher les auto-similarités de l'image, ce qui permet d'obtenir pour chaque bloc de l'image une liste d'appariements contenant des blocs similaires, c'est à dire dont la distance Euclidienne est inférieure à un certain seuil prédéfini. Dans un second temps, on peut générer de manière itérative des épitomes charts, qui contiennent les éléments de texture les plus représentatifs, et permettent de reconstruire l'image complète. L'ensemble des épitomes charts constituent l'épitome.

Nous présentons ensuite les méthodes de l'état de l'art en dé-bruitage, principalement conçues pour le cas du bruit blanc Gaussien additif. Dans un premier temps est décrit le filtre optimal de Wiener, qui opère dans le domaine transformée 2D (Fourier ou ondelettes par exemple). En admettant que le spectre du signal source est connu, le filtre de Wiener donne les coefficients de réduction qui permettent de minimiser l'erreur quadratique moyenne du signal débruité. Ce filtre optimal n'est pas utilisable en pratique, on remplace donc souvent le spectre du signal source par le spectre du signal débruité par une autre méthode [24], qualifié ici d'oracle.

La deuxième méthode présentée est le seuillage dans le domaine transformée 2D [25][26], connue sous le nom de *hard thresholding*. La technique consiste à annuler dans le domaine transformée les coefficients inférieurs à un certain seuil défini en avance. Le but est d'annuler les coefficients qui ont un faible rapport signal à bruit. Cette méthode peut par exemple être utilisée pour produire un signal débruité qui servira d'oracle pour le filtrage de Wiener.

Nous décrivons alors des méthodes de l'état de l'art particulièrement efficaces, qui s'appuient sur les redondances naturelles des images ou vidéos au travers de techniques multi-patchs. L'algorithme NLM (pour *Non Local Mean*) [27][28] commence par

chercher les K plus proches voisins d'un patch courant à débruiter dans une fenêtre de recherche centrée sur ce patch. Les K plus proches voisins sont alors combinés linéairement pour obtenir une version débruitée du patch courant. Les poids de la combinaison linéaire sont du type $w_i = e^{-\frac{d_i^2}{2\sigma_{NLM}^2}}$, où d_i est la distance entre le patch courant et son i-ème voisin, et σ_{NLM} est un paramètre qui règle le degré de filtrage.

Enfin, l'algorithme BM3D est présenté [29], qui est une des méthodes les plus performantes de l'état de l'art. Dans une première étape, pour chaque patch de l'image à débruiter, ses K plus proches voisins sont trouvés et empilés en un groupe 3D. Le groupe 3D est alors traité en utilisant le seuillage dans le domaine transformée 3D. L'image débruitée obtenue est alors utilisée dans une seconde étape, où les K plus proches voisins de chaque patch sont à nouveau trouvés, ce qui permet de construire de nouveaux groupes 3D. Les groupes 3D sont alors utilisés comme oracle du filtre de Wiener pour traiter les groupes 3D correspondant extraits de l'image bruitée.

La première contribution de ce chapitre est une méthode efficace de génération d'épitome, qui optimise la méthode décrite plus haut, présentée dans [23]. Cette méthode s'appuie sur des listes d'appariements intermédiaires ou du partitionnement *ad hoc* des blocs de l'image afin d'accélérer et d'alléger l'étape de recherche des auto-similarités, qui était effectuée jusqu'à présent avec une recherche exacte de tous les appariements.

Nous proposons ensuite un schéma d'amélioration des codecs modernes, qui se situe en dehors de la boucle de codage. Ce schéma consiste à générer coté encodeur des épitomes de la séquence source, afin de les transmettre avec une bonne qualité au décodeur. Ils sont alors utilisés pour débruiter la séquence complète encodée avec une qualité inférieure. Plusieurs cas d'application sont possibles pour ce schéma. Le premier consiste à améliorer un schéma de compression mono-couche. Un seul épitome est alors transmis pour un groupe d'images, afin de limiter le surcoût en débit. La seconde application s'intègre dans un schéma scalable, où un épitome est transmis pour chaque image de la séquence, l'ensemble des épitomes étant considéré comme une couche d'amélioration. Pour chaque image, les patchs contenus dans l'épitome sont alors utilisés pour débruiter les patchs de l'image en dehors de l'épitome. Dans le cas où la couche de base est sous-échantillonnée, les méthodes proposées réalisent conjointement la super-résolution et le débruitage des patchs en dehors des épitomes.

Nous décrivons plusieurs méthodes de débruitage basées épitome, qui commencent toutes par rechercher K patchs proches du patch à débruiter parmi les patchs bruités colocalisés avec des épitomes charts. La première technique est inspirée de l'algorithme NLM et des méthodes LLE présentées dans le chapitre précédent. Elle consiste à calculer des poids pour chacun des K patchs, soit de manière similaire à NLM, soit en utilisant une approximation LLE. Les K patchs de bonne qualité correspondants situés dans les épitomes charts sont alors combinés en utilisant ces poids afin d'obtenir le patch débruité.

La seconde méthode est une adaptation du BM3D. Un premier groupe 3D est constitué dans lequel sont empilés les K patchs bruités, ainsi qu'un second groupe qui contient les K patchs de bonne qualité correspondants. En s'appuyant sur ce deuxième groupe

3D, on peut déterminer un seuil optimal pour le seuillage dans le domaine transformée du premier groupe 3D bruité. De la même manière, des coefficients optimaux du filtre de Wiener peuvent être obtenus.

Enfin, un troisième procédé est décrit, qui calcule par régression une projection linéaire entre les K patchs bruités et leurs correspondants de bonne qualité situés dans les épitomes charts. La projection est ensuite appliquée au patch courant pour obtenir une version débruitée.

Les résultats montrent qu'il est difficile d'améliorer un codec mono-couche, ici HEVC, en utilisant le schéma proposé, car les coûts en débit des épitomes s'avèrent trop importants comparés aux gains de débruitage. En revanche, les performances de codage dans le cadre d'un schéma scalable sont meilleures que SHVC, l'extension scalable de HEVC, dans les deux cas testés. Dans le premier cas la couche de base est de la même résolution spatiale que la couche d'amélioration mais de qualité plus faible. Dans le second cas la couche de base est sous-échantillonnée avec un facteur 2x2 par rapport à la couche d'amélioration.

Chapitre 4 : techniques de réduction du bruit de quantification basées sur des clusters

Le chapitre commence par une description des méthodes existantes pour le partitionnement de données (*clustering*), ce qui est dans ce chapitre le pendant de l'épitome dans le chapitre précédent. Le schéma décrit ci-dessous s'appuie principalement sur la méthode des K-moyennes [30]. En considérant que le nombre de partitions K est connu, l'algorithme des K-moyennes assigne un point de l'espace à partitionner à la partition qui a la moyenne la plus proche. Une procédure itérative est décrite pour résoudre ce problème. Une revue non exhaustive de l'état de l'art en classification est ensuite faite. Nous présentons notamment les machines à vecteurs supports (SVM) [31][32] et les arbres de décisions [33][34]. Nous nous attachons principalement à décrire les modèles des classifieurs, sans rentrer dans le détail des algorithmes utilisés pour obtenir ces modèles.

Nous proposons dans ce chapitre un schéma d'amélioration en dehors de la boucle de codage comparable à celui proposé dans le chapitre précédent. Toutefois, nous adoptons ici un point de vue différent, car nous effectuons coté encodeur un partitionnement des blocs de l'image, et apprenons toujours coté encodeur une projection linéaire entre les patchs décodés et les patchs sources de chaque partition. C'est le résultat de cet apprentissage, les matrices de projections, qui est alors transmis au décodeur. Coté décodeur, on effectuera le même partitionnement qu'à l'encodeur, et on pourra alors appliquer les projections linéaires aux patchs des partitions correspondantes pour les débruiter. Nous espérons ainsi diminuer le surcoût en débit par rapport à la transmission des épitomes, tout en ayant de bonnes performances de débruitage, car l'apprentissage est réalisé par rapport au signal source.

Le partitionnement est effectué sur les blocs de la vidéo codée/décodée avec la méthode des K-moyennes, de manière à produire exactement les mêmes partitions à l'encodeur comme au décodeur. Le choix du nombre de partitions K devient essentiel, car il permet de réguler l'équilibre débit/distorsion du schéma d'amélioration. Nous

proposons donc une procédure basée sur un partitionnement en arbre binaire avec un critère de décision RDO pour trouver de manière adaptative la valeur optimale de K. La structure de l'arbre binaire doit alors être transmise au décodeur, pour un surcoût en débit négligeable.

Les projections linéaires apprises pour ces partitions sont obtenues sous forme de matrices, qu'il faut alors transmettre au décodeur. Nous décrivons alors une méthode pour encoder ces matrices, en les considérant comme les images d'une séquence vidéo. Nous pouvons alors les compresser avec un codec tel que HEVC, en étant attentif à ne pas trop réduire la profondeur de bits des matrices, qui contiennent originellement des valeurs flottantes, afin de ne pas dégrader les performances de débruitage.

Nous proposons également une variante en deux étapes, où un premier algorithme de débruitage "aveugle" (dans le sens ou il ne requiert pas de transmission d'information au décodeur) est appliqué sur la vidéo, avant d'appliquer le schéma décrit ci-dessus.

Les expérimentations montrent que le schéma proposé permet d'améliorer les performances de HEVC, notamment dans sa variante en deux étapes.

Nous présentons ensuite une procédure de partitionnement optimale pour notre application de débruitage. Les résultats obtenus permettent de définir une borne supérieur pour les performances du schéma proposé, qui montrent que des progrès peuvent encore être accomplis. Toutefois, cette méthode de partitionnement ne peut être utilisée directement dans le schéma d'amélioration. En effet le partitionnement obtenu n'est pas causale, c'est à dire qu'il ne peut être reproduit indépendamment au décodeur. Si on envisage alors de transmettre directement pour chaque patch un index indiquant la partition à laquelle il appartient, les performances de codage sont en dessous des performances de HEVC.

Le schéma d'amélioration proposé est ensuite intégré dans un schéma de compression scalable. La méthode d'apprentissage par partition est appliquée pour super-résoudre la couche de base, qui peut ensuite être utilisée pour la prédiction de la couche d'amélioration. Les résultats obtenus avec la méthode des K-moyennes dans SHVC montrent que si la super-résolution de la couche de base est bien plus efficace que le filtre de sur-échantillonnage de SHVC, les gains obtenus ne sont pas reportés sur la couche d'amélioration. L'utilisation de la procédure de partitionnement optimal permet toutefois de montrer que des progrès sont possibles, et que des gains suffisant sur la couche de base peuvent se répercuter sur la couche d'amélioration.

Enfin nous terminons ce chapitre en ouvrant sur un schéma basé classification, qui remplace alors le partitionnement. Le but est d'obtenir un compromis entre les performances du partitionnement des K-moyennes et optimal. Dans ce schéma, le partitionnement optimal est calculé coté encodeur. Un classifieur est alors appris, qui permet d'estimer la valeur de la partition optimale d'un patch à partir de la valeur de ses pixels décodées, ou d'autres caractéristiques associées au patch, telles que son mode de codage ou son débit associé. Le classifieur peut alors être transmis au décodeur, afin de retrouver, au moins partiellement, le partitionnement optimal. Une autre alternative est de transmettre au décodeur une partie des indexes du partitionnement optimal, et d'entraîner le classifieur coté décodeur où il peut être utilisé pour estimer le partitionnement des patchs pour lesquels les indexes optimaux sont inconnus.

Des expérimentations ont étés réalisées avec un schéma de codage simplifié basé sur le codec JPEG. Le schéma basé classification n'apparaît pas efficace appliqué dans un schéma mono-couche, il permet en revanche d'obtenir de meilleures performances que la méthode des K-moyennes dans un contexte scalable avec super-résolution.

Conclusion

Nous avons exploré dans cette thèse des méthodes originales ayant pour but d'améliorer les techniques existantes de compression vidéo. Un des principes de base de la compression vidéo est de réduire les redondances spatiales et temporelles du signal vidéo. Les standards de compression exploitent ce principe à travers les outils de prédictions Intra et Inter respectivement, qui sont décrits dans le Chapitre 1. La prédiction Inter s'appuie sur l'estimation/compensation de mouvement, tandis que la prédiction Intra propage l'information des pixels voisins reconstruits. Le but des travaux proposés dans ce manuscrit est d'améliorer les schémas de compression existants en exploitant d'avantage les auto-similarités intrinsèques au signal, notamment en s'appuyant sur des méthodes multi-patchs.

Dans un premier temps, nous avons cherché à améliorer les outils de prédiction des standards, en utilisant des méthodes multi-patchs basées LLE. Ces méthodes sont présentées dans le Chapitre 2, et nous prouvons leur efficacité en les implémentant dans H.264. Le principe reste toutefois valide dans HEVC. Nous montrons notamment que la combinaison des méthodes proposées pour la prédiction Inter et Intra donne les meilleures performances.

Nous nous sommes ensuite intéressés dans le Chapitre 3 à des méthodes de débruitage basées sur des épitomes, pour améliorer la qualité des vidéos décodées en post-traitement. Un épitome est une version condensée d'une image ou vidéo, contenant l'essence de ses propriétés de texture, que nous transmettons au décodeur en plus de la séquence complète, mais avec une qualité supérieure. Nous pouvons alors utiliser coté décodeur des méthodes de débruitage multi-patchs s'appuyant sur les patchs de bonne qualité contenus dans les épitomes. Nous montrons que les techniques proposées sont efficaces en comparaison du schéma scalable SHVC.

Enfin nous proposons dans le Chapitre 4 un autre schéma d'amélioration s'appuyant sur des méthodes de débruitage basée sur le partitionnement des patchs de la séquence. Ce partitionnement est effectué symétriquement à l'encodeur et au décodeur. Pour chaque partition nous apprenons coté encodeur une projection linéaire entre les patchs sources et les patchs codés/décodés. Ces projections sont transmises au décodeur, où elles peuvent être appliquées pour débruiter les patchs décodés. Une méthode de partitionnement optimale est également proposée, qui permet de définir une limite théorique supérieure pour les performances, et montre que des améliorations sont encore possibles. Le schéma d'amélioration est également intégré dans un schéma scalable avec super-résolution, et nous montrons une potentielle amélioration par rapport à SHVC en utilisant le partitionnement idéale. Enfin nous terminons en proposant un schéma original basé sur des méthodes de classification qui permet d'exploiter au moins de manière

partielle le partitionnement optimal.

Ainsi, dans cette thèse, nous nous sommes attachés à exploiter les redondances naturelles d'un signal vidéo pour améliorer les performances de compression. Nous nous sommes basés pour cela sur des méthodes multi-patchs, intégrées dans un premier temps dans la boucle de codage. Nous avons ensuite proposé des schémas d'amélioration en dehors de la boucle de codage, en se basant sur des techniques exploitant les auto-similarités d'une séquence, tels que les épitomes, le partitionnement de données, le débruitage, ou la super-résolution. Nous montrons l'efficacité de ces méthodes originales pour diverses applications, soit des schémas de compression simple couche, soit des schémas scalables. De plus, nous pensons que nos méthodes peuvent être étendues à des applications supplémentaires, tels que la compression de vidéos à dynamique étendue, ou la compression dans le *cloud*.

Introduction

Context

We are nowadays more and more greedy for video content, and at the same time more and more demanding of a high quality of experience, both in terms of speed and visual quality. Video content providers such as Youtube or Netflix are emblematic of this growing trend. Despite Moore's law and the subsequent increasing processing power, the increasing stocking capacities and modern efficient network infrastructures, transmitting over a network or storing video content at the desired speed and quality still requires compression. To further improve the user immersion, digital videos can now be produced with high resolution (4K - 8K), wide color gamut, high dynamic range, or high frame rate. These emerging formats represent a formidable volume of data, and call for very efficient compression. In a white paper published in 2014, Cisco predicted that by 2018, the Internet video traffic will be 79% of all consumer Internet traffic. Furthermore, 80% to 90% of the global consumer traffic is expected to be videos of all forms (TV, Internet, VOD, VoIP, P2P). We can see that efficient compression services are critical, and will still be in a close future. From this perspective, the MPEG standard HEVC is currently one of the most efficient video compression scheme, and is expected to gradually replace its predecessor H.264.

The main principles of modern video compression standards reside in the reduction of spatial and temporal redundancies, through prediction tools, the use of a transform to further reduce the inner correlations of the signal, followed by quantization to remove non-perceptive information, and entropy coding to remove the remaining statistical redundancies. In lossless compression, which aims at perfectly reconstructing the input signal, the quantization step is not performed. This type of compression is especially useful in medical imaging, where the image quality is crucial for efficient diagnosis. The work presented in this thesis deals with the broader scope of lossy compression, which is used in particular in consummer applications, such as the ones cited above. In this case, the quantization step act as a regulation parameter to balance the bit-rate and the visual quality.

To face the multiplicity of possible receivers (TV, computer, mobile devices), which requires transmitting the same video content with different visual qualities, spatial resolutions, frame rates, or bit-depths, scalable video compression schemes are used. To optimize the compression performances, the redundancies occurring across the different versions of the content should be taken into account. Thus, instead of encoding them

separately, one version is first encoded as a base layer. Another version, usually of better quality, and/or higher spatial/temporal resolution, and/or bit-depth, is then encoded as an enhancement layer, using the base layer as prediction. Several enhancement layer can then be recursively encoded, to account for all the different versions of the video.

Motivations

The recent MPEG standard HEVC [2] is undeniably one of the most efficient modern codecs, and achieves a bit-rate reduction of about 50% [35] compared to its predecessor H.264 [1]. However, the development and standardization process from H.264 to HEVC took about 10 years. Such pace is too slow to keep up with the projections of video traffic mentioned above.

In addition to the MPEG standards, many actors are now developing new codecs to address this issue. For example, Google's VP9 [36] is already competitive with HEVC, and rely on software optimization to speed-up its development process. Next-generation royalty-free codecs are also announced, e.g. Google's VP10, Cisco's Thor project [37], or Daala [38] (by the Mozilla Foundation and Xiph.org foundation). Despite this renewed interest in video compression, these codecs architectures are still close to the one of HEVC. HEVC itself can be seen as an optimization of H.264, as it primarily relies on the same tools.

As a matter of fact, the main principles of video compression presented above rely on concepts designed more than 30 years ago. The temporal prediction tool, which aims at reducing temporal redundancies, was first presented in 1972 in the form of frame differencing [39]. The motion compensation prediction used nowadays was presented in 1981 [40]. The discrete cosinus transform used in many modern codecs was designed in 1974 [41], and the first entropy coders date back to 1952 for the variable length coding [42] and 1972 for the arithmetic coding [43].

These observations call for further improvement of the existing tools mentioned above, or proposals for disruptive techniques. For example, the PROVISION project [44] proposed a change of paradigm in the form of perceptually optimized video compression, based on tools such as texture analysis/synthesis.

Contributions and structure of the thesis

We investigate in this thesis novel methods, which further exploit the natural redundancies occurring in video contents through multi-patches techniques. Note that we focus in this work on improving the compression performances, and thus allow significant complexity increase. Because of the aforementioned increasing processing capabilities, we expect the proposed methods to be acceptable in the future, and the core algorithms could already benefit from parallel implementations, e.g. on GPU.

In Chapter 1, we introduce the main principles used in video compression, and their implementation in the MPEG standards H.264 and HEVC, as well as the scalable extension of HEVC, SHVC.

In Chapter 2, we first present state-of-the-art template matching methods for prediction. The template consists in a set of known neighbor pixels on top and left of the current block of unknown pixels to be predicted. Template matching can then be used to perform motion compensation/estimation, similarly to the classical block matching algorithm. We then propose extensions of the template matching methods to multi-patches methods based on Local Linear Embedding (LLE) to improve temporal prediction. Instead of considering a single block predictor as in the motion estimation/compensation, we search for multiple similar patches, called neighbors, of the current patch (which comprises the block and its template). The known pixels of the template can then be estimated as a linear combination of its neighbors, which weights are computed with the LLE. The same weights can then be applied on the respective adjacent blocks to a obtain the block predictor. Several neighbor search strategies are proposed, and integrated in H.264. However, the concepts still hold in HEVC.

In Chapter 3, state-of-the-art techniques on epitome generation and de-noising are first described. The epitome is defined as a factorized representation of an image or video, containing the essence of its textural properties. Although it has emerged independently from standard compression methods, it is nevertheless closely related to the concept of compression, as it aims at reducing the self-similarities of an image or video. In addition, we observed that multi-patches methods were also proven effective for de-noising applications, notably with the Non Local Mean (NLM) and BM3D methods. Thus, we propose epitome-based de-noising methods, where we assume that the epitomes are available at a higher quality than the image or video to be de-noised. When de-noising a patch, we can then search for multiple similar patches in the epitome, and use the higher quality patches to perform efficient de-noising. More specifically, three epitome-based de-noising methods are proposed, first by combining the epitome patches using NLM or LLE weights, second by adapting the BM3D, and finally by computing a linear mapping between the noisy patches and their corresponding high quality epitome patches. The epitome-based de-noising methods are integrated in a proposed out-of-the-loop framework, where the epitomes are sent to the decoder along with the encoded video, but with a higher quality. The proposed de-noising methods can then be applied at the decoder side to reduce the quantization artifacts. The proposed framework is applied to a single layer scheme as well as scalable schemes.

In Chapter 4, we begin by presenting state-of-the-art clustering methods, which is in this chapter the counterpart of the epitome in the previous chapter, and classification methods. In fact we propose in this chapter an out-of-the-loop framework, with clustering-based de-noising, which is integrated in a compression scheme in order to remove the quantization noise at the decoder side. The method first clusters the coded/decoded patches at the encoder side. Then, linear mappings are learned for each cluster between the coded/decoded patches and the corresponding source patches. The linear mappings are then sent to the decoder, where the same clustering as the encoder side is performed. Finally, the linear mappings are applied to the cluster patches in order to reduce the quantization noise. Since, only the linear mappings are transmitted to the decoder, the rate of the side information is reduced compared to the previous chapter. The proposed framework is also adapted for a scalable scheme. In addition,

we propose an optimal clustering algorithm in order to improve the de-noising performances for a same number of cluster. However, this method only set an upper bound on the performances, as it can not be reproduced at the decoder side. We end this chapter by proposing a novel approach based on classification, where classifiers are learned in order to predict the optimal cluster labels. The classifier can for instance be trained at the encoder side and its model sent to the decoder, where it can be used to recover the optimal clusters. Alternatively, a subset of the true optimal clusters labels can be transmitted to the decoder, where the classifier can then be trained. We thus expect to approach the upper bound set by the optimal clustering.

For Chapters 3 and 4, the experiments are performed with HEVC and SHVC, however the out-of-the-loop frameworks can be advantageously adapted to any codec cited above.

Chapter 1

State-of-the-art in video compression

In this chapter, we first present the main principles used in video compression. We then described more in details how these principles are introduced in the video compression standards H.264/AVC and HEVC. Finally, we expose the motivations for the work presented in this thesis.

1.1 Main principles in video compression

A video compression scheme converts an input digital signal into a bit-stream lighter than the source signal. The goal of compression schemes is to remove the spatial and temporal redundancies of the video to reduce the amount of data needed to represent the signal. The main operations to achieve this goal are shown in Fig. 1.1 and described in the next sections.

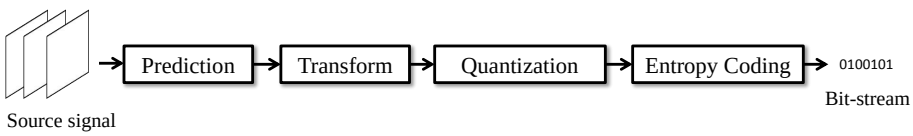


Figure 1.1: Main operations in a compression scheme.

1.1.1 Prediction

The prediction step aims at removing spatial and/or temporal redundancies. The two types of corresponding prediction are named Intra prediction and Inter prediction respectively. The prediction consists in estimating a block of pixels from previously decoded pixels. In the case of Intra prediction, the block of pixels is predicted from spatially neighboring pixels, while in Inter prediction, it is predicted from previously

decoded frames of the video. The prediction process of recent video compression standards will be described in details in the next sections. Once the prediction is obtained, the prediction error, also called prediction residue, is passed on to the transform step.

1.1.2 Transform

The transform reduces the correlations in a signal and compacts the energy in a limited number of coefficients. In a compression scheme, it can be applied directly on pixel blocks, such as in JPEG [3] or JPEG2000 [5], or on the prediction residue, such as in H.264/AVC [1] or HEVC [2]. The main transformed used in image or video compression schemes include the Discrete Cosinus Transform (DCT) in JPEG, H.264/AVC, HEVC, the Discrete Sinus Transform (DST) in HEVC, and Discrete Wavelet Transforms (DWT) in SPIHT [4] and JPEG2000.

1.1.3 Quantization

The quantization consists in reducing the signal precision, by mapping the set of input values of size n to a new finite countable set of size m inferior to n. There are several types of quantization, such as uniform and non-uniform, scalar quantization and vector quantization.

In a compression scheme, the quantization is applied on the transform coefficients obtained at the previous step. Compression schemes usually apply scalar quantization, in which a quantization step q is defined and evaluates the level of precision of the quantized signal. Formally, if $\{y_0, \ldots, y_{m-1}\}$ represents the set of target values, a signal x is quantized by a quantizer Q as follow:

$$Q(x) = y_i \text{ if } x \in [y_i - \frac{q}{2}, y_i + \frac{q}{2}] \tag{1.1}$$

The quantization is not a reversible step, and thus produces lossy compression. In other words, this step is not performed in lossless compression schemes. The quantization step, also called quantization parameter (QP), is used to regulate the bit-rate/quality balance. A strong quantization will result in low bit-rates, but also a low quality. On the contrary, little quantization will allow to maintain a high quality, at the cost of a high bit-rate.

1.1.4 Entropy coding

Entropy coding takes advantage of the statistics of a signal to reduce the statistical redundancy. An entropy coder uses the Variable Length Code (VLC) for code-words, such that short binary codes represent the values with the most occurrences, while values with the lowest occurrences are represented with long binary codes. Huffman coding and arithmetic coding are popular entropy coding methods.

In a compression scheme, the entropy coding is applied on the quantized transform coefficients. Recent compression schemes such as H.264/AVC and HEVC use the Con-

text Adaptive Binary Arithmetic Coding (CABAC) to encode the quantized transform coefficients, as well as syntax elements.

1.2 Standards

We describe in this section the video compression standards H.264/AVC and HEVC. In addition to the overall standard structures, we focus on the main features which are necessary to understand the work of this thesis, such as the prediction process. Other features such as transform, quantization, and additional tools like deblocking filters, are not described.

1.2.1 H.264/Advanced Video Coding (AVC)

The H.264/AVC standard, also known as MPEG-4 Part 10, has been introduced in 2003 as a joint project of the International Standards Organization (ISO) and the International Telecommunications Union (ITU). An overview of the standard is given in [1].

1.2.1.1 Hierarchical syntax

A video sequence in H.264/AVC is described by the following hierarchical levels, illustrated in Fig. 1.2:

- The **sequence**, which comprises global parameters, such as the number of frames, the frame rate, the spatial resolution of the frames, the color format, the bit-depth. A sequence is made of several groups of pictures.
- The **group of pictures (GOP)**, which contains several successive frames, or pictures, and which structure defines the order and the types of these frames (see section 1.2.1.2). The GOP structure is repeated to form the sequence, and defines a coding period.
- The **frame**, or **picture**, which is the sequence unit on the temporal axis.
- The **slice**, which consists in a group of macroblocks, and represents either a part of a frame of the full frame.
- The **Macroblock (MB)**, which is a luminance pixels block of size 16×16, with the associated chrominance pixels blocks. The chrominance blocks size depends on the color format. For example, the chrominance blocks size is 8×8 when using the prominent 4:2:0 format.
- The **block**, which is a sub-partition of a marcoblock, usually consists in a pixel block of size 8×8 or 4×4. Decisions on the prediction modes of transform can be made at this level to adapt to the local activity of the signal.

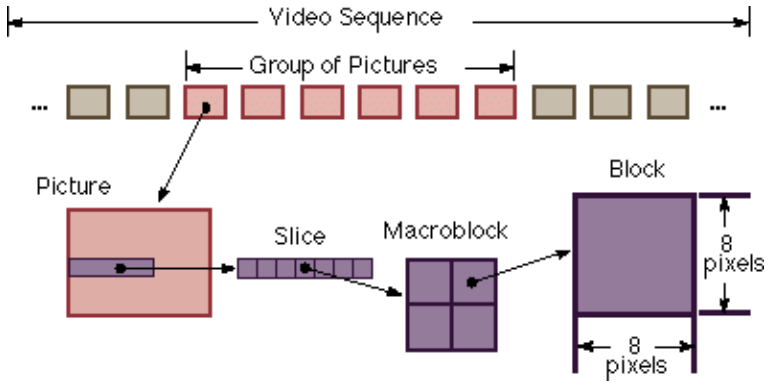


Figure 1.2: Hierarchical levels of a sequence in H.264/AVC.

1.2.1.2 Types of frame

Three types of frame can be contained in a GOP:

- The **I frame** (intra coded frame), which is coded independantly of all other pictures. All its macroblocks and blocks are coded using Intra prediction.
- The **P frame** (predictive coded frame), which can be temporally predicted from a reference frame. Its macroblocks and blocks can be coded using either Intra or Inter prediction. It can only be predicted from a past I or P frames.
- The **B frame** (bipredictive coded frame), which can be temporally predicted from two reference frames, usually one past and one future reference frame. This means that the decoding order and the display order can be different, as illustrated in Fig. 1.3.

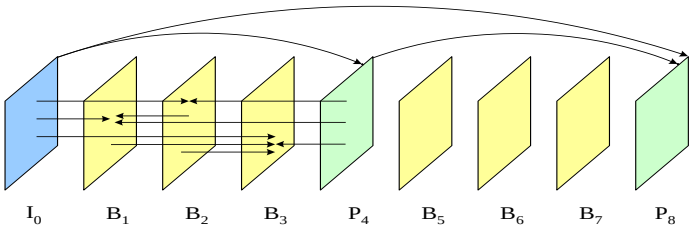


Figure 1.3: GOP structure for inter prediction in H.264/AVC. The frames are decoded in the order I_0, P_4, B_2, B_1, B_3, but displayed in the order I_0, B_1, B_2, B_3, P_4.

1.2.1.3 Encoder structure

The encoding scheme of H.264/AVC is shown in Fig. 1.4. When encoding a sequence, the frames already encoded are decoded in the loop and stored in a buffer of reference

frames to be used for Inter prediction. Inside a frame being encoded, the previously encoded macroblocks are also decoded, and then used for Intra prediction. The maroblocks in a slice or a frame are processed in a raster scan order, from top left to bottom right.

Note that in MPEG standards, only the decoder is normative. Although H.264/AVC encoders rely on the structure shown in Fig. 1.4, different encoder designs can be considered, which aim at improving the coding performances, generally under some complexity constraints.

The prediction modes are described more in details in the following sections.

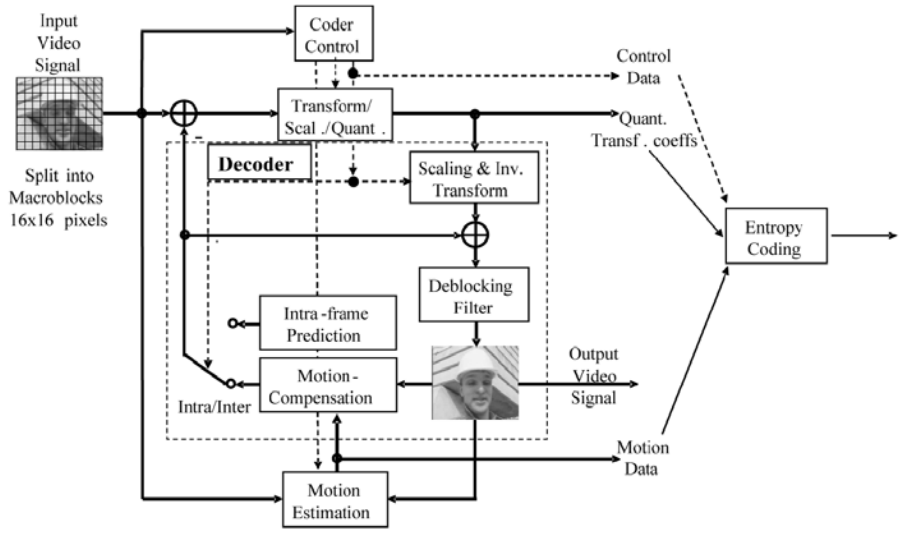


Figure 1.4: Basic coding structure for H.264/AVC for a macroblock. Source: [1].

1.2.1.4 Intra prediction

When using the Intra prediction modes, a block of pixels is predicted from its neighboring blocks previously encoded and reconstructed. Given the raster scan order, the neighboring blocks are on the left, on top, and diagonal top left of the current block. The Intra prediction assumes that the neighboring blocks are highly correlated, and propagates the information of the neighboring pixels in the current block. In H.264/AVC, the Intra prediction can be used for blocks of size 16×16, 8×8 or 4×4.

For 16×16 blocks, four modes are defined, as shown in Fig. 1.5: horizontal, vertical, DC (averaging of the neighboring pixels) and plane.

For 8×8 and 4×4 blocks, nine modes are defined, as shown in Fig. 1.6 for the 4×4 blocks: height directional modes and a DC mode.

In order to signal the best Intra mode to the decoder, only its difference with a mode predictor is sent. The mode predictor is obtained as the minimum between the Intra index modes of the blocks on top and on the left of the current block.

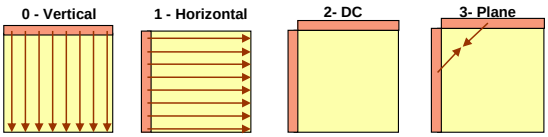


Figure 1.5: Intra prediction modes for 16×16 blocks.

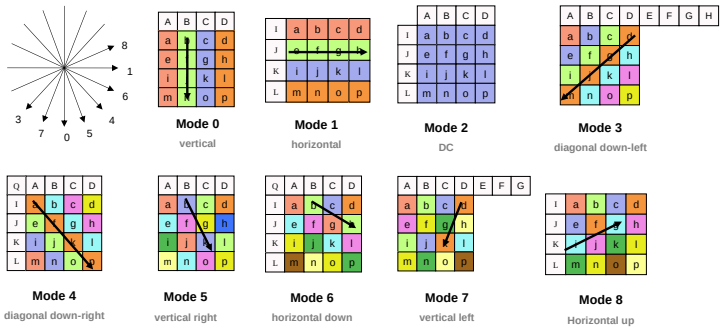


Figure 1.6: Intra prediction modes for 8×8 and 4×4 blocks (represented for 4×4 blocks).

1.2.1.5 Inter prediction

In H.264/AVC, Iner prediction consists in a block motion estimation/compensation from a previously decoded frame. The motion estimation finds the displacement of the current block with respect to a reference frame (see Fig. 1.7). Motion estimation is usually performed using a block matching (BM) algorithm, which searches in the reference frame for the block which minimizes a measure of distance with regards to the current block. The measure of distance used is usually the sum of absolute distance (SAD) or the sum of square error (SSE). The motion estimation produces a motion vector which is sent to the decoder, where the motion compensation can be performed. The motion compensation produces the prediction block by copying the pixels from the block pointed by the motion vector in the current block.

In H.264/AVC, the motion estimation can be performed at a quarter-pixel accuracy, by up-sampling the reference frames. More precisely, a 6-tap interpolation filter is used to produce half-sample positions, followed by bilinear filtering for quarter-sample positions.

To transmit the motion vector, only its difference with a motion vector predictor is sent. The motion vector predictor is obtained as the median of the motion vectors of the neighboring blocks previously decoded.

Several block sizes are defined for Inter prediction, including rectangular partitions in order to improve the motion estimation flexibility: 16×16, 16×8, 8×16 and 8×8. The 8×8 blocks can be further divided into 8×4, 4×8 and 4×4 blocks (see Fig. 1.8).

For 16×16 blocks, a Skip mode is also defined, for which no side information is sent

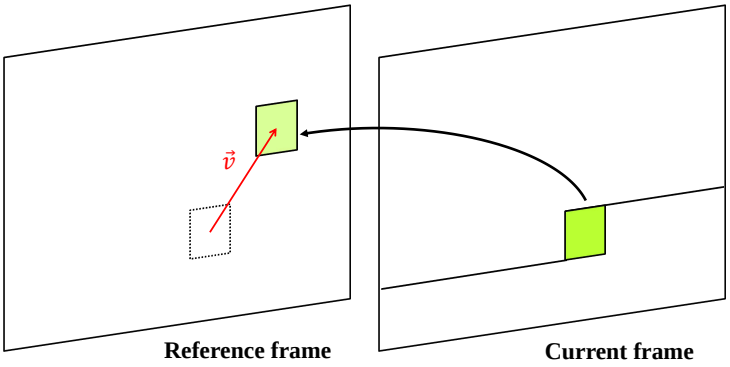


Figure 1.7: Motion estimation with block matching. The motion vector obtained $\vec{v}$ is shown in red.

to the decoded, *i.e.* no motion vector nor transform residue coefficients. In this mode, the prediction is performed by directly copying the pixels of the block pointed by the motion vector predictor.

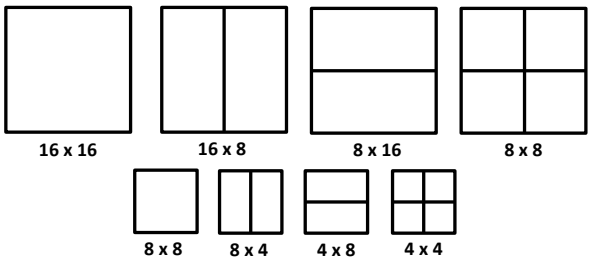


Figure 1.8: Block partitions for Inter prediction in H.264/AVC.

For P frames, the motion estimation is performed in past reference frames, while B frames can perform the motion estimation in both past and future reference frames. More formally, the past reference frames are stored in a list denoted L0, while the future reference frames are stored in a list L1. Furthermore, a block in a B frame can be predicted using two motion vectors, usually found in the L0 and L1 list respectively. The block predictor is obtained by combining the two compensated blocks. In Fig. 1.9, the blue block represents a bi-predicted block.

Thus, in addition to the 2D motion vector, the reference frame index and the list index must be signaled to the decoder.

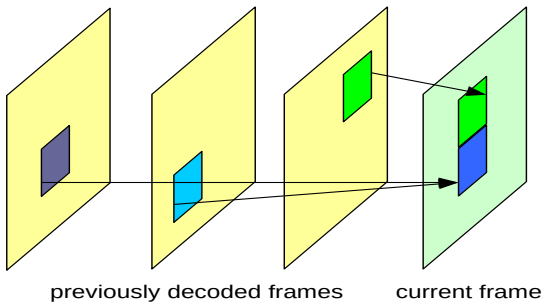


Figure 1.9: Multiple reference frames for inter prediction.

1.2.1.6 Coding mode selection

In H.264/AVC, the selection of the coding modes is not normative, thus its implementation in an encoder is open. The mode selection is usually made by minimizing a cost function. The cost function is generally based on a distortion criterion, or a rate/distortion optimization (RDO) criterion. The latter criterion is the most popular, as it yields the best coding performances, and was for example implemented in the test model of H.264/AVC. In return, it usually requires an increased complexity.

In order to select the best mode among the modes available in Intra and Inter prediction, the RDO criterion aims at minimizing the distortion under a rate constraint, or minimizing the rate under a distortion constraint [45][46]. For an input signal X and a mode μ, the criterion is formally defined as:

$$\min_{\mu} D(X, \mu), \text{ s.t. } R(X, \mu) \leq R_c \tag{1.2}$$

or:

$$\min_{\mu} R(X, \mu), \text{ s.t. } D(X, \mu) \leq D_c \tag{1.3}$$

where D refers to a distortion measure function, R refers to a rate measure function, and D_c and R_c are the constraint on the distortion and the rate respectively. In practice, the solution to this twofold minimization problem is found by minimizing the following Lagrangian function:

$$J_{\lambda}(X, \mu) = D(X, \mu) + \lambda R(X, \mu) \tag{1.4}$$

where λ is a Lagrangian parameter which allows to balance the two above criteria.

1.2.2 HEVC

HEVC is the latest standard developed by the Joint Collaborative Team on Video Coding (JCT-VC), finalized in January 2013. HEVC is a conceptual generalization of the design of H.264/AVC [2], introducing optimizations which allow to reduce about 50% of the bit-rate compared to H.264/AVC for a similar quality [35].

In particular, larger prediction and transform blocks are used, which are more efficient for modern high resolution sequences. A new flexible partitioning for the prediction and transform blocks is also introduced.

1.2.2.1 Hierarchical syntax

The high level syntax elements, sequence, frame, and slice, are not modified compared to H.264/AVC (see Fig. 1.10). One of the main change of HEVC over H.264/AVC is the replacement of the macroblock concept by the notion of coding tree unit (CTU). In HEVC, a frame is partitioned using a recursive quadtree structure, and new hierarchical levels are defined:

- the **Coding Tree Unit (CTU)**, is the basic processing unit. A slice is thus divided into CTUs. The size of the CTU is defined at the encoder and can be either 16×16, 32×32 or 64×64.

- the **Coding Unit (CU)**, is the coding element into which the CTU is divided. The CU itself can then be recursively divided into four CUs, creating a quatree structure, as shown in Fig. 1.11. The root of the quadtree partitioning is at the CTU level. The size of the CU varies between 64×64 and 8×8. The Intra or Inter prediction mode decision is made at the CU level.

- the **Prediction Unit (PU)**, contains the information related to the prediction process (e.g. directional mode for Intra prediction or motion vector for Inter prediction). A PU can be recursively partitioned into PUs. The root of a PU partitioning is at the CU level, and the PU size can be from 64×64 down to 4×4.

- the **Transform Unit (TU)**, is the basic unit for transform and quantization. A TU can be recursively partitioned into TUs, and the root of a TU partitioning is at the CU level. The TU size can be from 32×32 down to 4×4. The TU partitioning is independent from the PU partitioning, as illustrated in Fig. 1.11, where the PUs are represented with blue lines while the TUs are represented with red dotted lines.

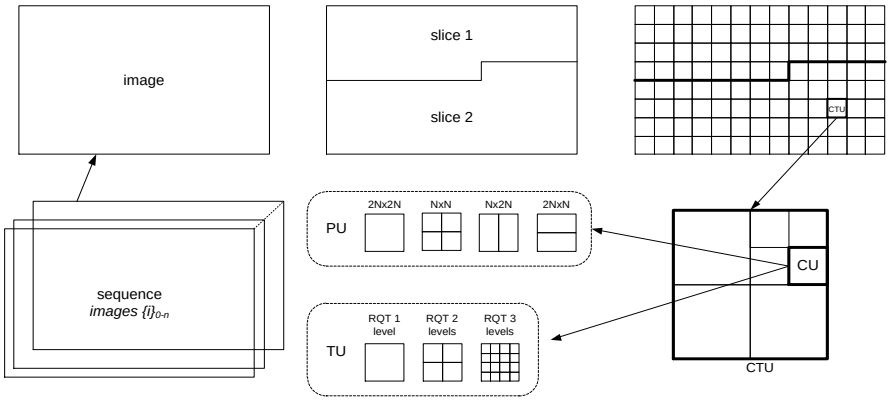


Figure 1.10: Hierarchical levels of a sequence in HEVC.

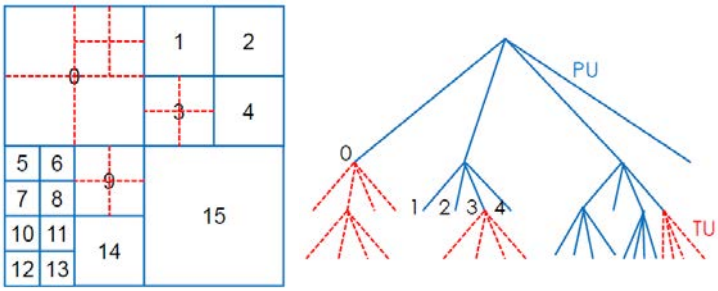


Figure 1.11: Recursive partitioning of a CTU based on a quadtree structure.

1.2.2.2 Types of frame

The same types of frame are defined in HEVC as in H.264: I, P and B frames.

1.2.2.3 Encoder structure

The encoder structure of HEVC is very similar to the one of H.264/AVC, as shown in Fig. 1.12.

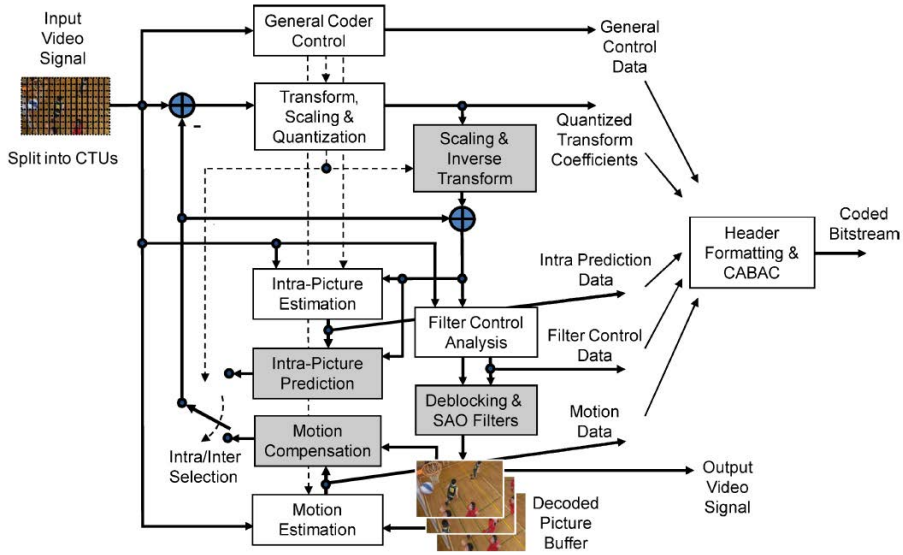


Figure 1.12: Typical HEVC video encoder (with decoder modeling elements shaded in light grey). Source: [2]

1.2.2.4 Intra prediction

The Intra prediction in HEVC relies on the same principle as the Intra prediction in H.264/AVC. However, the concept is optimized, as 33 directional modes are considered in addition to a DC (averaging of the neighboring pixels) mode and a planar mode (see Fig. 1.13). The Intra prediction is applied on square blocks of size 4×4 to 32×32.

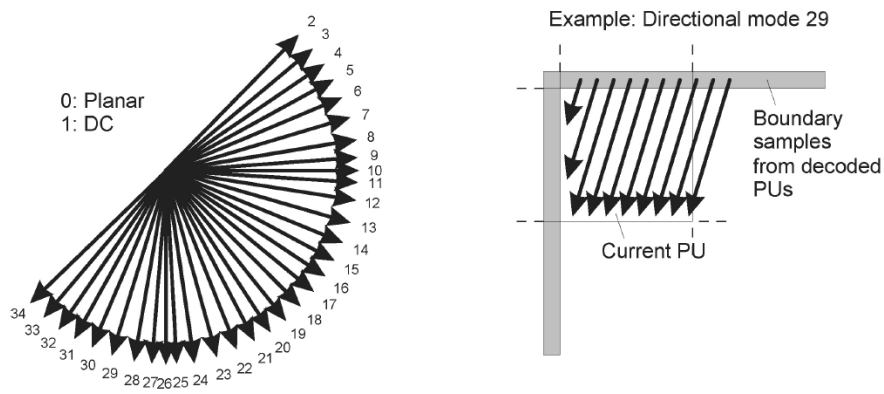


Figure 1.13: Modes and directional orientations for Intra prediction. Source: [2].

The coding of the best Intra mode in HEVC relies on the concept of most probable mode (MPM). Three MPMs are derived from the Intra prediction modes of the above and left PU. If the current best Intra mode is one of the three MPMs, the index of this

MPM is sent to the decoder. Otherwise, the best Intra mode is directly coded using a 5 bits fixed length code.

1.2.2.5 Inter prediction

The Inter prediction in HEVC is based on motion estimation/compensation. As in H.264/AVC, the motion estimation is usually performed using a BM algorithm (see Fig. 1.7).

Motion vectors can be obtained with a quarter-pixel accuracy. In HEVC, a 8-tap filter is used to produce the half-sample position and a 7-tap filter is used to produce the quarter sample positions.

The main optimization for Inter prediction in HEVC compared to H.264/AVC relates to the prediction of the motion vector, prior to differential encoding. In HEVC, several motion vector predictors (MVP) derived from spatially/temporally neighboring PUs are in competition, and the index of the best one is signaled to the decoder. Three different modes are considered for the derivation of the list of MVP candidates: Inter, Merge, and Merge-Skip. For the Inter mode, the texture residual is coded along with MVP index and the motion vector residual. For the Merge mode, the texture residual is coded along with MVP index, and the motion data are derived from the MVP. For the Merge-Skip mode, no texture residual is coded, only the MVP index is signaled, and the motion data are derived from the MVP.

The Inter prediction is applied on blocks of size 8×8 to 64×64. In addition to square partitions, asymmetric rectangular partitions are also considered.

As in H.264/AVC, bi-prediction is also allowed for the B frames.

1.2.2.6 Coding mode selection

As for H.264/AVC, the coding mode selection in HEVC is not normative. However, it is usually based on a RDO criterion, solved in practice by the minimization of the Lagrangian function of Equation 1.4.

1.2.3 SHVC

HEVC has been further extended in a scalable version named SHVC [47]. SHVC support different scalability features such as temporal, spatial, SNR, bit-depth, color gamut, and hybrid codec scalability.

An example of SHVC decoder is given in Fig. 1.14 for a two-layer scalable system, but SHVC supports up to height layers. The two-layer scalable system consists in a base layer (BL) and an enhancement layer (EL).

The EL codec is a modified version of HEVC, but was designed to minimize the changes in the codec architecture compared to a single layer HEVC. Thus, the reconstructed BL frames are added in the EL reference frames buffer at a specific index, after appropriate inter-layer processing. The inter-layer prediction then consists in performing Inter prediction in this specific reference frame with a zero-motion constraint.

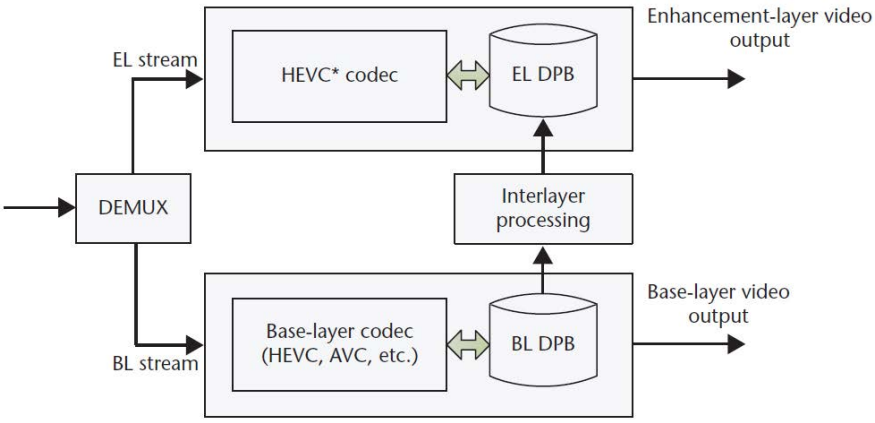


Figure 1.14: SHVC decoder architecture. The modified EL decoder is indicated as HEVC*. Source: [47].

Since the inter-layer process relies on the reconstructed BL frames, the BL codec can be considered as a black box, which allows the hybrid codec scalability. In the case where the BL codec is HEVC, a motion field re-sampling is performed in order to predict the EL motion field.

Other tools for inter-layer processing include notably texture re-sampling for spatial scalability. The texture re-sampling is performed using 8-tap filters, which support arbitrary spatial ratios.

The spatial scalability feature refers to the case where the spatial resolution of the EL is greater than the one of the BL. Similarly, the temporal scalability feature refers to the case where the frame rate of the EL is greater than the one of the BL. The SNR scalibility concerns the case where both layer share the same spatial resolution, but the EL is transmitted with a higher quality than the BL, in practice by setting a lower QP for the EL than the BL. The bit-depth scalability indicates that the bit-depth of the EL is higher than the one of the BL. The color gamut scalability specifies that the color formats of the two layers are different. Finally, the hybrid codec scalability allows to encode the EL with SHVC, while the BL is encoded with a different codec, such as H.264.

Chapter 2

Inter prediction methods based on linear embedding for video compression

Video compression schemes achieve data compression by exploiting similarities within frames (i.e., the spatial redundancy), as well as between the target frame and one or several reference frames (i.e., the temporal redundancy). Intra coding techniques are used to reduce the spatial redundancy within each frame separately, whereas Inter coding techniques are used to reduce the temporal redundancy between successive frames of a video sequence.

The block matching (BM) algorithm is a popular technique to perform the motion estimation/compensation used in the standards for Inter prediction. However, alternative technique were introduced to perform motion estimation/compensation, based on a template matching (TM) technique [6]. The methods exploits the correlation between the current block and a pre-defined set of neighboring pixels, called the template of the current block. Rather than looking for the most correlated block in the reference frames, one looks for the most correlated template. The block which is adjacent to this template is used as a predictor for the current block. The motion compensation is performed using the exact same process, so no motion information needs to be transmitted to the decoder. This technique efficiency has also been demonstrated for Intra prediction [7]. The RD performance of this method can be improved by using a weighted combination of multiple predictors. Initially a simple averaging of the predictors was performed [8][9], but methods using adaptive weights, e.g. using sparse approximation [10][11], were shown to bring significant improvements.

In this chapter, we consider an approximation method called Locally Linear Embedding (LLE), introduced in [48] for data dimensionality reduction, which we adapt to the problem of temporal prediction. The LLE technique has already been shown to be very efficient for Intra prediction in [12][13]. However, the derivation from Intra to Inter prediction is not trivial, mainly because the proposed techniques are now in competition with the motion estimation/compensation, which is a more efficient prediction

tool than the Intra directional modes. The idea is to first search for a representation of the template as a linear combination of K nearest neighbor templates (called K-NN templates) taken from a search window denoted SW. The linear combination coefficients (or weights) are then applied on the blocks adjacent to the K-NN templates to yield the current block predictor. The LLE weights are computed using a least square formulation of the template approximation problem under the constraint that they sum to one.

The K-NN search strategy has a strong impact on the predictor quality. In fact, the TM technique efficiency rely on the hypothesis that the template and its adjacent block are well correlated. First, we proposed a direct derivation of the TM technique, where the K-NN can be found by computing distances between the template of the current block and those of candidate blocks in the reference frames. This method is denoted Template Matching LLE (TM-LLE) and, as for the TM method, no side information (i.e., no motion vector) needs to be sent to the decoder. A variant of this method is introduced where the first neighbor is searched by template matching (as in TM-LLE), but the remaining $(K-1)$-NN are found by computing a distance between the complete patch formed by the template and adjacent block of the first neighbor and the candidate patches in the search window. The method is denoted Improved Template Matching LLE (ITM-LLE).

Second, to further improve the K-NN search, we introduce a method enforcing the correlation between the templates and their adjacent blocks, but requiring the transmission of side information to the decoder. Thus, we propose a method where the K-NN search is initialized with a block-matching algorithm. This implies that a motion vector is sent to the decoder. We then find the remaining $(K-1)$-NN as in ITM-LLE. This method is named Block-Matching LLE (BM-LLE).

Finally, we propose an improved variant of the ITM-LLE method, denoted optimized ITM-LLE (oITM-LLE). In this method, we basically obtain L predictors by running L times the ITM-LLE method. The best iteration in a RD sense is retained, and its index is sent to the decoder.

The experiments and their analysis focus on RD performance evaluations of the proposed prediction methods against the standard reference techniques: directional and motion estimated/compensated prediction modes of H.264 and template matching averaging (TM-A). This analysis is carried out using a legacy H.264 implementation, but note that the proposed techniques are still applicable in HEVC, since the Inter prediction tool in HEVC follow the same principles as those used in H.264. Simulation results show that significant RD performance improvements are achieved compared to the reference prediction methods. The performed analysis includes elements of complexity in terms of execution times measured at the encoder.

This chapter is organized as follows. Section 2.1 reviews background on prediction methods based on template matching. Section 2.2 describes the proposed LLE-based temporal prediction techniques. Section 2.3 explains how the proposed prediction methods have been used in an H.264 codec. We then give the PSNR-rate performance gains compared to the reference H.264 codec.

2.1 Background: Prediction based on Template Matching

In addition to the standard prediction modes presented in the previous chapter, prediction methods based on Template Matching (TM) have been considered, for both Intra [7] and Inter [6] prediction. The TM method is very close to the BM method, although here the template pixels (on the top and to the left of the blocks) are used to find the best match instead of the pixels in the current block to be predicted (see Fig. 2.1).

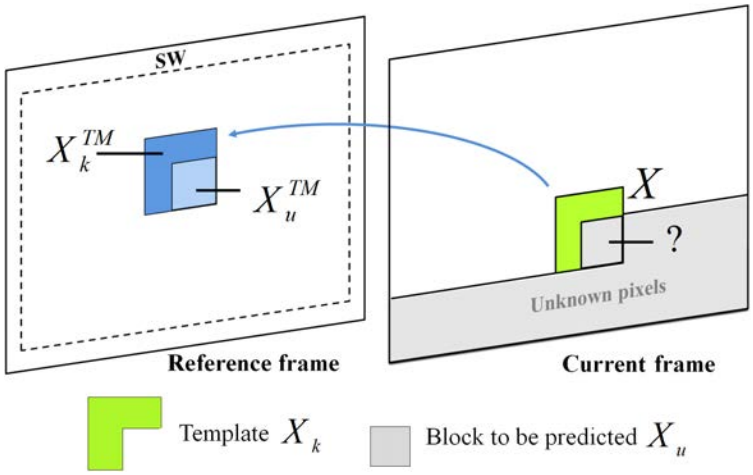


Figure 2.1: Template Matching for Inter prediction.

The union of the template X_k and of its adjacent block X_u forms the patch X. The underlying basic idea of the algorithm is to take advantage of a supposed correlation between the pixels in the block and those in its template. For Inter prediction, the first step of the algorithm is to look for the nearest neighbor (NN) of the template in a search window defined in one or more reference frames. As for the BM algorithm, the metric used to find the NN can be the sum of absolute difference (SAD) or the sum of square error (SSE). Once the NN X_k^{TM} of X_k is found, the adjacent block X_u^{TM} of this template is used as a predictor for the current block X_u. The benefit of this method is that this prediction process can be reproduced at the decoder, hence no side information (motion vector, list index, reference frame index) needs to be sent to the decoder anymore [49]. Although efficient in terms of bit-rate reduction in the case of homogeneous texture, the method yields low quality predictors in image areas where the block and its template are not well correlated.

The technique has been extensively studied in H.264 [6][7][49], and was considered for Intra prediction in the early stage of HEVC [50]. Even though it was not retained for the standard, it was later shown that it can improve the RD performance of HEVC, e.g. when used for Inter bi-prediction in combination with block matching [51].

In order to improve the predictor quality, a so-called template matching averaging (TM-A) method has been proposed in [9]. In this method, one looks for the K nearest neighbors (K-NN) of the current template and not only the first one. The predictor of the current block X_u is then obtained by averaging the K blocks adjacent to the K-NN

of the template. This enables to smooth the predictor, which is advantageous most of the time, and computationally reasonable.

Note that different approaches relying on K-NN combination have been explored for Intra or Inter coding, such as sparse representations [10][11][52], or Nonnegative Matrix Factorization [12][13]. However, results in [11] show that neighbor embedding techniques such as LLE or NMF outperform sparse representations in terms of RD performances. In [13], results show that NMF performs slightly better than LLE in terms of RD performance, but complexity for NMF is much higher than LLE, especially at the encoder side. These conclusions motivate the use of the LLE in this chapter, described in the following section. More recently, prediction methods based on weighted template matching have been proposed to improve HEVC Intra RD performances, e.g. in [18]. The method presented in [18] demonstrates the efficiency of weighted template matching against HEVC Intra mode. However, this method is optimized to reduce complexity, e.g. by using tabulated exponential weights, while we focus on optimizing the RD performances by using optimal weights in a least square sense.

2.2 Inter prediction based on LLE

This section describes the proposed Inter prediction techniques using LLE. LLE-based prediction methods search for the linear combination of K nearest neighbors which best approximates in a least squares sense the template of the current block, under the constraint that the weights of the linear combination sum to one. The block predictor is computed by applying the found weights to the blocks which are adjacent to the K-NN templates. The section below first presents the weights computation, assuming that the K-NN are available. We then describe the proposed K-NN search strategies.

2.2.1 LLE-based predictor computation

The weighting coefficients are computed by formulating the template approximation problem as a least squares problem, under the constraint that the weights sum to one. The found weighting coefficients are applied in the linear combination of the adjacent block pixels in order to compute the block predictor (see Fig. 2.2).

Let $\mathbf{A} = [\frac{\mathbf{A}_k}{\mathbf{A}_u}]$ denote a so-called dictionary represented by a matrix of dimension $N \times K$. The columns of the dictionary $\mathbf{A}$ are constructed by stacking the K candidate texture patches found after the K-NN search step. The sub-matrices $\mathbf{A}_k$ and $\mathbf{A}_u$ contain the pixel values of the templates and of the blocks respectively. Let $X = [\frac{X_k}{X_u}]$ be the vector composed of the known pixels of the template X_k and the unknown pixels of the current block X_u.

The LLE-based prediction problem can be re-written as:

$$\min_V \|X_k - \mathbf{A}_k V\|_2^2 \text{ s.t. } \sum_m V_m = 1 \tag{2.1}$$

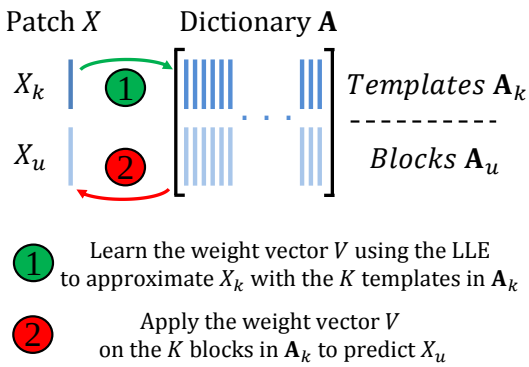


Figure 2.2: Predictor computation technique based on LLE. Assuming the dictionary $\mathbf{A}$ is available, a weighting coefficient vector V is learned using the LLE, which approximates the current template X_k with the K templates in $\mathbf{A}_k$. The current block predictor $\hat{X}_u$ is obtained by applying the learned weights to the K blocks in $\mathbf{A}_u$.

where V denotes the optimal weighting coefficients vector which is computed as

$$V = \frac{\mathbf{D}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{1}}. \tag{2.2}$$

The term $\mathbf{D}$ denotes the local covariance matrix (*i.e.*, in reference to X_k) of the selected K-NN templates stacked in $\mathbf{A}_k$, and $\mathbf{1}$ is the column vector of ones. In practice, instead of an explicit inversion of the matrix $\mathbf{D}$, the linear system of equations $\mathbf{D}V = \mathbf{1}$ is solved, then the weights are rescaled so that they sum to one.

The predictor of the current block $\hat{X}_u$ is then obtained as:

$$\hat{X}_u = \mathbf{A}_u V \tag{2.3}$$

2.2.2 Template-based K-NN search for LLE-based prediction

This section first describes two LLE-based methods in which the K-NN search is not based on real motion estimation but rather on template matching, and thus does not need to send side information to the decoder.

The first K-NN search method simply looks for the K-NN of the current template (see Fig. 2.3). The K patches stacked in the dictionary $\mathbf{A}$ thus consist in the K templates found ($\mathbf{A}_k$), along with their adjacent blocks ($\mathbf{A}_u$). The current block can then be predicted as explained in section 2.2.1. This K-NN strategy is as simple as the one of the TM-A method, however it suffers from the same limitations in the sense that the candidate patches found may not lead to the best predictor. This method, referred to as Template Matching LLE (TM-LLE), does not require transmitting extra side information to the decoder which can perform the same K-NN search.

An improved variant of the previous method is to find the first NN of the current template in a first step. Then in a second step, the $(K-1)$-NN of the previous first patch

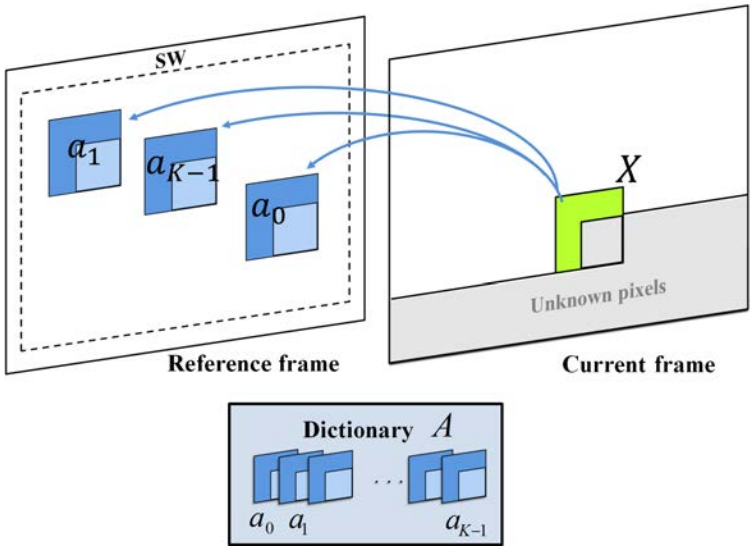


Figure 2.3: Search for the K-NN of the current template in the TM-LLE method.

are found in order to build the complete dictionary $\mathbf{A}$. Note that these $(K-1)$-NN are now found with respect to the whole patch, which tends to reinforce the correlation between the templates and the blocks (inner correlation of the patch). However, if the first NN found is not well correlated with the current patch, the predictor quality will decrease. This method is referred to as Improved Template Matching LLE (ITM-LLE).

2.2.3 Block-based K-NN search for LLE-based prediction

To better overcome the potential lack of correlation between the blocks and their templates, we propose to use the current block to guide the K-NN search. This implies that additional information needs to be sent to the decoder, so that it can find the exact same K-NN. The following methods use the current block to find the first NN. The $K-1$ remaining patches are found with respect to the previous NN. To signal the NN to the decoder, a motion vector is transmitted.

The prediction first proceeds by computing the motion vector using a classical block matching algorithm. The best matching block (the first nearest neighbor) X_u^{BM} of the current block X_u is found. The patch containing this block $X^{\mathrm{BM}} = [\frac{X_k^{\mathrm{BM}}}{X_u^{\mathrm{BM}}}]$ is used as the first patch of the dictionary for the LLE. The second step is to search for the $(K-1)$-NN $[\mathbf{a}_1, ..., \mathbf{a}_{K-1}]$ of the patch X^{BM}. The union of these K patches $\mathbf{A} = \left[X^{\mathrm{BM}}, \mathbf{a}_1, ..., \mathbf{a}_{K-1}\right]$ forms the dictionary (see Fig. 2.4) used for the LLE computation described above by equations (2.1), (2.2) and (2.3). The motion information can then be encoded as in the reference codec (e.g. H.264 or HEVC), which does not increase the corresponding rate cost.

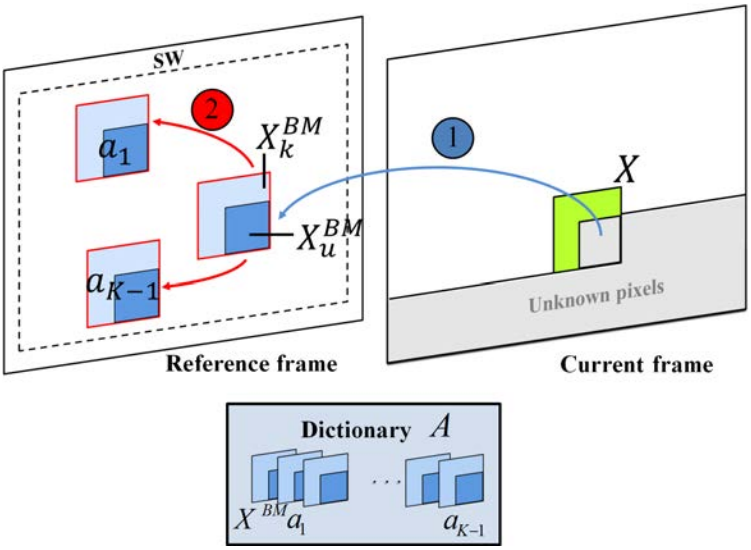


Figure 2.4: Illustration of the first two steps of the BM-LLE method.

2.2.4 Optimized template-based K-NN search for LLE-based prediction

To further enhance the RD performance, the ITM-LLE method was optimized. First, L nearest neighbors to the template of the current block are found. For each patch c_l found by this L-NN search, a dictionary is constructed leading to a set of L dictionaries $\mathbf{A}^0, ..., \mathbf{A}^{L-1}$. Each dictionary $\mathbf{A}^l$ is formed by stacking the patch c_l and its $(K-1)-NN$ (see Fig . 2.5). Since the patches are found using only the template of the current block, the same set of dictionaries can be found by the decoder. An LLE-based predictor is computed with each dictionary and the dictionary $\mathbf{A}^{l_{opt}}$ giving the best RD performances is retained, and its index l_{opt} is signalled to the decoder. The number of dictionaries L is taken as a power of 2, and the index is coded with a fixed length binary code. This method is referred to as optimized ITM-LLE (oITM-LLE). The different steps of the algorithm are detailed as pseudo-code in Algorithm 1.

2.3 Simulations and results

2.3.1 Integration in the MPEG-4 AVC/H.264 coding scheme

The reference and proposed methods presented in sections 2.1 and 2.2 respectively have been tested in a H.264 framework. Note that only the TM-A was considered and not TM, as TMA gives higher performances. The TM-A and the LLE-based prediction methods have been introduced in the coding scheme in competition with the existing H.264 inter and intra prediction modes. This solution was chosen over a simple replacement of the existing motion compensation prediction (MCP) method by an Inter LLE-based method because they are complementary. In fact the MCP method is already

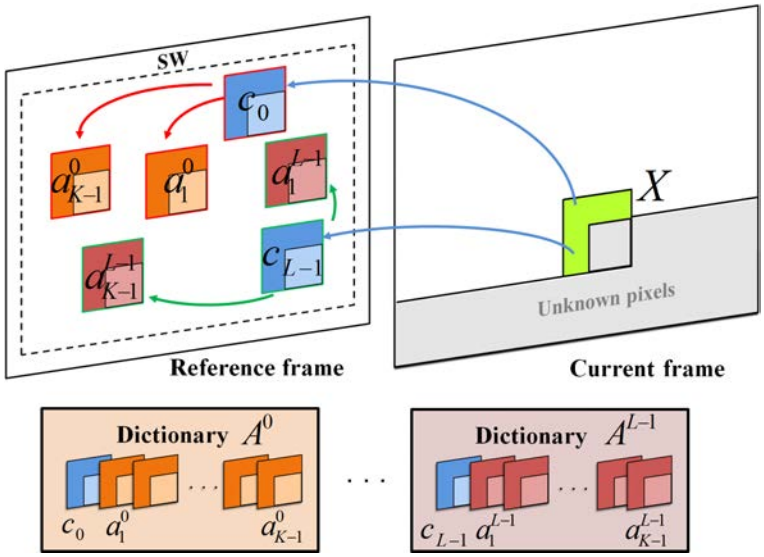


Figure 2.5: Example of construction of A^0 and A^{L-1} dictionaries from candidate patches c_0 and c_{L-1} (oITM-LLE method).

quite efficient, e.g. for smooth textures with little or no motion. A typical example is a static background where the Skip mode can be used. Multi-patches methods such as the proposed ones using LLE are better at predicting high frequency pseudo-periodic textures, which are not easy to reconstruct using only one block (as shown in Fig. 2.6).

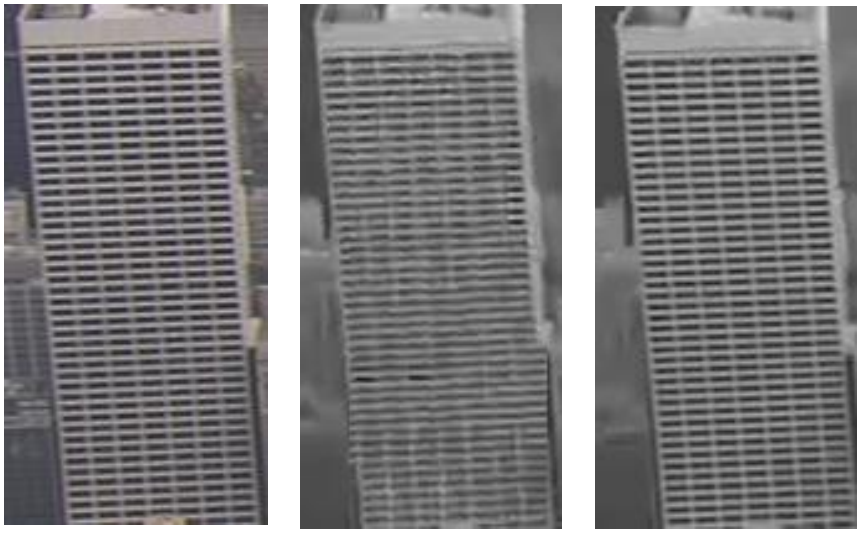


Figure 2.6: Left: close-up of a building from the City source sequence. Middle: H.264 prediction. Right: oITM-LLE prediction.

Algorithm 1 oITM-LLE prediction method

Input: X, K, L
Output: current block predictor $\hat{X}_u$
Determine the L nearest neighbors of the current template X_k, *i.e* the templates $\mathbf{c}_{k_0}, ..., \mathbf{c}_{k_{L-1}}$ such that $d_0 \leq ... \leq d_{L-1}$ with $d_i = ||X_k - \mathbf{c}_{k_i}||$
Retain the L patches associated with the L templates determined in the previous step: $\mathbf{c}_i = [\frac{\mathbf{c}_{k_i}}{\mathbf{c}_{u_i}}], i = 0, \ldots, L-1$
for $l = 0 \rightarrow l = L-1$ **do**
  Find $K-1$ neighbors close to $\mathbf{c}_l$, *i.e.*, the patches $\left[\mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
  Set $\mathbf{A}^l = \left[\mathbf{c}_l, \mathbf{a}_1^l, ..., \mathbf{a}_{K-1}^l\right]$
  Retrieve $\mathbf{A}_k^l = \left[\mathbf{a}_{k_0}^l, ..., \mathbf{a}_{k_{K-1}}^l\right]$ and $\mathbf{A}_u^l = \left[\mathbf{a}_{u_0}^l, ..., \mathbf{a}_{u_{K-1}}^l\right]$ from $\mathbf{A}^l$. Note that here $\mathbf{c}_l = [\frac{\mathbf{a}_{k_0}^l}{\mathbf{a}_{u_0}^l}]$
  Solve the constrained least squares problem:
  $\min_{V^l} \left\|X_k - \mathbf{A}_k^l V^l\right\|_2^2$ s.t. $\sum_m \mathbf{V}_m^l = 1$
  Get the predictor:
  $\hat{X}_u^l = \mathbf{A}_u^l V^l$
  Compute the corresponding RD cost RD^l
end for
Select the optimum $l_{opt} = \arg\min_l RD^l$
Set $\hat{X}_u = \mathbf{A}_u^{l_{opt}} V^{l_{opt}}$

The MCP method is used for block partitions going from size 16×16 to 8×8. Experiments using the H.264 reference software show that further sub-partitioning into 8×4, 4×8 and 4×4 blocks brings little improvement in terms of coding performance, while increasing the complexity. In fact, the signaling cost of the motion information usually becomes prohibitive for such small partitions. The TM-A and our LLE-based prediction methods are applied on 8×8 blocks.

The Skip mode is allowed for both P and B frames, as well as the additional bi-predictive mode for the B frames. The Inter TM-A or LLE-based method to be tested is introduced in competition with the MCP method for 8×8 partitions only, and for both P and B frames. We set the template width to 3 pixels. The choice between the classical MCP method or the additional TM-A or LLE-based method is based on a RDO criterion. The syntax then needs to be modified to send a flag to the decoder indicating which method is selected. The rate is therefore increased by 1 bit for each partition that features an additional Inter prediction method (TM-A or LLE-based). The index l_{opt} of the selected dictionary is also transmitted using a fixed length code when the oITM-LLE method is retained for predicting the current block. Experiments showed that the bit-rate corresponding to the flag for the selected method amounts in average to about two to three percents of the full bit-rate, while the bit-rate of the dictionary index reaches four to five percents. Note that this extra information bit-rate could be further reduced using the CABAC.

The search window for the Inter TM-A or Inter LLE-based prediction methods is defined in the reference frames of the L_0 list for the P frames, of the L_0 and L_1 lists for the B frames, and in the decoded part (causal part) of the current frame for both P and B frames. Note that if an LLE-based prediction method is applied to Intra prediction, the search window is only defined in the causal window. The search window is centered

Table 2.1: Simulations parameters

Parameter	**Setting**
Sequence type:	IBBPBBP
Number of encoded frames:	31 (Matrix: 34)
Number of reference frames:	
- for BM:	4
- for K-NN:	4 (& causal part)
Search-range:	
- for BM:	32 pel for CIF
	64 pel for 720p
- for K-NN:	32 pel for all
Search method:	
- for BM:	Fast Full Search
- for K-NN:	Full Search
Search accuracy:	
- for BM:	Quarter Pel
- for K-NN:	Full Pel
Block sizes for intra:	all 16x16 to 4x4
Block sizes for MCP:	all 16x16 to 8x8
Block sizes for LLE:	8x8
Template size:	+3 pel on width and height
Quantization parameter (I/P/B):	22/23/24, 27/28/29,
	32/33/34, 37/38/39
Number L of dictionaries for oITM-LLE	32

on the position of the current block.

Note that the solution we propose to integrate our method in the H.264 codec is still valid in HEVC. In fact, the prediction tools in HEVC follow the same principles as those used in H.264. Detailed explanations are given in the perspectives (section 2.4).

2.3.2 Experimental conditions

The proposed schemes have been implemented in the latest release of the JM-KTA software [53], in order to be compared with the H.264 prediction modes. Simulations have been run using 5 test sequences presenting different characteristics in terms of motion and texture (see Fig. A.1 in annex, section A), and having different resolutions (three CIF sequences and two 1280×720 sequences). The CIF sequences are respectively made of 31 frames extracted from the Foreman sequence (frames 149-179), characterized by fast motion and smooth natural texture, 31 frames extracted from the "Rushes" sequence (frames 597-627) containing high frequency texture and complex motion, and 34 frames extracted from the trailer of the Matrix movie[1] (frames 1810-1843). This sequence was chosen because of its fast scene changes, occurring every 2-3 frames. The different scenes contain little motion but cover a wide range of textures. The 1280×720 sequences are made of 31 frames extracted from the "City" (frames 230-260) and the "Spincalendar" (frames 500-530) sequences respectively. The urban scene in the "City" sequence contains non-stochastic high frequency textures, with a slow camera motion.

[1] ©1999-2015 Warner Bros.

The "Spincalendar" sequence contains different textures following a smooth rotation motion. The modified encoder (see section 2.3.1) has been configured with the Main profile [1] and the parameters given in Table 2.1. The rate gains are obtained using the Bjontegaard measures [54]. The complexity is measured trough the percentage of the tested encoder processing time over the one of the H.264 reference encoder.

Due to different experimental conditions, the given complexity values correspond to averaged estimates with a standard deviation of about 10 %. Note that the main goal of contribution in this chapter is to assess the coding performances of the different methods, and thus we did not focus on an optimized development. For this reason, and although we are aware that TM-based methods increase the decoder complexity, we only give some elements of complexity at the encoder side. The main complexity increase is due to the K-NN search, for which fast methods exist [14][15], and the LLE weights computation. However, contrary to the encoder side, at the decoder side it only affects the blocks for which the LLE-based prediction method has been selected. As an example, in section 2.3.4 the LLE-based prediction methods account for about 20 % of the selected modes.

In the following section, we first analyze the performances of the proposed methods when used for Inter prediction. The performances are analyzed as a function of the key parameters: the number K of nearest neighbors, the reference frames number and the search window size. We also give the corresponding complexity. Second, the methods are also assessed when used for Intra prediction only or when used for both Intra and Inter prediction.

2.3.3 RD performance analysis & elements of complexity

In this section we discuss the gains in terms of bit-rate reduction of the proposed methods against the H.264 reference, along with the execution times measured at the encoder. The values are averaged over the full panel of test sequences.

2.3.3.1 Impact of the parameter K

Fig. 2.7 shows how the bit-rate savings for each method vary as a function of the parameter K. Fig. 2.8 shows the corresponding complexity for each method, except the oITM-LLE method, which reaches significantly higher levels (about 5000 to 6000 %). In terms of RD performance, the TM-LLE, ITM-LLE and oITM-LLE methods outperform the TM-A method, which demonstrates the better adaptation of the LLE weights compared to a simple averaging, especially when K increases. In average, the highest bit-rate saving is achieved with the oITM-LLE method, reaching more than 8 %.

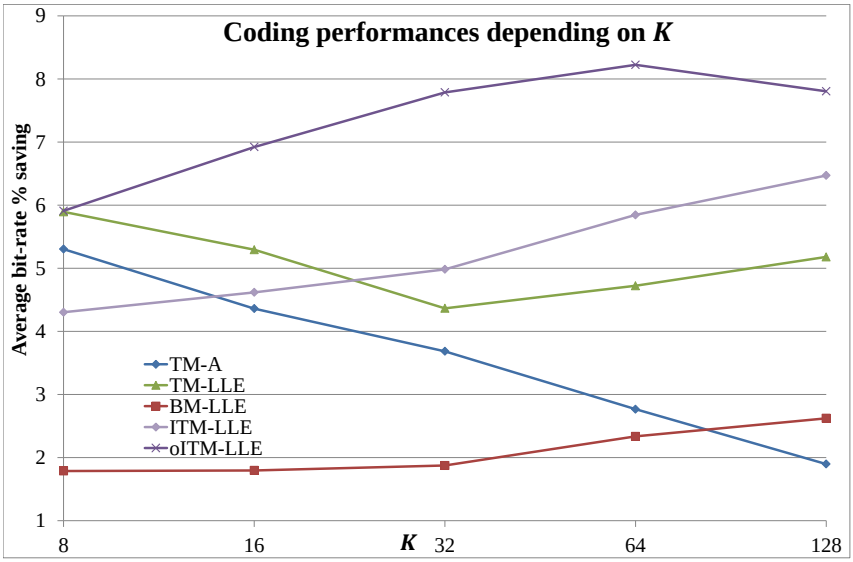


Figure 2.7: Coding performances obtained with the different proposed prediction methods as a function of the number K of nearest neighbors.

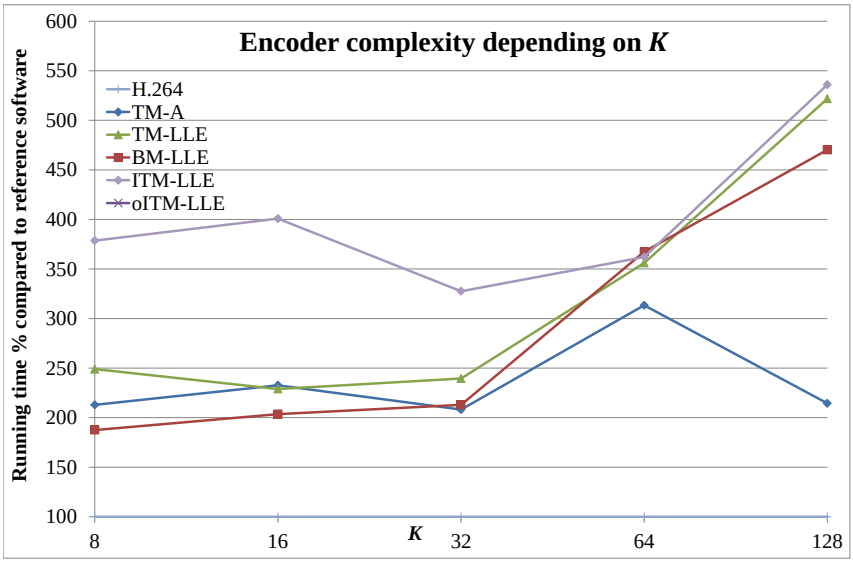


Figure 2.8: Encoder complexity obtained with the different proposed prediction methods as a function of the number K of nearest neighbors.

In Table 2.2, we give, for each method, the K value tuned to obtain a satisfying trade-off between the execution time and the coding gain (from Fig. 2.7 and Fig. 2.8), which are also given. We can see that the TM-LLE and ITM-LLE methods outperform

the state-of-the-art methods in terms of RD performances. The TM-LLE method is also competitive in terms of complexity. However, the BM-LLE is only competitive in terms of complexity, but not in terms of coding gains. The oITM-LLE does produce the best results in term of bit-rate reduction, but at a really high cost in terms of execution time. For the aforementioned reasons, the detailed analysis of the BM-LLE and oITM-LLE methods is not pushed further for Inter prediction. We will see however in section 2.3.4 that, when used for Intra prediction, the oITM-LLE method can be efficiently combined with less complex Inter prediction method such as TM-LLE. For the next simulations, K is set to the values presented in Table 2.2.

Table 2.2: K values set to achieve a trade-off between RD performances and complexity.

Method	K value	Execution time in %	Bit-rate gain in %
TM-A	8	213	-5.30
TM-LLE	8	249	-5.89
BM-LLE	16	204	-1.80
ITM-LLE	64	362	-5.85
oITM-LLE	32	5077	-7.79

2.3.3.2 Impact of the reference frames number

Fig. 2.9 shows how the bit-rate gains for each method vary as a function of the number of reference frames. Fig. 2.10 shows the corresponding complexity. In terms of RD performance, we can see that the proposed methods can outperform the TM-A and the highest bit-rate reduction is achieved with the ITM-LLE method, reaching 6.33 %.

In Table 2.3, the reference frames number is set in order to achieve a satisfying trade-off between complexity and RD performances (from Fig. 2.9 and Fig. 2.10). The corresponding execution times and bit-rate reductions are given. We can see that the TM-LLE method can perform better than the TM-A method in terms of RD performances for a similar complexity. The ITM-LLE can achieve the highest bit-rate reduction, which requires an increased complexity.

Table 2.3: Reference frames number set to achieve a trade-off between RD performances and complexity.

Method	Reference frames number	Execution time in %	Bit-rate gain in %
TM-A	3	157	-5.14
TM-LLE	3	156	-5.83
ITM-LLE	1	276	-6.33

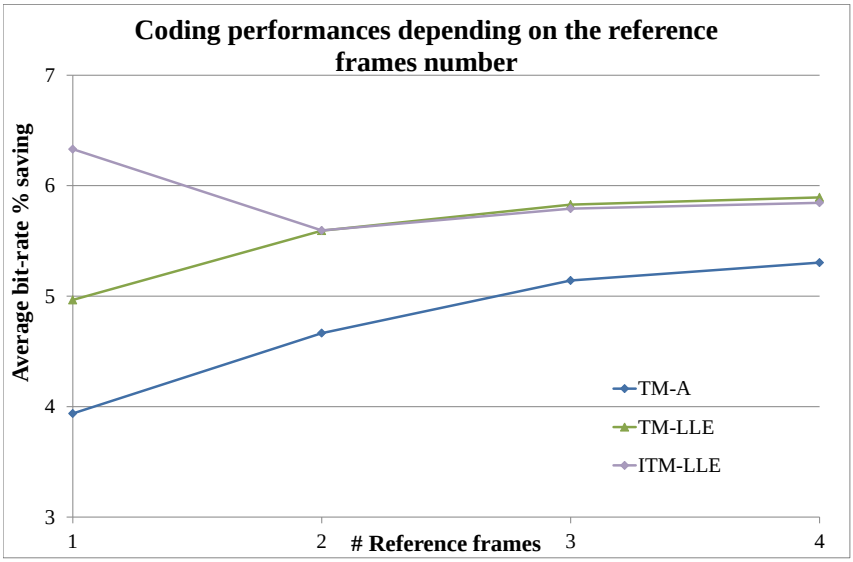


Figure 2.9: Coding performances obtained with the different proposed Inter prediction methods as a function of the number of reference frames.

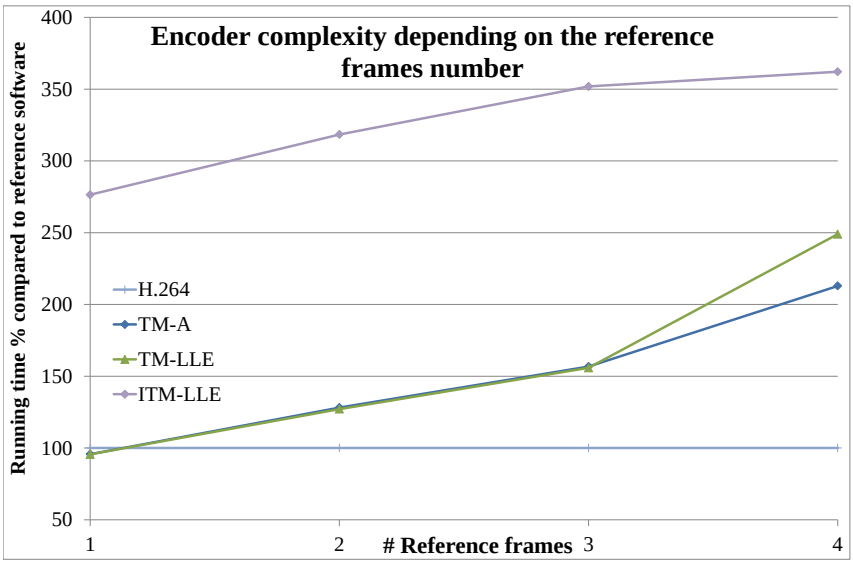


Figure 2.10: Encoder complexity obtained with the different proposed Inter prediction methods as a function of the number of reference frames.

2.3.3.3 Impact of the search window size

Fig. 2.11 shows how the bit-rate reductions for each method vary as a function of the search window range. Fig. 2.12 shows the corresponding complexity. Table 2.4 gives the coding gains and corresponding execution time when the search window range is set to achieve a satisfying trade-off between RD performances and complexity. The number of reference frames is not fixed, and set depending on this range. The search window range is first set to 16, with 4 reference frames, then set to 32, with 4 reference frames, and finally set to 64, with 1 reference frame. The last configuration was chosen with only 1 reference frame in order to limit the complexity. However, the same amount of patches is available for the K-NN search for the last two configurations.

We can see that the proposed methods can outperform the TM-A, reaching up to 7.20 % bit-rate savings for the ITM-LLE method. As for the previous results, the highest bit-rate reduction can be achieved by the ITM-LLE method, while the TM-LLE method allows a better trade-off between RD performances and complexity.

Table 2.4: Search window size set to achieve a trade-off between RD performances and complexity.

Method	SW range	Execution time in %	Bit-rate gain in %
TM-A	32	213	-5.30
TM-LLE	32	249	-5.89
ITM-LLE	16	207	-4.94

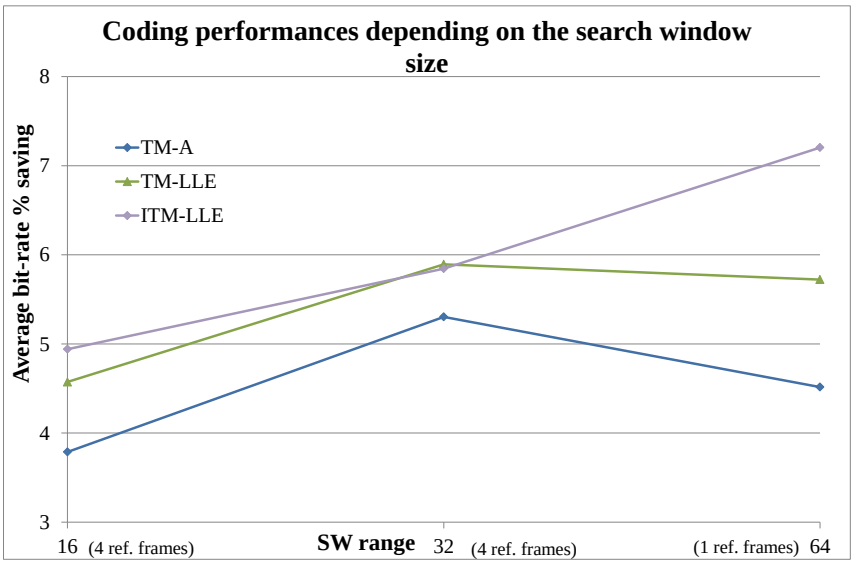


Figure 2.11: Coding performances obtained with the different proposed prediction methods as a function of the SW range and the number of reference frames.

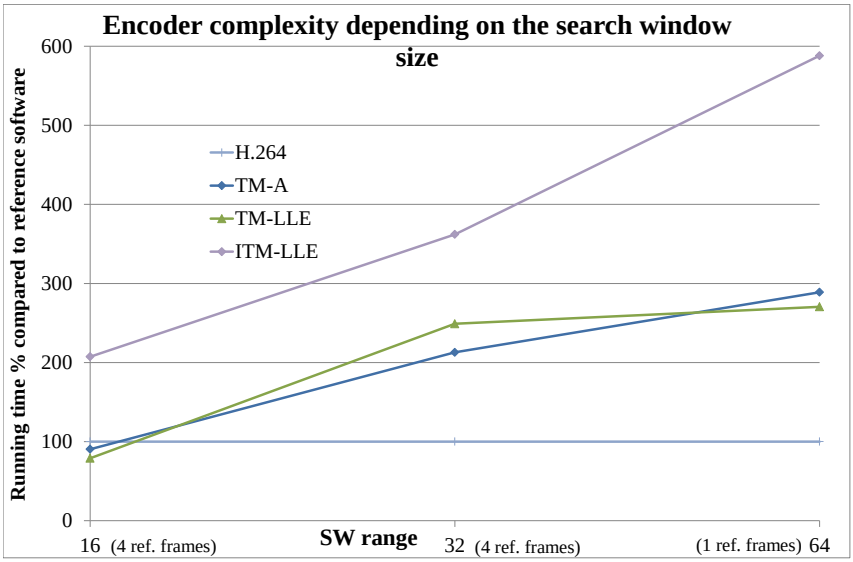


Figure 2.12: Encoder complexity obtained with the different proposed prediction methods as a function of the SW range and the number of reference frames.

2.3.4 Combining Intra and Inter LLE-based prediction methods

In this section, we focus on the integration of both Intra and Inter prediction based on LLE. As in [13], the LLE-based Intra prediction method uses both TM-LLE and oITM-LLE prediction techniques in competition, integrated in H.264 by replacing two of the eight directional modes. The replaced modes are the least statistically used. The statistics are obtained by first running the original H.264 software without the additional LLE-based prediction methods. The proposed Intra method is allowed for 8×8 and 4×4 partitions. The method is here denoted Intra TM/oITM-LLE. Note that this method is not only applied to I frames, but also to P and B frames. The method chosen for LLE-based Inter prediction is TM-LLE, since it allows a good trade-off between complexity and coding performances, and is here denoted Inter TM-LLE. First, the RD and complexity performances are assessed, followed by an in-depth analysis of the encoder behavior. For the Intra TM/oITM-LLE method, K is set to 32, the search range is set to 32, and L is set to 8. For the Inter TM-LLE method, K is set to 8, the search range is set to 32, with 1 reference frame.

Table 2.5 shows the coding performances achieved with the Inter TM-LLE method alone, the Intra TM/oITM-LLE method alone, and the combination of both methods, against the H.264 reference. Table 2.6 shows the corresponding encoder complexity. The results demonstrate that the LLE-based Inter and Intra prediction methods are complementary, especially when evaluating the coding gains of the proposed methods for each sequence separately. We can see that the combined Inter TM-LLE alone and Intra TM/oITM-LLE methods can achieve up to 15.31 % bit-rate saving.

Table 2.5: Coding gains (in %) for the Inter TM-LLE, Intra TM/oITM-LLE and combined methods for each sequence.

Sequence	Inter TM-LLE	Intra TM/oITM-LLE	Inter TM-LLE and Intra TM/oITM-LLE
Foreman	-2.73	-1.68	-4.40
Rushes	-5.15	-2.90	-7.27
Matrix	-2.43	-6.53	-6.47
City	-6.57	-3.60	-8.80
Spincalendar	-7.95	-6.60	-15.31
Average	-4.97	-4.26	-8.45

Table 2.6: Encoder complexity (in %) for the Inter TM-LLE, Intra TM/oITM-LLE and combined methods for each sequence.

Sequence	Inter TM-LLE	Intra TM/oITM-LLE	Inter TM-LLE and Intra TM/oITM-LLE
Foreman	101.34	154.81	210.73
Rushes	98.89	148.47	202.59
Matrix	89.85	187.80	171.14
City	94.38	119.13	233.52
Spincalendar	93.39	167.93	426.61
Average	95.57	155.63	248.92

Note that, even with the combined methods, the percentage of extra-information (flag indicating if the LLE-based method is used and the index for the oITM-LLE method) only amounts in average to two to three percents of the complete bit-rate.

Fig. 2.13 shows the coding performances, averaged over all sequences, of the combined methods as a function of the frame type. For I frames, only the Intra TM/oITM-LLE method is used, and the bit-rate saving reaches 5.92 %. For the P and B frames, the bit-rate saving reaches 7.20 % and 9.27 % respectively, which shows the gain brought by the LLE-based Inter prediction.

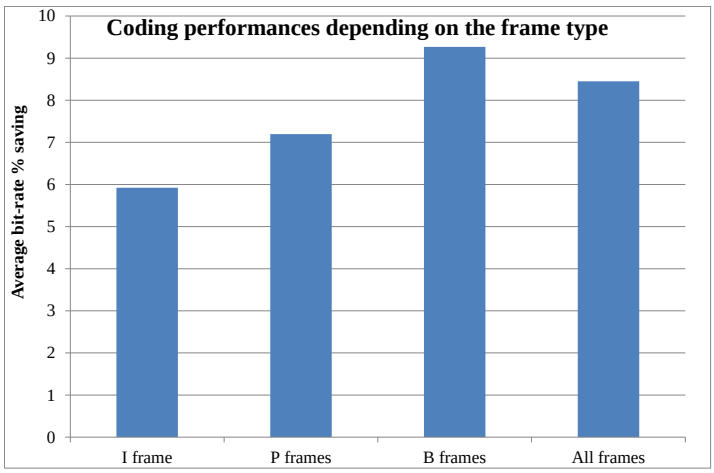


Figure 2.13: Coding performances of the combination of both Inter TM-LLE and Intra TM/oITM-LLE as a function of the frame type.

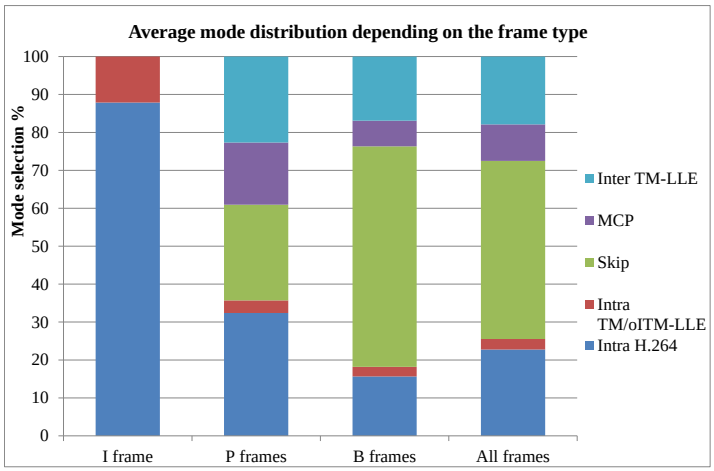


Figure 2.14: Distribution of the selected prediction modes as a function of the frame type.

Fig. 2.14 shows the mode distribution, averaged over all sequences, as a function of the frame type. For I frames, the Intra TM/oITM-LLE method only accounts for 12.11 % of the selected modes. Paradoxically, this little amount of selection shows the method efficiency, since it still brings a significant bit-rate reduction. For the P frames, the combined LLE-based methods account for 25.98 % of the selected modes, which is more than the MCP mode (16.39 %) and the Skip mode (25.25 %). For the B frames, the combined LLE-based methods account for 19.42 % of the selected modes, which is more than the MCP mode (6.79 %), but less than the Skip mode (58.11 %). Although

it is less selected than for the P frames, the combined LLE-based methods coding gains for the B frames are still significant (as shown in Fig. 2.13), since it is accumulated over more frames. Thus, we can see that the combined LLE-based methods amount for a significant part of the selected modes for the temporal frames.

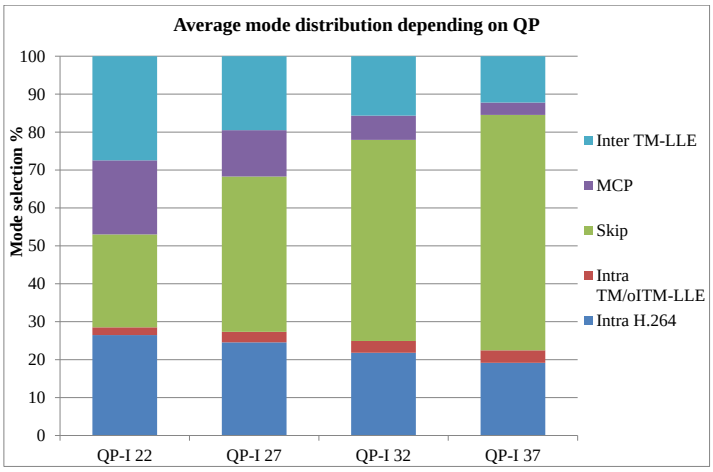


Figure 2.15: Distribution of the selected prediction modes as a function of the quantization parameter.

Fig. 2.15 shows the mode distribution for all frames, averaged over all sequences, as a function of the quantization parameter. The results show that the mode distribution strongly varies depending on the bit-rate. The Inter TM-LLE method is much more selected at high bit-rates (QP-I 22), while at low bit-rates (QP-I 37), the Skip mode is the most selected mode. As explained in section 2.3.1, the LLE-based methods are mainly effective for high-frequency pseudo-periodic textures, which are well preserved at high bit-rates, but on the contrary over-smoothed at low bit-rates, which tends to favor the skip mode. Moreover, we can see that the Inter TM-LLE is more selected than the classical MCP for all bit-rates, and in proportion much more selected than the MCP at low bit-rates. These results confirm the better efficiency of the proposed multi-patches method for reconstructing high-frequency textures compared to the single block MCP.

The City sequence is a good example to illustrate this kind of behavior. The RD curves of H.264 and the combined LLE-based methods for this sequence are displayed in Fig. 2.16, and clearly show that the RD performances of the combined LLE-based methods are better at high bit-rates.

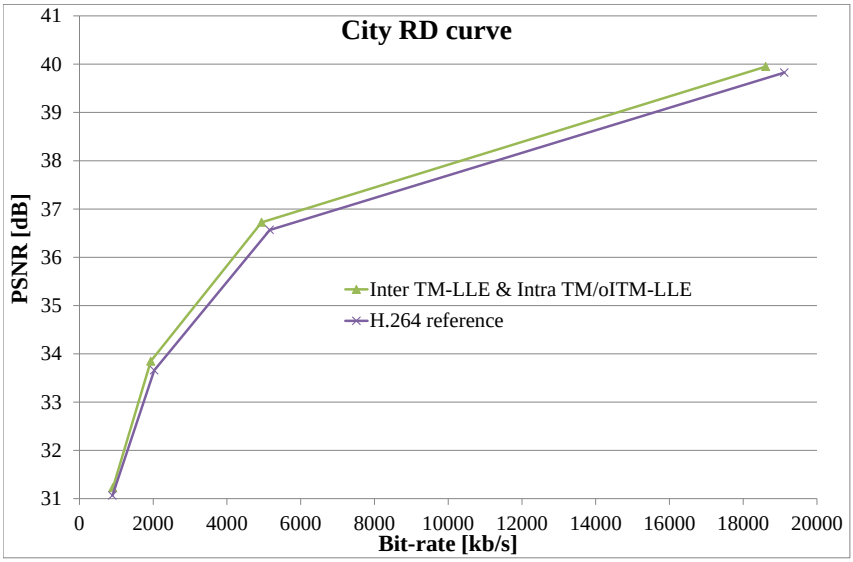


Figure 2.16: Rate-Distortion performances of the City sequence.

2.4 Conclusion and perspectives

In this chapter, we have introduced new Inter prediction methods for video compression based on LLE. The proposed methods rely on the same multi-patches combination with different K-NN search strategies which are aided or not by motion information. It is shown that the methods that are not aided by the motion information give the best coding performances. This shows that our approach is an interesting alternative mode, complementary to current video compression techniques. The different methods can significantly improve the coding efficiency compared to the H.264 reference software, and the best methods outperform the state-of-the-art TM-A.

Through different experiments, we showed that the proposed methods are not extremely sensitive to the key parameters in terms of RD performances. However, the tuning of these parameters can be used to reduce the complexity. The oITM-LLE method reaches the highest bit-rate reduction, which requires in return a high computation time. Nevertheless, when applying the oITM-LLE method to Intra prediction, and combined with a less complex method such as TM-LLE for Inter prediction, we showed that significant bit-rate saving can be obtained for a reasonable complexity cost.

Furthermore, the complexity essentially comes from the K-NN search for which efficient and fast methods exist [14][15], as well as hardware acceleration modules. In particular, matching methods based on hash functions have been recently introduced to perform efficient NN search [16][17], and have been used to improve HEVC for screen content coding. This process is also highly parallelizable and much reduced execution

times can therefore be expected, e.g. using GPU implementation [19]. The study instead focused on the assessment of the coding performances and not on the development of an optimized paralellized implementation. Note that such optimized implementation could also allow to refine the K-NN search to sub-pel level, so LLE-based methods could potentially be combined with up-sampling filters.

In addition, we believe that our results would still hold in HEVC. In fact, the prediction tools in HEVC follow the same principle as those used in H.264: directional propagation modes for Intra prediction and motion estimation/compensation for Inter prediction. Our results show that the proposed methods are complementary with the standard tools, and improve the RD performance. Template-based prediction methods were extensively studied in H.264, and recent work shows that they can be effectively used in HEVC [51][18]. We expect our methods to outperform these techniques, as they yield a better prediction quality. The proposed methods are meant to be in competition with the standard Intra and Inter prediction tools, which would require applying them to different PU sizes. In fact, the numerous PU sizes, and especially the larger ones, are known to be an effective tool of HEVC [35]. This adaptation is conceptually straightforward, and can even be extended to rectangular PUs. Furthermore, additional tools providing RD gain, such as adaptive transform sizes, de-blocking filter [35], are directly compatible with the proposed methods. The overhead in the bitstream corresponding to the flag indicating if the LLE-based method is used and the index for the oITM-LLE method is low for H.264, and we do not expect an increase for HEVC, since applying the proposed methods on larger PUs would reduce the number of flags or indexes to be transmitted.

Chapter 3

Epitome-based quantization noise removal

In chapter 2, we studied Inter prediction methods based on Locally Linear Embedding (LLE) in order to improve coding performances. Following the work presented in [12][55][13] for Intra prediction, we showed that the proposed LLE-based multi-patches methods were an efficient Inter prediction tool. The coding performances are especially improved when the proposed methods are combined for both Intra and Inter prediction, which corroborates the interest of an in-loop implementation of such methods.

Independently from the standard video compression techniques emerged the concept of epitome, first presented in [20][21], which is defined as a condensed representation of an image or video containing the essence of its textural properties. An epitome thus consists in a texture epitome, which we denote E, containing texture patches representative of the whole input signal, and a mapping ϕ which links the texture patches to their original position in the input image/video. Multiple application of the epitomes are presented in [20][21], including segmentation, image editing, de-noising, super-resolution and inpainting. The concept of epitome is in addition closely related to image compression, as it relies on the reduction of similarities within an image, and was used in [56] to improve H.264 Intra prediction. Moreover, a new epitome factorization model was presented in [22], which presents among several applications a direct approach for epitome-based image compression, which merely consists in compressing the texture epitome and the transform map. However, the work of [23] showed that the coding performances of such approach are limited, especially at high bit-rate. Based on the work of [22], an adapted epitome generation algorithm is proposed in [23] for a novel still image compression method, which consists in coding the texture and the transform map along with a reconstruction residue. The method was proven efficient compared to H.264 Intra coding. Furthermore, an improved approach is proposed in [57][1] which exploits the concept of epitome in combination with the efficient LLE-based multi-patches prediction methods. More precisely, the epitome is first transmitted to the decoder, then the non-epitome blocks are predicted in an inpainting manner from the epitome using LLE-based

[1] Co-authored publication

multi-patches methods, with an in-loop residue coding. Results showed that the method clearly outperforms H.264 Intra coding.

The work presented in this chapter can be seen as an extension of the work of [57] from still image to video compression. However, the approach chosen here is an out-of-the-loop scheme, where the epitome is transmitted to the decoder with a high quality in addition to the whole encoded sequence, and used at the decoder side to perform multi-patches based de-noising on the non-epitome blocks, instead of in-loop prediction. The proposed approach thus gains in genericity, as it can be applied to any existing coding scheme.

This chapter is organized as follow. Section 3.1 reviews background on epitomic models and de-noising. Section 3.2 presents a method for efficient epitome generation. Section 3.3 describes the proposed quantization noise removal out-of-the-loop scheme for video compression.

3.1 Background

3.1.1 Epitome generation

3.1.1.1 Generative models

N. Jojic and V. Cheung first introduced the notion of epitome in [20][21]. An epitome is defined as the condensed representation (meaning its size is only a fraction of the original size) of an image (or a video) signal containing the essence of the textural properties of this image.

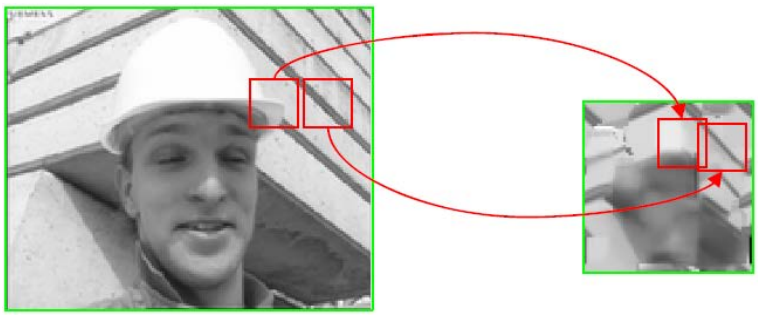


Figure 3.1: Epitome (right) generated from a frame of the Foreman sequence (left) and part of the mapping (in red). Source: [58].

Given an epitome E and the input image I, a mapping ϕ can be derived, which links patches from the epitome to their position in the original image (see Fig. 3.1). If the mapping and the epitome are given, the original image can thus be reconstructed. However, the main applications of this approach have been for image analysis such as segmentation, de-noising, recognition, indexing or texture synthesis. The epitome is computed from an image through Maximum Likelihood Estimation (MLE) using an

Expectation Maximization (EM) algorithm. The mapping ϕ is then a hidden variable in the computation process. This patch based probability model was shown to be of high "completeness" in [59] but introduces undesired visual artifacts, which is defined as a lack of "coherence". In fact, since the model is learned by compiling patches drawn from the input image, patches that were not in the input image can appear in the epitome (see Fig. 3.2). These artifacts can be a drawback for some applications, e.g. intra coding such as in [56][58][60], because they will limit the quality of the prediction. As this chapter mostly deals with epitome dedicated to image coding, this generative model approach will not be discussed in more details in the next sections.

This approach was also extended into a so-called Image-Signature-Dictionary (ISD) optimized for sparse representation [61]. The ISD is generated by replacing the probabilistic averaging of patches in the previous approach by their sparse representation. Thus is obtained an image (see Fig. 3.2) that can be used as a dictionary for sparse representations and has several important features such as shift and scale flexibility. For the same reasons as the previous method it will not be considered in the rest of this chapter.

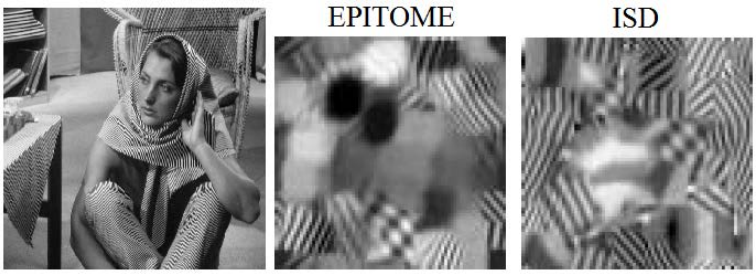


Figure 3.2: Epitome (middle) and ISD (right) generated from Barbara (left). Input image is 512x512, ISD and epitome are 75x75 (figure is not at scale). Source: [61].

3.1.1.2 Factoring similar content within an image

Wang et al. introduced in [62] a new approach to build epitome based on self-similarity tracking within the image. This approach has been designed with in mind (lossy) image reconstruction (and more generally texture mapping) rather than image analysis applications. Thus the epitome obtained by this method is composed of image patches that are not altered.

In this approach, the input image I is factored in an epitome E and a transform map ϕ (corresponding to the mapping in previous section). The input image is divided into a regular grid of blocks and each block will be reconstructed from a transformed epitome patch. The epitome itself is composed of disjoint texture pieces called "epitome chart". The transform map contains all the parameters that link the patches from the epitome to the input image blocks. Wang et al. proposed a so called epitome atlas which aim is to compact the different epitome charts to obtain a smaller image.

Ideally, one seek to minimize the size of the epitome and the transform map, $|E|+|\phi|$, as well as the image reconstruction error $||I' - I||^2$. Assuming that the block size is fixed, the transform map size $|\phi|$ is fixed as well. The problem can then be expressed as:

$$\min_{E,\phi} |E| \text{ such that } \forall B \in I, e(B) \leq \varepsilon \tag{3.1}$$

where $e(B)$ represents the reconstruction error of the current block B.

This minimization is solved using a greedy construction process that iteratively grows epitome charts copied from the input image. More precisely, the epitome construction procedure has the following steps:

- Find self-similarities for each block B_i in I.
- Create an epitome chart EC for each repeated content, to satisfy a maximum norm on the image reconstruction error (see Eq. 3.1).
- Optimize the transform map ϕ, to minimize the reconstruction error given the epitome content.
- Assemble all epitome charts EC into an epitome atlas (optional).

We describe below the first two steps of the procedure.

Finding self-similarities. The first step is performed using the Kanade-Lucas-Tomasi (KLT) feature tracker [63][64] that optimizes affine alignment of two windows. Thus for each block $B_i \in I$ a set of matched patches (also called matchings) $Match(B_i) = \{M_{i,0}, M_{i,1}, ...\}$ is obtained, which is found through the different transformations considered by the KLT algorithm: translation, rotation and scaling (see Fig. 3.3). In this approach mirror reflections were also considered. To be a viable matching the error between the current block B_i and the candidate patch must be inferior to a threshold ε_M.

Epitome chart extension. To extend an epitome chart EC one considers all the matchings overlapping a block B_j included in EC. For the initialization all the block of the image are considered and not only the blocks in the current epitome chart. The region ΔEC encompassing said matchings is considered as a candidate for the extension of EC (see Fig. 3.4). The actual extent is the candidate that can reconstruct the largest region. The epitome chart growth stops when the regions reconstructed by the extent candidates ΔEC are smaller than the extent candidate itself. The process stops when the entire image is reconstructed.

Once the epitome is obtained the quality of the reconstruction image can be improved by finding potential better matching in the epitome for each block.

At this point the epitome itself is still the same size as the input image, but is filled with "empty" pixels outside the epitome chart. The authors proposed a method

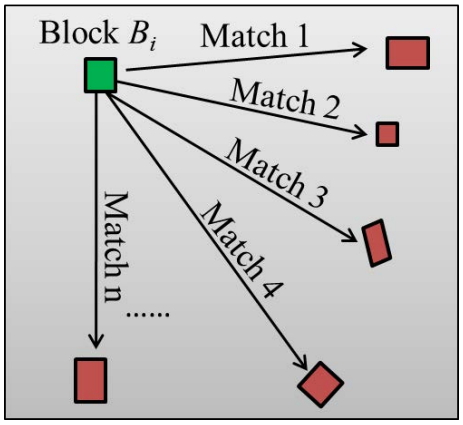


Figure 3.3: Find self-similarities within an image. Source: [62].

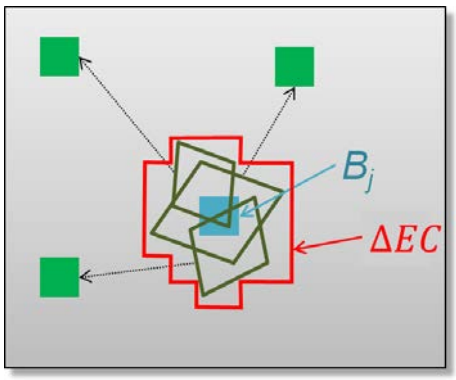


Figure 3.4: Candidate region ΔEC for epitome growth, formed as the union of matchings that overlap the block B_j. Source: [62].

to compact the representation in a so-called epitome atlas, which can reduce the size of the image (see Fig 3.5).

Figure 3.5: Example of factor image. In the center is the epitome obtained from the input image on the left. On the right is the epitome atlas obtained from the epitome. Source: [62].

3.1.1.3 Epitome dedicated to image coding

In this section, we introduce an approach first presented in [23]. The method is inspired by the previous one, but only relies on a translational model to find self-similarities, with in mind image coding applications.

Finding self-similarities. The first step of the epitome construction consists in searching, for each block in the input image, the set of patches in the image with similar content. That is, for each block B_i, belonging to the block-grid, we determine the list of matchings $ML(B_i) = \{M_{i,0}, M_{i,1}, ...\}$ that reconstructs B_i with a given error tolerance ε_M (see Fig. 3.6). The procedure of matching is performed with a block matching algorithm using an average Euclidian distance. Compared to previous method, the only transformation allowed is translation. Note that an exhaustive search is performed in the entire image. Once all the Match lists have been created for the set of image blocks in block-grid, a new list $RL(M_{j,k}) = \{B_j, B_l, ...\}$ will be built indicating the set of image blocks that could be represented by a matching block $M_{j,k}$. This list is used in the second step. Note that all the matching blocks $M_{j,k}$ found during the full search step belong to the pixel grid. Furthermore the matching search can be extended to a sub-pixel grid.

Epitome chart extension. Here, the second step that consists in building the epitome charts differs from the previous approach. Each chart EC is initialized by the matched patch which is the most representative in the input image and whose reconstructed image blocks are not represented yet by the epitome. Formally the initialization is based on a Mean Squared Error (MSE) criterion between the original image I and the reconstructed image I', where the non reconstructed pixels of I' are set to 0.

$$EC_{init} = \arg\min_{M}\left(\frac{\sum_i^W \sum_j^H ||I_{i,j} - I'_{i,j}||^2}{W * H}\right) \text{ for } M \in \mathbf{M}\backslash\mathrm{E} \tag{3.2}$$

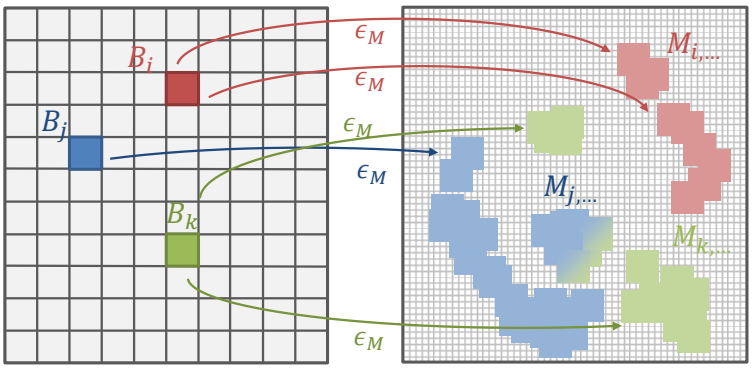


Figure 3.6: For each block B in the block-grid a list of matchings $ML(B)$ is found at pixel accuracy. Note that in this example $ML(B_j)$ and $ML(B_k)$ have elements in common.

where W and H are the width and the height of the image respectively, $\mathbf{M}$ is the set of all available matchings and E is the set of matchings that constitute the current epitome.

Once initialized the epitome chart EC is extended by considering a set of candidate extension regions $\Delta EC_m, m = 0, ..., N-1$ where N is the number of candidates. The candidates considered are the matchings overlapping with the current epitome chart. The actual extension ΔEC_{opt} should minimize the MSE between I and I'. Furthermore, in [23] is introduced an optimization taking into account so-called inferred blocks. The inferred blocks are the potential matchings that can overlap the current chart EC and the extension ΔEC_m (see Fig. 3.7) and thus help reconstruct new blocks in I'. The criterion also takes into account the extension size so that the epitome chart growth is limited. Formally the selection is conducting according to the minimization of the following Lagrangian criterion:

$$\Delta EC_{opt} = \underset{\Delta EC_m}{\arg\min}\left(\frac{\sum_i^W \sum_j^H ||I_{i,j} - I'_{i,j}||^2}{W * H} + \lambda \times \frac{|EC + \Delta EC_m|}{W * H}\right) \tag{3.3}$$

where $|EC+\Delta EC_m|$ represents the size of the epitome chart and its extension evaluated by the number of pixels. Note that the criterion of Eq. 3.2 is very similar but does not need the second term because all candidates have the same size. Finally the extension stops when no more matchings overlapping with the current epitome chart can be found.

The global process stops when the whole image is reconstructed.

Thus is obtained the epitome E and a so-called assignation map ϕ (corresponding to the previous transform map), which is actually a subset of the lists RL. The epitome charts in E are originally obtained at a pixel accuracy. In order to encode the epitome

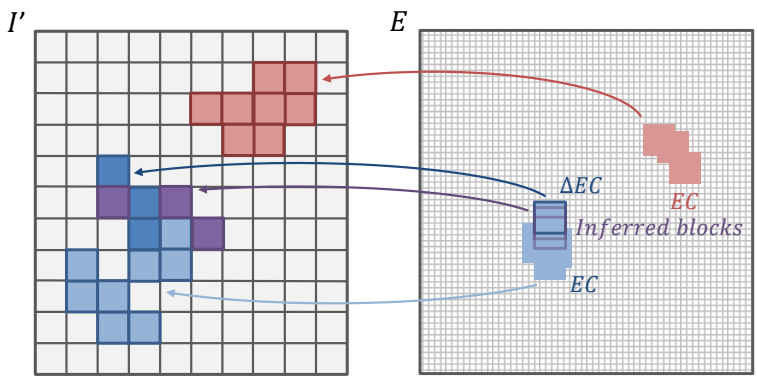


Figure 3.7: Epitome chart extension process. On the right, in red is presented a finished epitome chart, while in blue is presented the current epitome chart being extended. The inferred blocks are in purple.

efficiently, it has to be padded so it suits the block structure of the encoder used. Thus in [23], E was padded to a 8x8 block grid to be later encoded with a H.264 encoder. The padding process adds blocks and inferred blocks in the epitome that can then be used to improve the assignation map. In [23] were also conducted experiments where the matchings were found at sub-pixel accuracy, which lead to better coding performances at the price of an increased complexity.

In Fig. 3.8, we give an example of epitome obtained with this method on the Calendar image, with a threshold $\varepsilon = 10.0$.

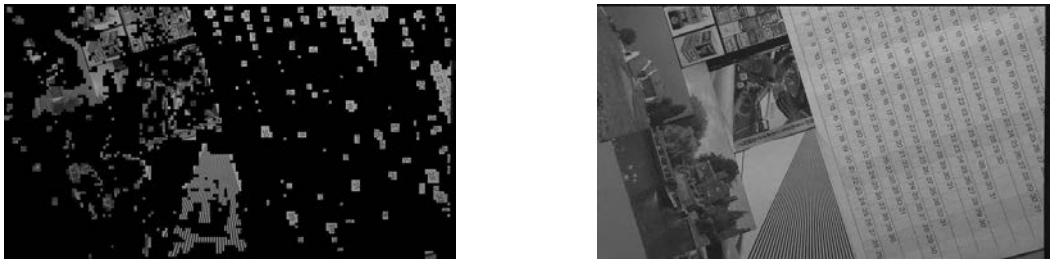


Figure 3.8: Epitome (left) extracted from Calendar and the corresponding reconstructed image (right). The epitome was obtained using the full search method for the self-similarities search with the threshold $\varepsilon = 10.0$.

3.1.2 De-noising

De-noising is the task of recovering an estimate of a source signal from a noisy version of this signal. Image de-noising has been a very active research topic, and we present

in this section several methods among the most popular and best performing. These methods assume that the source image is corrupted with additive white Gaussian noise (AWGN). We note Y the noisy version of the source image X. For all the pixels y_i and x_i in Y and X respectively, we have the following relation:

$$y_i = x_i + n_i \tag{3.4}$$

where n_i is a zero-mean Gaussian variables with variances σ_n^2.

In the remaining of this section, we will note y and x patches extracted from the noisy image and the source image respectively.

More sophisticated noise models than AWGN have been proposed, such as the signal-dependent noise model [65], for which the pixels y_i and x_i have the following relation:

$$y_i = x_i + x_i^{\gamma} \cdot u_i + w_i \tag{3.5}$$

where γ is the exponential parameter which controls the dependence on the signal, and u and w are zero-mean Gaussian variables with variances σ_u^2 and σ_w^2 respectively.

However, the main approaches to remove such noise rely on the assumption that the noisy image can be partitioned into segments or clusters for which the noise can be modeled by AWGN, such as in [66][67]. The approach thus consists in segmenting/clustering the noisy image, estimate the noise level in each segment/cluster, and finally apply a known de-noising method for AWGN on each cluster. Thus, in the next sections, we only present methods assuming AWGN.

We note that many recent successful de-noising methods relying on sparse representations have been proposed [68][69][70]. Efficient techniques relying on the local learning of dictionaries through clustering were also proposed [71][72]. However, it was shown that these methods are usually outperformed by the BM3D algorithm [73]. Thus, we choose not to describe these methods in details, and focus instead on the BM3D algorithm, which is later use for comparison in the contributions of this chapter.

3.1.2.1 Optimal Wiener filter

The optimal Wiener filter for image de-noising proceeds by shrinking coefficients in a transform domain. We note T the transform, e.g. a discrete Fourier transform (DFT), a discrete cosinus transform (DCT) or a discrete wavelet transform (DWT).

The Wiener filter coefficients in the transform domain are expressed as follow:

$$W = \frac{|T(x)|^2}{|T(x)|^2 + \sigma_n^2} \tag{3.6}$$

The de-noised estimate $\hat{x}$ of the patch x is obtained from the noisy patch y by applying the filter to the transform coefficients of y and inverse transform the result:

$$\hat{x} = T^{-1}(W \cdot T(y)) \tag{3.7}$$

where $\cdot$ represents the element-by-element multiplication.

The Wiener filter is denoted optimal because its computation relies on the knowledge of the source signal. In practice, the filter can be used by computing a first de-noised estimate of the image, and using as "oracle" in Eq. 3.6, such as in [24] or in the BM3D algorithm described below.

3.1.2.2 Transform domain thresholding

The idea of thresholding coefficients in a transform domain was introduced in [25], and has been proved to be very efficient since. The method is conceptually similar to the Wiener filter, but instead of applying the shrinkage coefficients, a thresholding operator (which does require any knowledge on the source signal) is applied. In [24], the method is used to obtain a first de-noised estimate which is then used as oracle for a Wiener filter.

If we note τ the thresholding operator, the de-noised estimate $\hat{x}$ of the patch x is obtained from the noisy patch y as:

$$\hat{x} = T^{-1}(\tau(T(y))) \tag{3.8}$$

Two types of operators τ can be used, the soft thresholding as in [25][26] or the hard thresholding as in [24]. We give below the definition of soft and hard thresholding of a coefficient c with threshold $\lambda > 0$:

$$\tau_{soft}(c, \lambda) = \begin{cases} 0, & \text{if } |c| < \lambda \\ c - \text{sign}(c)\lambda, & \text{otherwise.} \end{cases} \tag{3.9}$$

$$\tau_{hard}(c, \lambda) = \begin{cases} 0, & \text{if } |c| < \lambda \\ c, & \text{otherwise.} \end{cases} \tag{3.10}$$

The threshold λ is usually determined depending on the noise level σ_n, as the rationale behind thresholding in the transform domain is to retain coefficients with a high signal-to-noise ratio.

3.1.2.3 Non Local Mean

Although the previous methods performing in a transform domain are very efficient, another type of methods chose to perform de-noising directly in the pixel domain, such as the bilateral filtering [74]. This method directly estimates a pixel as weighted average of its neighboring pixels. Although a photometric distance is taken into account to compute the weights, this method often results in over smooth estimates.

The Non Local Mean (NLM) algorithm [27][28] relies on a similar idea, but instead searches for neighbors of the patch surrounding the pixel to be de-noised in the Euclidean space, and combines the central pixels of the neighbor patches depending on the Euclidean distance between the current patch and its neighbor. The neighbor patches can be spatially located anywhere in the image, hence the "non local" property of the method.

Formally, we note y^c the central pixel of the noisy patch y of size $N \times N$. We note K the number of neighbors, and $y_i, i = 0 \dots K-1$ the K nearest neighbors (K-NN) of y. The distance between y and its neighbor y_i is noted $d_i = \sqrt{\frac{||y-y_i||_2^2}{N^2}}$. Note that y_0 is the patch y itself, as it is its closest neighbor. Originally, all the patches available in the image were considered as neighbors. To limit the complexity, only the patches in a search window around the current patch were then considered. Alternatively, a fixed number K defined by the user can be used.

The de-noised pixel $\hat{x}^c$ is obtained as follow:

$$\hat{x}^c = \frac{\sum_{i=0}^{K-1} w_i * y_i^c}{\sum_{i=0}^{K-1} w_i} \tag{3.11}$$

The averaging weights are computed as:

$$w_i = e^{-\frac{d_i^2}{2\sigma_{NLM}^2}} \tag{3.12}$$

where the parameter σ_{NLM} act as a degree of filtering, and is set in the original NLM algorithm to $\sigma_{NLM} = 10 * \sigma_n$.

A patch-wise version of the method can be derived from the pixel-wise NLM by directly combining the K-NN patches:

$$\hat{x} = \frac{\sum_{i=0}^{K-1} w_i * y_i}{\sum_{i=0}^{K-1} w_i} \tag{3.13}$$

The averaging weights are computed exactly as before. In this case, several estimates are obtained for each pixel, which are averaged to obtained the final de-noised version of the pixel.

3.1.2.4 BM3D

The block-matching 3D (BM3D) algorithm was presented in [29], and is still to this day one of the best performing methods in image de-noising. The method advantageously combines the strengths of the non local approaches and transform domain de-noising, by stacking a patch and its K-NN in a 3D group, and then applying a 3D transform on the 3D group of patches.

In a first step, the 3D transformed group is processed using hard thresholding. In a second step, the first de-noised image estimate obtained at the previous step is used to compute new K-NN and the corresponding 3D transformed group is processed using Wiener filtering. In the Wiener filtering step, the 3D transformed group of the first estimate patches is used as oracle to filter the 3D transform group of the corresponding noisy patches.

The 3D transform not only increases the sparsity of the transform compared to a 2D transform (and thus the efficiency of the transform domain processing), but also allows a collaborative filtering, *i.e.* all the patches of a 3D group are de-noised at once. Several estimates are thus obtained for a pixel, which are finally aggregated to obtain the final estimate.

The two steps of the method are described in detail below. Many notations are needed to formally describe the BM3D algorithm, which are summarized in Table 3.1 for clarity. The method processes overlapping patches of size $N \times N$ with a step of s pixels in both horizontal and vertical directions. For each patch, the K-NN search is limited to a search window of size $N_{SW} \times N_{SW}$. In the original algorithm, the distance between the current patch y and one of its retained neighbor y_i must not be higher than a pre-defined maximum acceptable distance ε_{BM}:

$$d_i = \frac{||y - y_i||_2^2}{N^2} \leq \varepsilon_{BM} \tag{3.14}$$

Thus, K refers to the maximum number of neighbors. To keep the notations tractable, we still using in the description below the term K-NN, although the actual number of neighbors could be inferior to K. Note that these parameters (denoted common parameters in Table 3.1) may vary between the two steps in practice, but we keep below the same notations for clarity purpose.

Table 3.1: Notations used for the BM3D

Common parameters	
$N \times N$	Size of the patches
s	Sliding step between the overlapping patches
$N_{SW} \times N_{SW}$	Size of the search window for the K-NN search
K	Maximum number of neighbors
ε_{BM}	Maximum acceptable distance between the current patch an its neighbors
First step: Hard Thresholding	
T_{HT}	3D transform used for the Hard Thresholding step
λ	threshold parameter for the hard thresholding
$\mathbf{G}_y^{HT}$	3D group containing the K-NN of the current patch to be de-noised y
$\mathbf{G}_{\hat{x}}^{HT}$	3D group containing the de-noised estimates of the patches in $\mathbf{G}_y^{HT}$
Second step: Wiener filtering *This step is performed on the de-noised frames obtained at the previous step.*	
T_{Wien}	3D transform used for the Wiener filtering step
$\mathbf{G}_{\hat{x}}$	3D group containing the K-NN of the first estimate of current patch $\hat{x}$
$\mathbf{G}_y^{Wien}$	3D group containing the noisy patches corresponding to the patches in $\mathbf{G}_x^{Wien}$
$\mathbf{G}_{\hat{x}}^{Wien}$	3D group containing the de-noised estimates of the patches in $\mathbf{G}_y^{Wien}$

Hard Thresholding In this first step, the K-NN $y_i, i = 0 \ldots K-1$ of the patch y are first found. Note that, as for the NLM, the patch y_0 is the patch y itself. The patches are stacked in a 3D group noted $\mathbf{G}_y^{HT}$. A 3D transform T_{HT} is applied on this 3D group, which is then processed using hard thresholding. The de-noised group noted $\mathbf{G}_{\hat{x}}^{HT}$ is obtained after applying the inverse 3D transform:

$$\mathbf{G}_{\hat{x}}^{HT} = T_{HT}^{-1}(\tau(T_{HT}(\mathbf{G}_y^{HT}), \lambda)) \tag{3.15}$$

where λ is the threshold parameter, usually determined depending on the noise level. Note that all the patches of the group are thus de-noised at once.

At the end of this step, several estimates can be thus obtained for a pixel, which are then aggregated to obtain the final estimate.

The hard thresholding step is represented in Fig. 3.9.

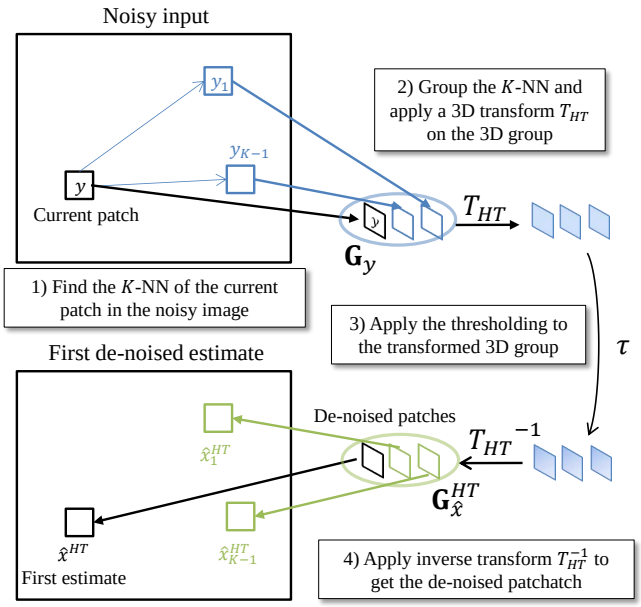


Figure 3.9: First step of BM3D: hard thresholding.

Wiener filtering This second step takes advantage of the first de-noised image estimate obtained at the previous step to perform a new K-NN search, which is expected to be more accurate. The K-NN of the first estimate of the current patch $\hat{x}$, noted $\hat{x}_i, i = \ldots K-1$ with $\hat{x} = \hat{x}_0$, are stacked in a 3D group noted $\mathbf{G}_{\hat{x}}$. The corresponding patches in the input noisy image $y_i, i = \ldots K-1$ are stacked in a second 3D group noted $\mathbf{G}_y^{Wien}$. This noisy patches group can then be processed by a Wiener filter, using $\mathbf{G}_{\hat{x}}$ as an oracle. The Wiener filter is computed as:

$$W = \frac{|T_{Wien}(\mathbf{G}_{\hat{x}})|^2}{|T_{Wien}(\mathbf{G}_{\hat{x}})|^2 + \sigma_n^2} \tag{3.16}$$

The group of de-noised patches can then be obtained as follow:

$$\mathbf{G}_{\hat{x}}^{Wien} = T_{Wien}^{-1}(W \cdot T_{Wien}(\mathbf{G}_y^{Wien})) \tag{3.17}$$

As in the previous step, several estimates can be obtained can be obtained for a pixel, and are aggregated to obtain the final estimate.

The Wiener filtering step is represented in Fig. 3.10.

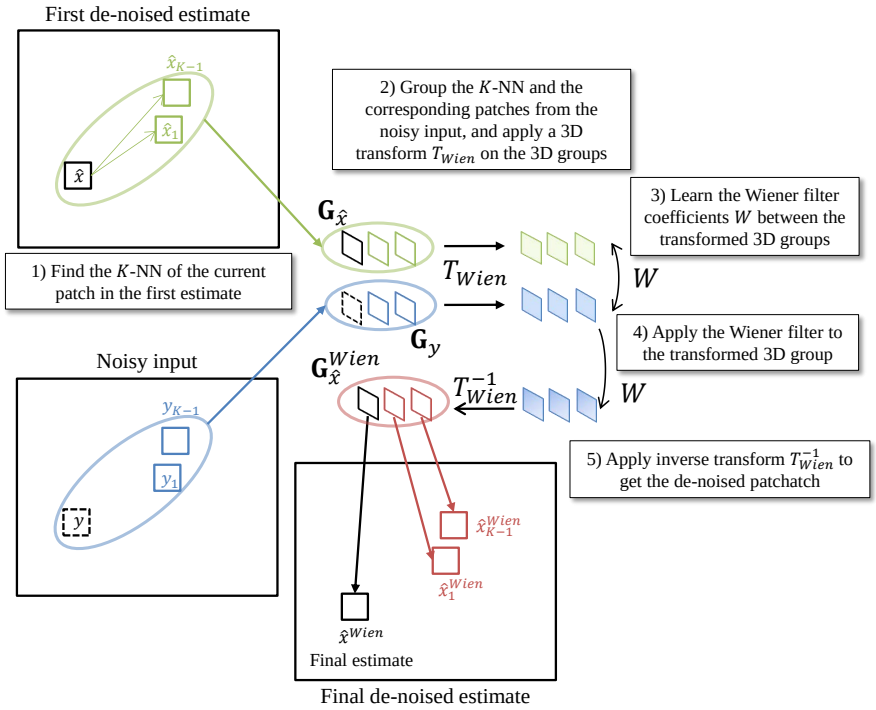


Figure 3.10: Second step of BM3D: Wiener filtering.

Several extensions of the BM3D have been proposed by its authors, e.g. color images de-noising [75], de-blurring [76], or even super-resolution [77]. Applications also include a Video BM3D (VBM3D) algorithm [78], in which the K-nearest neighbors of a patch are also searched in the adjacent frames. More precisely, the neighbors are searched in the current frames, N_{FR} past frames, and N_{FR} future frames.

3.2 Clustering-based methods for fast epitome generation

In this section, we propose a novel method for fast epitome generation. We build on the approach described in section 3.1.1.3 and propose to improve the self-similarities search step.

In fact, the self-similarities search method described in section 3.1.1.3 ensures that the best matches are found since an exhaustive search is performed. However, this method is known to be expensive in terms of memory consumption and processing time, since each block in the block-grid will be associated to a list that can contain a high number of matches. Different methods have been proposed to reduce the complexity of such nearest neighbor (NN) search. Video codecs such H.264 or HEVC usually integrate approximate BM algorithm. Classical approach to accelerate a NN search is to consider a hierarchical search, where the search is first conducted in sub-sampled version of the input image and the result is used to initialize the next level search. Faster approximate method for NN search based on k-dimensional tree (kd-trees) have been proposed [14]. State of the art concerning approximate NN (ANN) search optimization have been proposed in [79] and generalized for K-NN search in [15]. However all these methods are designed to find only a few best NN, *i.e.* use a relatively small and fixed value for K. The problem we address here is different because the number of NN, K, is not determined in advance and can reach really high values (up to several thousands, depending on ε_M).

We propose to replace the exhaustive match search by an approximate approach that takes advantage of the self-similarities within the blocks in the block-grid. In a first step, "sufficiently similar" blocks are grouped together. In a second step a match list is computed for each group, with respect to a representative block from the group. Note that this self-similarities search method is still compatible with the epitome generation step described in section 3.1.1.3, but here blocks in a same group will use the same match list. This not only accelerates the process since less match lists are computed but also saves memory space.

The similarity between the blocks is assessed using the average sum of absolute difference (SAD). We define a tolerance error ε_A that will be used to assign blocks to groups. The assignation threshold ε_A should be smaller than the matching threshold ε_M. Thus we define ε_A *via* a coefficient α_A such as :

$$\varepsilon_A = \alpha_A * \varepsilon_M, 0 \leq \alpha_A < 1 \tag{3.18}$$

Two grouping methods are presented below : first a novel list-based method, second a threshold-based clustering method adapted from [80]. We are here interested in a clustering method working without any prior information on the cluster number, which makes the use of classical methods such as K-means algorithm uneasy.

Note that the grouping is applied only for blocks in the block-grid.

3.2.1 List-based method for self-similarities search

This method was designed to obtain a simple grouping of similar non-overlapping blocks and follows the steps below:

- For each block B_i, find all blocks whose distance is below the assignation threshold ε_A. The association of B_i and such blocks form a potential list $PL(B_i)$.

- Find the potential list PL_{opt} with the highest cardinal. This potential list is set as an actual list AL. Blocks in AL are removed from the other potential lists they may belong to. If the block B used to compute a potential list $PL(B)$ is removed from this list, this potential list no longer exists and is simply not considered in future iterations.

- The previous step is iterated until all blocks belong to an actual list.

Finally for each actual list $AL(B_i)$ we determine a match list $ML(B_i) = \{M_{i,0}, M_{i,1}, \ldots\}$ obtained through an exhaustive search in the pixel-grid. The list is computed with respect to B_i but is then used by all blocks in $AL(B_i)$ for the epitome generation step. To satisfy the constraint that all the blocks have a reconstruction error inferior to ε_M, the blocks different from B_i in the list $AL(B_i)$ only use a subset of $ML(B_i)$ defined as:

$$\{M \in ML(B_i) | d(B_i, M) \leq \varepsilon_M - \varepsilon_A\} \tag{3.19}$$

This solution avoids computing the distance between blocks and all elements of a match list, which can be time consuming.

On one hand, this method tends to favor the creation of a few actual lists with large sizes, which is interesting since less match lists are computed. On the other hand, the blocks B_i used to compute the match lists are not necessarily the most representative blocks of the actual lists, which can limit the matches quality for the blocks in the actual list different from B_i. The method described in the subsection 3.2.2 addresses this issue.

In the example of Fig. 3.11, the size of $AL(B_j)$ is superior to the one of $AL(B_i)$ or $AL(B_k)$, so $B^0 = B_j$, and B_j and its associated blocks form the cluster C^0. Here $A_{j,4} = A_{k,0}$, but since it belongs to C^0, it is removed from the list $AL(B_k)$. The next cluster C^1 is thus formed by B_i and its associated blocks since it has now more associated blocks than B_k.

At the end of the process each block B in the block-grid is linked to a list of matchings ML that can be used to build an epitome, e.g. using the method described in section 3.1.1.3. Contrary to the approach presented in section 3.1.1.3, we do not need to actually compute the list of matchings for each block B, but only for a subset of blocks corresponding to the different clusters. This process thus reduces computing time and memory occupation.

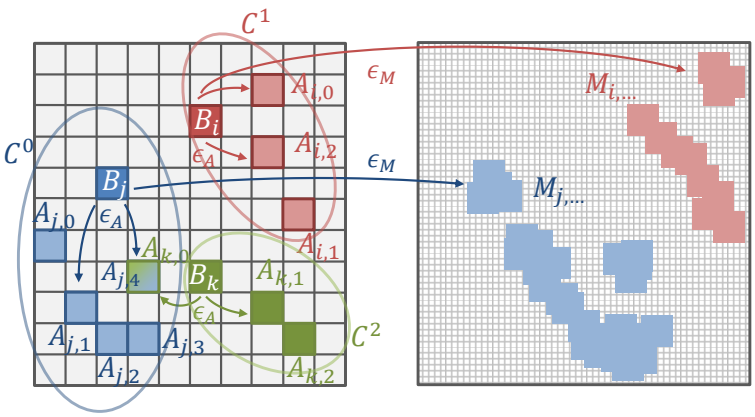


Figure 3.11: Example of self-similarities search using the associated lists.

3.2.2 Threshold-based clustering for self-similarities search

This method is derived from the threshold-based clustering algorithm presented in [80], and proceeds as follow:

- A block B_0 is randomly selected and used to initialize the first cluster C_0.
- For every block B not assigned to a cluster, compute its distance to the centroid of all existing clusters. If all distances are higher than the assignation threshold ε_A, initialize a new cluster with B as a seed. Otherwise assign B to the closest cluster and recompute the centroid of this cluster as the average of all blocks in the cluster.
- As a result, all blocks are assigned to a cluster. For every cluster, recompute the distances between the cluster blocks and the centroid. If a block B is found to have a distance to the centroid superior to ε_A, it is considered as a singularity and remove from the cluster. B is then used as a seed to initialize a new cluster.

We thus obtain clusters whose blocks are consistent with each other, and thus for each cluster C_i a match list $ML(B_i)$ is computed through an exhaustive search in the pixel-grid, with respect to the block B_i closest to the centroid. We choose not to use the centroid itself, as it is computed as the average of all blocks in the cluster, and as result can contain artifacts not suited for the match search. Thus except for the block B_i, all blocks in C_i will use approximate matches. As for the previous grouping method, we want to ensure that all blocks have a reconstruction error inferior to ε_M. The blocks different from B_i thus only use a subset of $ML(B_i)$, defined as in Eq. 3.19.

Contrary to the method described in subsection 3.2.1, this method does not favor groups of large sizes and thus may produce more groups. However, the blocks B_i are better representative for their groups, which improves the quality of the matches.

3.2.3 Trade-off between complexity and quality

The trade-off between the complexity reduction and the matching approximations is set using the parameter α_A. When $\alpha_A = 0$ the self-similarities search is performed for each block as in section 3.1.1.3 and the complexity is not reduced. When $\alpha_A \to 1$, on one hand groups of higher size are built and therefore we achieve higher complexity reduction, but on the other hand the approximation between the group blocks and the matches can lead to lower epitome quality. Furthermore the subset of matches used for the group blocks defined in Eq. 3.19 can be really small since $\varepsilon_A \to \varepsilon_M$, which can degrade the efficiency of the epitome generation step. In practice a good trade-off is obtained when $\alpha_A = 0.5$ (see section 3.2.4).

3.2.4 Simulations and results

Experiments were conducted on a set of 4 images : a frame extracted from the Foreman sequence (CIF), Lena (512×512), City (1280×720) and Calendar (1280×720). The size of the blocks is set to 8×8, and the epitome is padded with blocks of the same size. The epitomes were computed on a processor Intel core i7 @2.1 GHz.

First tests were carried out with $\varepsilon_M = \{3.0, 5.0, 10.0, 15.0\}$. The parameter α_A was here set to 0.5. Results are displayed in Table 3.2. Best results between list-based or cluster-based methods are displayed in bold font. The complexity reduction is assessed by the percentage of the optimized memory occupation over the original one, and the optimized method processing time over the original one. (Note that the maximum absolute processing time for the original method to generate an epitome ranges from a few seconds for CIF resolution, to several ten minutes for SD resolution). Two processing times are displayed : the self-similarities search time, which is the algorithm step actually optimized, and the complete epitome generation time. For all images, the complexity decreases when ε_M increases, because higher approximations are allowed. Thus the more interesting results are obtained when $\varepsilon_M = 10.0$ or $\varepsilon_M = 15.0$. The cluster-based method is overall faster than the list-based method for the self-similarities search, but is overall slower for the complete epitome generation. On average, the memory occupation reduction is better with the cluster-based method, but the lowest memory occupation is achieved with the list-based method. The memory occupation is prohibitive for images City and Calendar when $\varepsilon_M = 15.00$ with the original method. This shows a very important limitation of the full search method for high resolution images. Note that because of this, comparative results can not be displayed, but the epitome can be still generated when using optimized methods.

The quality of the epitome produced is assessed by the reconstructed image quality. The graphs representing the reconstructed image PSNR as a function of the epitome size are displayed in Fig. 3.13. The epitome size approximately ranges from 10% to 80% of the input image size, and is inversely proportional to the matching threshold ε_M. The reconstruction PSNR approximately ranges from 30 dB to 45 dB. For all images, whatever the epitome generation method, the curves are almost identical. The two methods presented in this section can thus reduce complexity while keeping the

Table 3.2: Memory occupation and computation time % with respect to original method, depending on ε_M, with $\alpha_A = 0.5$.

I	ε_M	List-based method			Cluster-based method		
		Memory load (%)	Search time (%)	Total time (%)	Memory load (%)	Search time (%)	Total time (%)
Foreman	3.0	50.18	68.62	73.93	**47.85**	**67.31**	**70.15**
	5.0	49.54	60.72	**60.12**	**45.22**	**55.80**	61.88
	10.0	25.00	32.65	**41.39**	**19.06**	**28.58**	50.36
	15.0	18.08	21.61	**42.37**	**14.59**	**18.00**	55.25
Lena	3.0	98.18	75.54	78.92	**96.81**	**69.05**	**70.13**
	5.0	63.61	61.00	**65.24**	**48.28**	**56.12**	66.19
	10.0	29.98	37.49	**51.92**	**23.66**	**31.60**	55.74
	15.0	**19.96**	**23.09**	**59.69**	21.64	23.56	64.48
City	3.0	60.54	**66.75**	**65.27**	**55.85**	68.62	69.25
	5.0	60.71	64.165	64.160	**56.47**	**62.48**	**62.80**
	10.0	51.47	48.07	60.07	**48.18**	**47.06**	**59.75**
Calendar	3.0	84.66	**68.36**	**66.44**	**80.69**	70.36	68.02
	5.0	48.43	54.76	**58.83**	**42.08**	**53.25**	59.94
	10.0	**28.33**	**32.58**	**52.97**	53.98	42.43	57.91

same epitome quality.

Visual results are given in Fig 3.12 for the Calendar image with threshold $\varepsilon_M = 10.0$ for the full search, list-based method and threshold-based clustering respectively. For the last two methods, we set $\alpha_A = 0.5$. We can see that the epitomes are different, but their sizes and corresponding reconstructed images are overall similar. Note the exhaustive visual results for the different test images, self-similarities search methods and thresholds are given in annex, section B.1.

Despite some disparities in the complexity results presented in Table 3.2, the performances between the two methods remain overall close. Complexity results for different values of α_A are displayed in Table 3.3, with $\varepsilon_M = 10$. As expected, the complexity decreases as α_A increases. For $\alpha_A = 0.25$ and $\alpha_A = 0.5$, performances of the two methods remain similar, despite some disparities. However when $\alpha_A = 0.75$, the complexity reduction is more important for the list-based method. This illustrates the behavior of this method, which can reduce drastically the complexity when allowing important approximations. However, when evaluating average reconstruction performances (see Fig. 3.14), we can see that it has a negative impact, as it is the only point which increases the epitome size while decreasing the reconstruction PSNR. Note that, as for the previous experiment, the reconstruction performances are very similar for all images, but for clarity reasons we only show the average results. For the cluster-based method, the complexity gain when $\alpha_A = 0.75$ is limited compared to $\alpha_A = 0.5$, and the reconstruction PSNR seems low compared to the epitome size, even though it is not as evident as for the previous method. Therefore $\alpha_A = 0.5$ seems to be, for both methods, a good trade-off value between complexity and epitome quality.

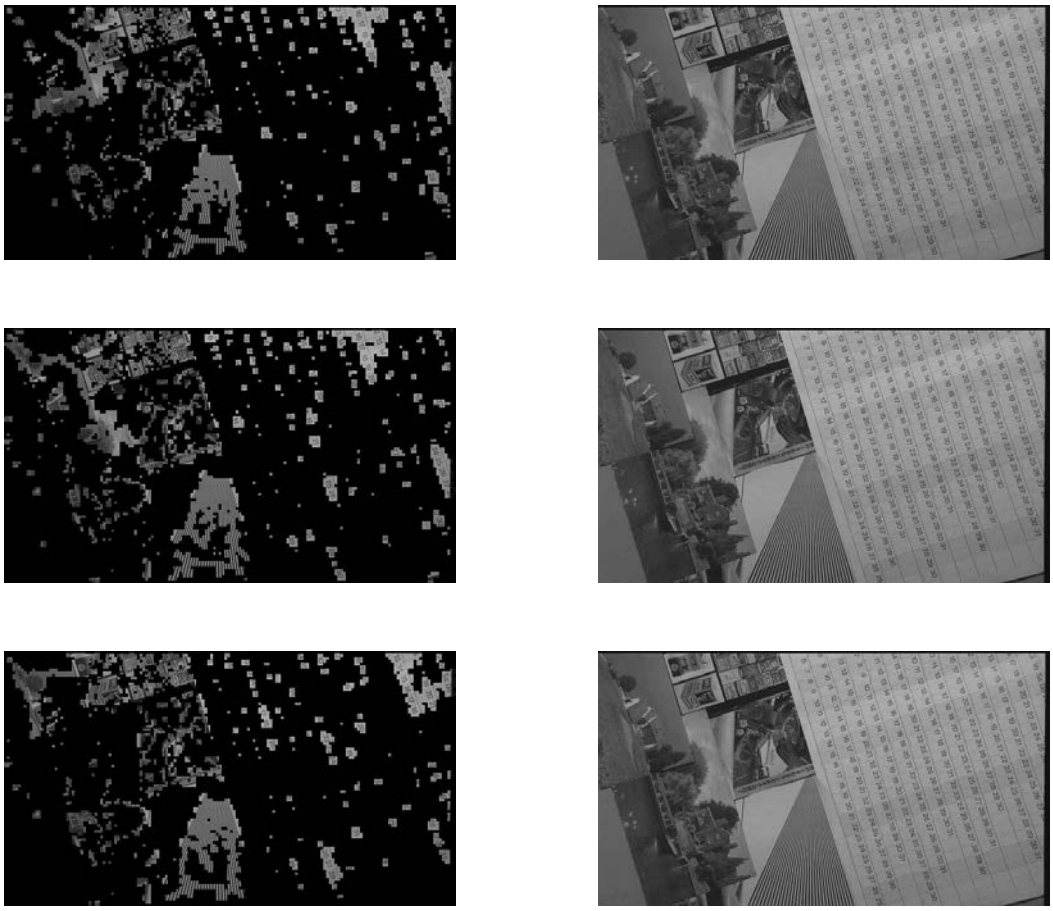


Figure 3.12: Epitomes (left) extracted from Calendar and the corresponding reconstructed images (right). The epitome was obtained using the full search method (top), the list-based method (middle), and the threshold-based clustering (bottom) for the self-similarities search with the threshold $\varepsilon_M = 10.0$ and the parameter $\alpha_A = 0.5$.

Table 3.3: Memory occupation and computation time % with respect to original method, depending on α_A, with $\varepsilon_M = 10.0$.

I	α_A	List-based method			Cluster-based method		
		Memory load (%)	Search time (%)	Total time (%)	Memory load (%)	Search time (%)	Total time (%)
Foreman	0.25	70.66	53.17	61.51	**67.86**	**51.42**	**58.41**
	0.50	25.00	32.65	**41.39**	**19.06**	**28.58**	50.36
	0.75	9.58	21.94	**24.74**	**7.32**	**19.96**	46.74
Lena	0.25	79.99	60.93	72.73	**70.13**	**56.41**	**65.69**
	0.50	29.98	37.49	**51.92**	**23.66**	**31.60**	55.74
	0.75	**9.03**	**23.27**	**26.57**	15.91	25.48	53.73
City	0.25	86.30	64.06	67.54	**84.09**	**63.30**	**66.99**
	0.50	51.47	48.07	60.07	**48.18**	**47.06**	**59.75**
	0.75	**19.46**	**33.84**	**42.41**	38.17	40.90	58.83
Calendar	0.25	69.76	51.43	65.30	**65.85**	**49.95**	66.65
	0.50	**28.33**	**32.58**	**52.97**	53.98	42.43	57.91
	0.75	**10.13**	**22.69**	**29.61**	48.35	36.86	57.94

3.2.5 Conclusion

This section presents efficient algorithms for epitome generation, based on list or cluster methods. The grouping of non-overlapping blocks limits the number of subsequent exhaustive searches over all overlapping blocks, and thus reduces the memory occupation as well as the processing time. Experiments show that interesting complexity results can be obtained without degrading the epitome quality.

In the next section, the proposed methods are implemented in an out-of-the-loop de-noising scheme for video compression.

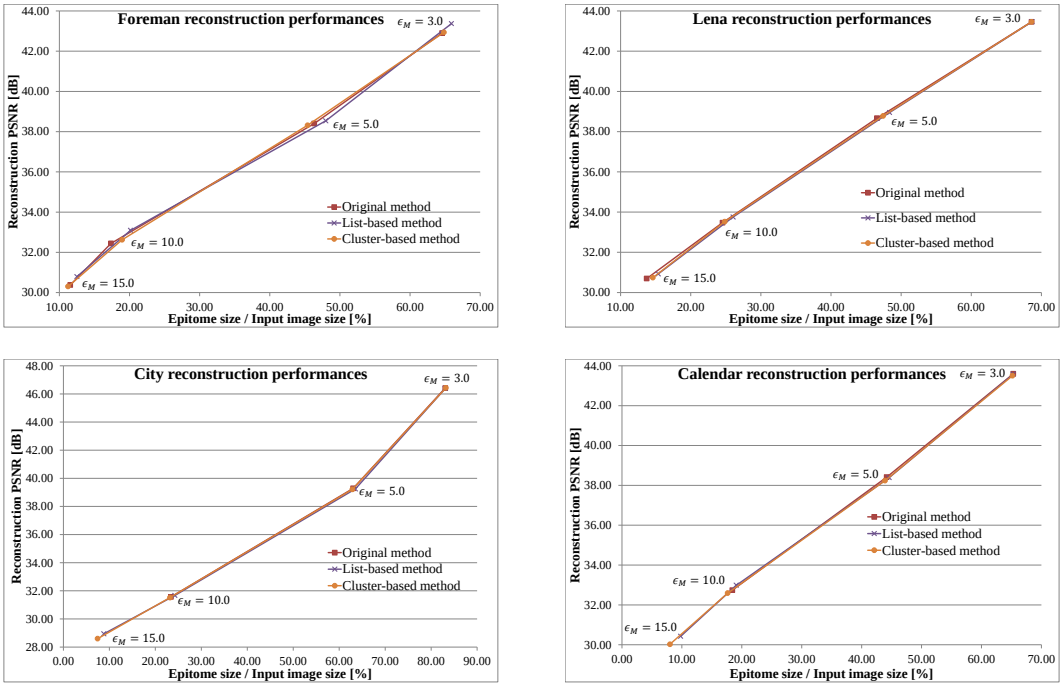


Figure 3.13: Reconstruction performances for the test sequences depending on the epitome size.

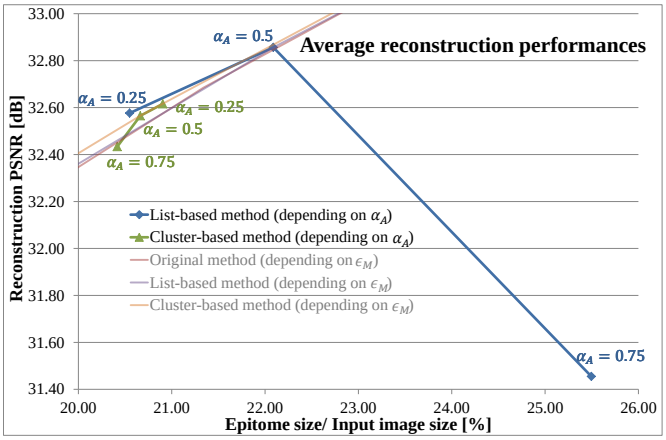


Figure 3.14: Reconstructed average PSNR vs epitome size.

3.3 Epitome-based quantization noise removal

In this section, we use an epitome as a compact representation of the image/sequence. The epitome is sent to the decoder at a high bit-rate, in order to provide high quality patches which are used to restore the decoded sequence.

3.3.1 Overview of the proposed method

Here, we described the proposed approach in a compression context. Formally, we consider the following notations: the source image/sequence is denoted X, the corresponding coded/decoded image/sequence is denoted Y, and $\hat{X}$ the de-noised image/sequence. We note E a texture epitome extracted from the source signal X. We will consider patches of size $M \times N$.

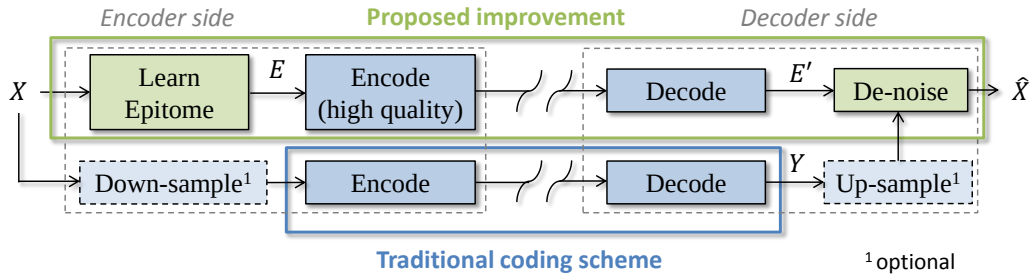


Figure 3.15: Proposed de-noising scheme for coding noise using epitomes.

The main idea of the proposed method is represented in Fig. 3.15 and consists in the following steps:

- At the encoder side:
  - learn a set of texture epitomes $\{E\}$ from the source sequence X (see section 3.3.2)
  - encode the set of epitomes at high quality (see section 3.3.3)
- At the decoder side:
  - decode the set of texture epitomes $\{E'\}$
  - de-noise the decoded sequence Y using the set of high quality epitomes $\{E'\}$ (see section 3.3.4)

The specific epitome-based de-noising methods are described in details in the section 3.3.4. Based on the proposed scheme, three applications are considered: first, we consider our method as an improvement of the traditional single layer coding scheme such as HEVC, and compare our performances directly against those of the single layer coding scheme. Second, the proposed improvement is considered as an enhancement layer for SNR scalability, and compared to a two-layer scalable coding scheme such as

SHVC (see section 1.2.3). Finally, the proposed scheme is considered as an enhancement layer for spatial scalability. In this last application, the base layer is a down-sampled version of the original sequence. At the decoder side, the proposed epitome-base de-noising methods are applied on an up-sampled version of the decoded base layer, and thus perform joint de-noising and super-resolution. The specific configurations of the two applications are described in the following sections.

3.3.2 Epitome generation

For the first application, the epitomes are generated using the method described in section 3.2.2, with a set of three matching thresholds $\varepsilon_M = 15.0, 10.0, 7.0$, and the corresponding association thresholds $\varepsilon_A = 0.5 * \varepsilon_M$, which generates epitomes of different sizes. The decoded sequence Y is processed by GOP, and the epitome are generated from the first frame of the GOP only, called key frames (KFs), in order to limit the additional bit-rate. In the experiments, the source sequence is encoded using HEVC with the random access configuration, thus a GOP consists in 8 frames. A GOP can thus be processed using two epitomes, the one generated from the first frame of the current GOP, and the one from the first frame of the following GOP (see Fig. 3.16).

In Table 3.4, we show the different sizes of epitomes obtained for the two KFs surrounding the first GOP of the test sequences. All the corresponding epitomes can be seen in annex in section B.2.

Table 3.4: Epitomes sizes for the key frames surrounding the first GOP of the test sequences. The sizes are evaluated as the percentage of pixels of each epitome over the total number of pixels in the key frame, and then over the total number of pixels in the GOP (GOP columns).

	$\varepsilon_M = 15.0$			$\varepsilon_M = 10.0$			$\varepsilon_M = 7.0$		
Sequence	KF 0	KF 1	GOP	KF 0	KF 1	GOP	KF 0	KF 1	GOP
City	21.15	24.62	5.09	36.36	40.91	8.59	53.66	59.59	12.58
Foreman	19.82	17.36	4.13	33.90	28.59	6.94	43.62	44.63	9.81
Macleans	49.31	48.17	10.83	66.29	66.10	14.71	79.29	78.85	17.57
Mobile	70.71	71.65	15.82	80.49	79.48	17.77	83.14	83.21	18.48

For the second and third applications, an epitome is generated from each frame of the decoded sequence Y, using the method described in section 3.2.2. The matching thresholds $\varepsilon_M = 7.0$ and $\varepsilon_M = 3.0$ are used for the SNR scalability and spatial scalability respectively, and the corresponding association thresholds is set to $\varepsilon_A = 0.5 * \varepsilon_M$. The non-epitome blocks of a frame are then processed using using three epitome, the one generate from the current frame and the ones from the two surrounding frames. In Tables 3.5 and 3.6, we show the different sizes of epitomes obtained for each frame of the first GOP of the test sequences, with $\varepsilon_M = 7.0$ and $\varepsilon_M = 3.0$ respectively.

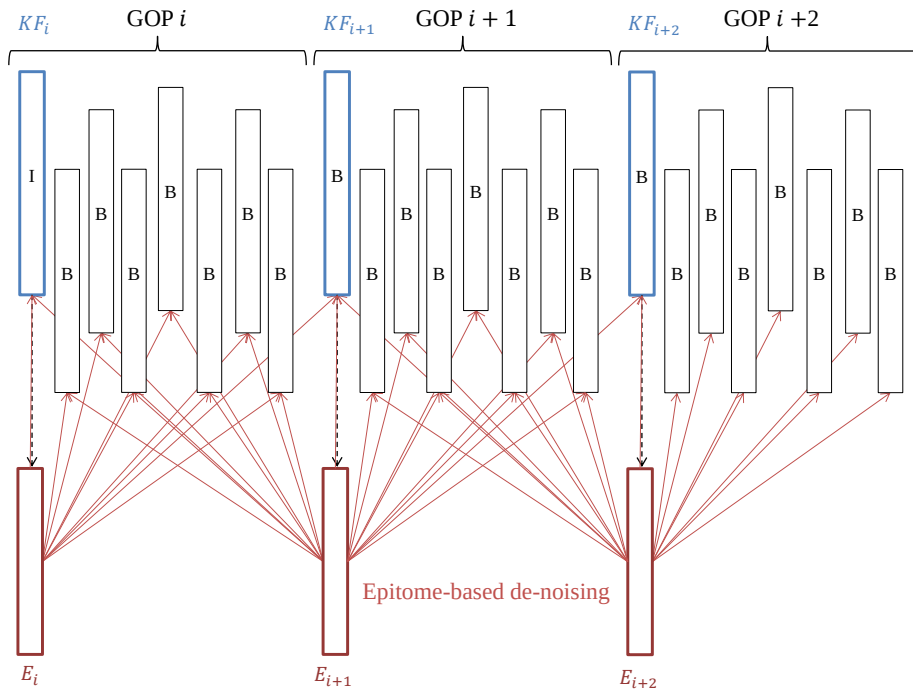


Figure 3.16: Epitomes are generated from the first frame of each GOP. A GOP i can then be processed using the two surrounding epitomes E_i and E_{i+1}. Note that the GOP structure depends on the random access coding structure of HEVC.

Table 3.5: Epitomes sizes for the frames of the first GOP of the test sequences, with $\varepsilon_M = 7.0$. The sizes are evaluated as the percentage of pixels of each epitome over the total number of pixels in the key frame, and then over the total number of pixels in the GOP (GOP columns).

	$\varepsilon_M = 7.0$									
Frame	0	1	2	3	4	5	6	7	8	GOP
City	53.66	53.03	54.10	56.94	56.76	58.08	58.65	59.15	59.60	56.66
Foreman	43.62	44.57	42.80	44.07	42.93	46.72	45.20	44.63	44.63	44.35
Macleans	79.29	81.62	80.87	80.05	80.24	80.28	79.67	78.47	78.85	79.93
Mobile	83.14	82.01	81.76	81.94	81.50	81.31	82.20	82.89	83.21	82.22

Table 3.6: Epitomes sizes for the frames of the first GOP of the test sequences, with $\varepsilon_M = 3.0$. The sizes are evaluated as the percentage of pixels of each epitome over the total number of pixels in the key frame, and then over the total number of pixels in the GOP (GOP columns).

	$\varepsilon_M = 3.0$									
Frame	0	1	2	3	4	5	6	7	8	GOP
City	86.43	85.29	86.49	88.01	87.18	88.64	90.21	89.33	89.46	87.89
Foreman	78.60	79.48	79.61	79.36	80.62	80.56	81.00	80.49	81.06	80.09
Macleans	96.09	97.03	96.02	96.34	96.78	96.34	95.83	96.97	97.16	96.51
Mobile	90.34	90.78	89.90	89.77	90.34	90.03	90.47	90.78	90.91	90.37

3.3.3 Epitome encoding

The texture epitomes are sent in addition to the complete sequence with a higher quality. Much information of the epitome texture is actually contained in the lower quality decoded frame from which it is extracted. We only need to encode the residue between the source epitome patches and the corresponding patches in the decoded frame.

For this task, we use the scalable extension of HEVC, SHVC (see section 1.2.3). For all applications, the encoded frames are used as base layer (BL), and the texture epitome are considered as the enhancement layer (EL). The non-epitome blocks of the EL are directly copied from the BL, thus their rate-cost is practically non-existent.

In Fig. 3.17, we show an example of epitomes encoding for the first application.

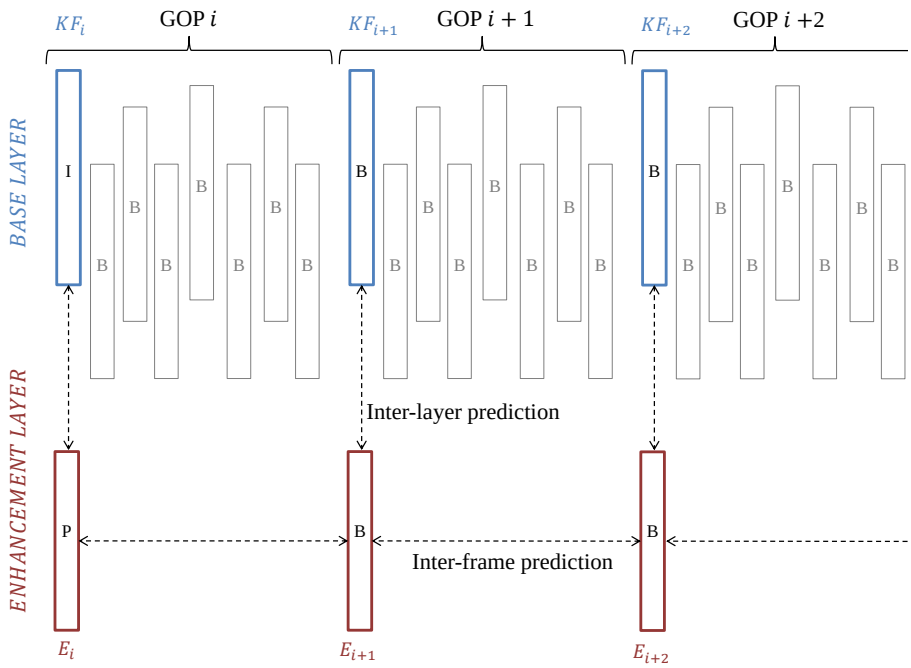


Figure 3.17: Epitomes are encoded using a scalable scheme. The encoded sequence is used as the base layer, while the epitome is considered as the enhancement layer. Note that the epitomes can then be predicted from the base layer as well as previously encoded epitomes.

For the first application, the quantization parameter (QP) for the EL has to be selected carefully, since a trade-off must be achieved between the quality of the epitomes received at the decoder, which will impact the de-noising performance, and the additional bit-rate associated with the epitomes. The QP value of the EL will also depend on the QP value of the BL, in order to limit the bit-rate of the epitomes with respect to the bit-rate of the BL (see section 3.3.5.2).

For the second and third applications, the QP value of the EL is the same as the EL of the scalable scheme to which we compare our performances.

3.3.4 Epitome-based de-noising

We describe here the proposed epitome-based de-noising techniques. To perform the de-noising, the frames are divided in $N \times N$ overlapping patches. To limit the complexity, not all the overlapping patches are processed, but instead we define a step s in both rows and columns between two processed patches. In the end, when several estimates are obtained for a pixel, they are averaged in order to obtain the final estimate. The main idea consists in the following steps:

- Search for the K-NN of the current patch among the overlapping coded/decoded patches colocated with the epitome patches
- Derive a de-noising method between the noisy K-NN patches and the corresponding high quality patches in the epitome
- Apply the previous de-noising method on the current patch to obtain the de-noised patch

In the case of the first application, the K-NN search is performed in the epitomes from the two closest key-frames. For the second and third applications, the K-NN search is performed in the epitome from the current frame and the closest past and future frames. The aforementioned "de-noising method" expression is to be defined, as many can be derived to solve this problem. Below, we propose several methods.

3.3.4.1 Epitome-based Local Linear Combination

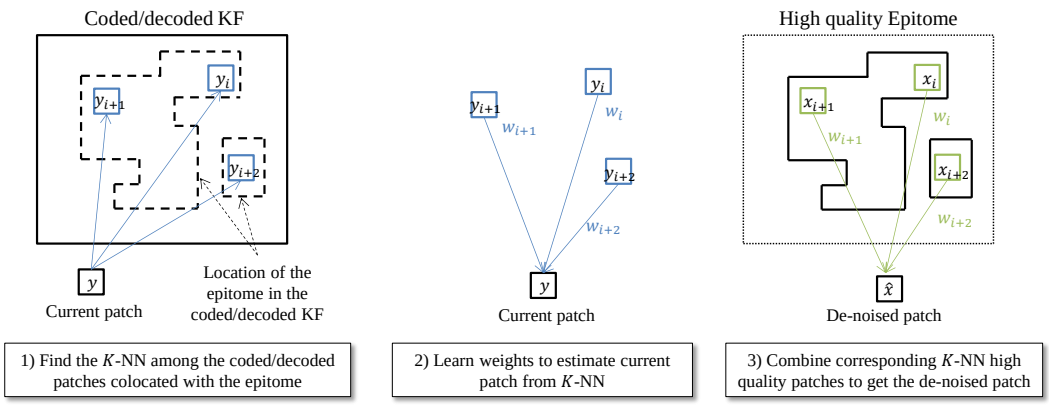


Figure 3.18: Epitome-based de-noising of coded/decoded patches using Local Linear Combination.

First, we propose a method that we denote "epitome-based Local Linear Combination" (LLC). Let y be the current patch to be de-noised. We note $y_i, i = 1 \dots K$ the K-NN of y. The corresponding high quality patches from the epitome are denoted $x_i, i = 1 \dots K$.

From y and its K-NN, we compute a set of weights $w_i, i = 1 \dots K$. This step is detailed below.

The de-noised estimated patch $\hat{x}$ is obtained as the linear combination of the K high quality patches:

$$\hat{x} = \frac{\sum_{i=1}^{K} w_i * x_i}{\sum_{i=1}^{K} w_i} \tag{3.20}$$

The method is illustrated in Fig. 3.18.

Two variants are proposed to estimate the weights of the linear combination. First, we adapt the well-known NLM algorithm and use exponential weights depending on the distance between y and its K-NN. We note $d_i = \frac{||y-y_i||_2^2}{N^2}$, and the weights are then computed as:

$$w_i = e^{-\frac{d_i}{2\sigma_{NLM}^2}} \tag{3.21}$$

where σ_{NLM} is a parameter that acts as a degree of filtering. As in the original NLM algorithm it is set to $\sigma_{NLM} = 10 * \sigma_n$. The noise level σ_n is estimated as the square root of the MSE between the epitome patches and the corresponding coded/decoded patches.

This method is named "Epitome-based NLM".

In a second variant, the weights are computed using Locally Linear Embedding (LLE), which was presented in the previous chapter for in-loop prediction. Here, the assumption is that the noise preserves the local geometry of the manifold. Thus, for each noisy patch, we can learn its embedding into the noisy patches colocated with the epitome patches, and then apply the weights of the embedding on the corresponding high quality patches from the epitome to obtain the de-noised patch.

The LLE finds the best estimate of y, in a least square sense, from a weighted combination of its K-NN, under the constraints that the weights sum to one. Let $\mathbf{M}_y$ be a matrix whose columns contain the vectorized K-NN of y, $y_i, i = 1 \dots K$. Let W be a vector containing the weights $w_i, i = 1 \dots K$. The LLE weights are obtained by solving the following equation:

$$\min_W ||\upsilon - \mathbf{M}_y W||_2^2 \text{ s.t. } \sum_{i=1}^{K} w_i = 1 \tag{3.22}$$

where υ is the vectorized version of y. The weights vector W is computed as

$$W = \frac{\mathbf{D}^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{1}}. \tag{3.23}$$

The term $\mathbf{D}$ denotes the local covariance matrix (*i.e.*, in reference to υ) of the K-NN stacked in $\mathbf{M}_y$, and $\mathbf{1}$ is the column vector of ones. In practice, instead of an explicit

inversion of the matrix $\mathbf{D}$, the linear system of equations $\mathbf{D}W = \mathbf{1}$ is solved, then the weights are rescaled so that they sum to one.

This method is named "Epitome-based LLE".

3.3.4.2 Epitome-based BM3D

We propose here an adaptation of the very popular BM3D algorithm. Note that the proposed technique can be generalized to any de-noising technique based on the thresholding/shrinkage of transformed coefficients. We explain below how the two steps of the BM3D algorithm, hard thresholding and Wiener filtering, can be adapted for epitome-based de-noising.

For clarity purpose, we group in Table 3.7 the numerous notations necessary to describe this method.

Table 3.7: Notations used for the epitome-based BM3D

First step: Hard Thresholding	
T_{HT}	3D transform used for the Hard Thresholding step
$\mathbf{G}_y$	3D group containing the K-NN of the current patch to be de-noised y
$\mathbf{G}_x^{HT}$	3D group containing the K high quality patches corresponding to the noisy K-NN contained in $\mathbf{G}_y$
$\mathbf{G}'_y$	3D group containing the same patches as $\mathbf{G}_y$,
	except for the closest NN of y which is replaces by y itself
$\mathbf{G}_{\hat{x}}^{HT}$	3D group containing the de-noised estimates of the patches in $\mathbf{G}'_y$
$\hat{x}_{HT}$	estimate of y at the end of the Hard Thresholding step
Second step: Wiener filtering	
This step is performed on the de-noised frames obtained at the previous step.	
T_{Wien}	3D transform used for the Wiener filtering step
$\mathbf{G}_{\hat{x}}$	3D group containing the K-NN of the current patch $\hat{x}$
$\mathbf{G}_x^{Wien}$	3D group containing the K high quality patches corresponding to the K-NN contained in $\mathbf{G}_{\hat{x}}$
$\mathbf{G}_n$	3D group containing the noise patches $\mathbf{G}_x^{Wien} - \mathbf{G}_{\hat{x}}$
$\mathbf{G}'_{\hat{x}}$	3D group containing the same patches as $\mathbf{G}_{\hat{x}}$,
	except for the closest NN of $\hat{x}$ which is replaces by $\hat{x}$ itself
$\mathbf{G}_{\hat{x}}^{Wien}$	3D group containing the de-noised estimates of the patches in $\mathbf{G}'_{\hat{x}}$
$\hat{x}_{Wien}$	final de-noise estimate of y at the end of the Wiener filtering step

First step: Hard Thresholding. One of the key and challenging point of such techniques is to choose the value of the threshold, that we note here τ. In the case of AWGN de-noising, the value of τ is usually set depending on the noise level. For example, the threshold in [29] is set to $\tau = 2.7 * \sigma_n$. However, an optimal de-noising rule for the thresholding is proposed in [81]. In general, this rule can not be applied as it requires the knowledge of the source signal, but it gives an upper bound on the performances of such transform-thresholding techniques. According to this rule, if we note c_x a coefficient from the transformed source signal and c_y the corresponding transform coefficient from the noisy signal, the corresponding de-noised transform coefficient $c_{\hat{x}}$ is obtained as follow:

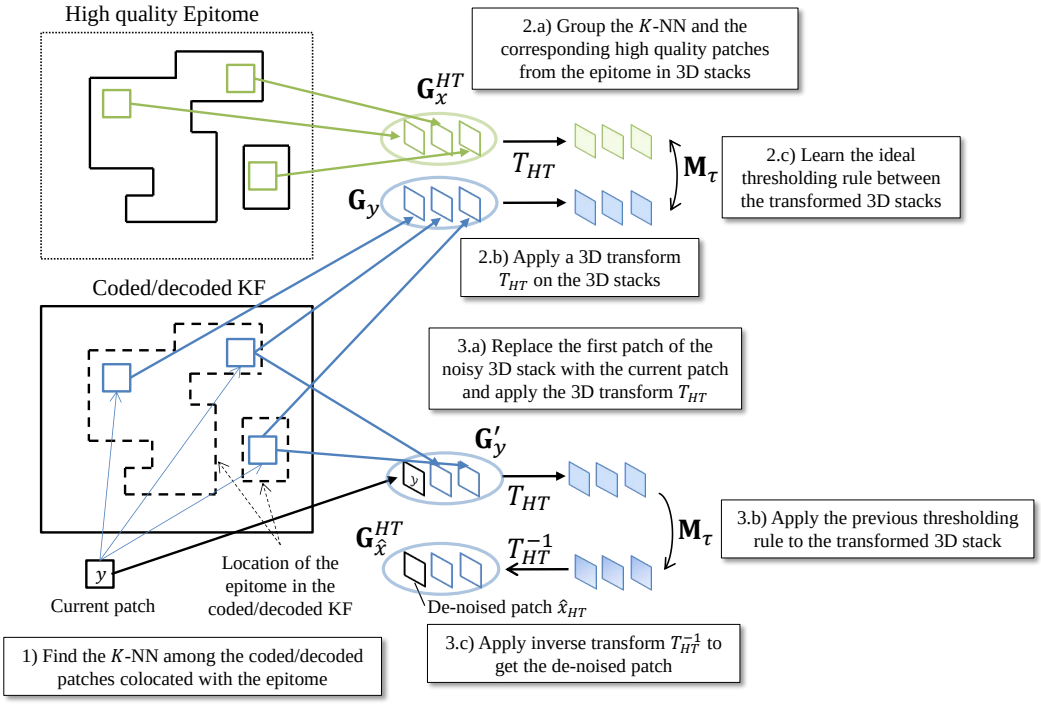


Figure 3.19: Epitome-based de-noising - first step of the adapted BM3D.

$$c_{\hat{x}} = \begin{cases} 0, & \text{if } |c_x - c_y| > |c_x| \\ c_y, & \text{otherwise.} \end{cases} \tag{3.24}$$

In other words, the coefficients with a SNR lower than a 1:1 ratio are canceled.

Here, we propose to use this rule in a epitome-based BM3D method, as the high quality patches from the epitome can be used to implement the above rule. More precisely, the K-NN patches and their corresponding high quality patches from the epitome are stacked in 3D groups that we note $\mathbf{G}_y$ and $\mathbf{G}_x^{HT}$ respectively. We then apply a 3D transform T_{HT} on both groups. From the two transformed groups and using the rule of Eq. 3.24, we can obtain a de-noising rule in the form of a binary 3D mask $\mathbf{M}_\tau$ which contains 0 where the coefficients are put to 0 and 1 otherwise:

$$\mathbf{M}_\tau(\xi) = \begin{cases} 0, & |T_{HT}(\mathbf{G}_x^{HT})(\xi) - T_{HT}(\mathbf{G}_y)(\xi)| > |T_{HT}(\mathbf{G}_x^{HT})(\xi)| \\ 1, & \text{otherwise.} \end{cases} \tag{3.25}$$

To de-noise the current patch y, we replace in $\mathbf{G}_y$ the closest NN of y by the patch y itself, to obtain a 3D group noted $\mathbf{G}'_y$. We can then apply the transform T_{HT} to $\mathbf{G}'_y$ followed by the thresholding rule $\mathbf{M}_\tau$, and finally the inverse transform to obtain the de-noised group $\mathbf{G}_{\hat{x}}^{HT}$:

$$\mathbf{G}_{\hat{x}}^{HT} = T_{HT}^{-1}(\mathbf{M}_\tau \cdot T_{HT}(\mathbf{G}'_y)) \tag{3.26}$$

where $\cdot$ represents an element-by-element multiplication.

The first step de-noised patch $\hat{x}_{HT}$ is then extracted from $\mathbf{G}_{\hat{x}}^{HT}$ at the same position as the one of y in $\mathbf{G}'_y$. This step is illustrated in Fig. 3.19.

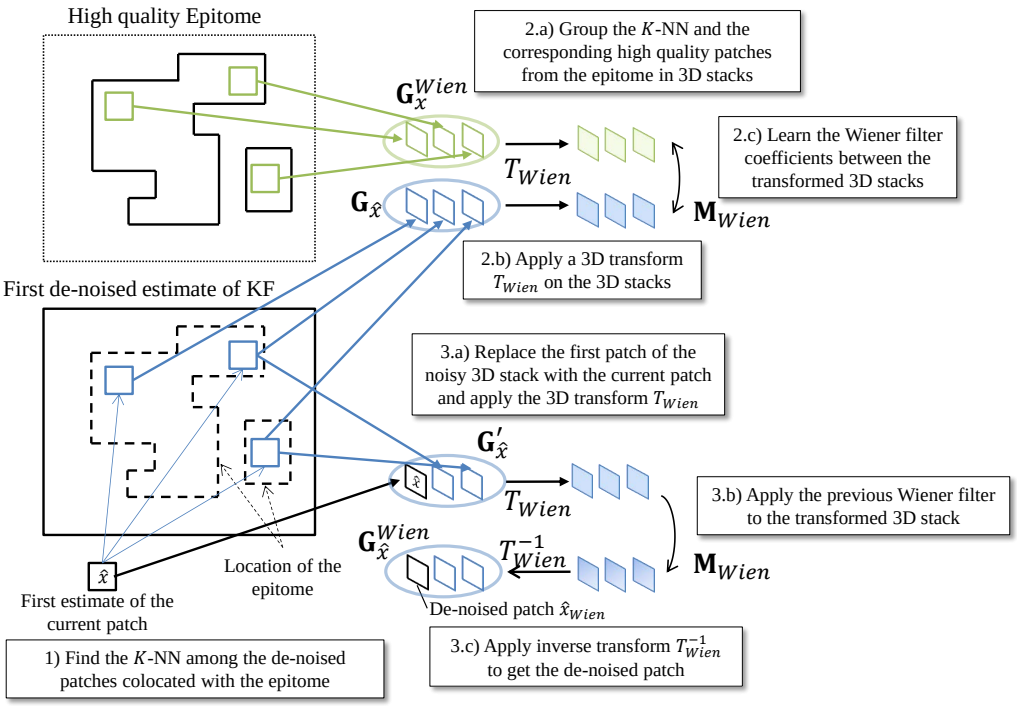


Figure 3.20: Epitome-based de-noising - second step of the adapted BM3D.

Second step: Wiener filtering. The second step of the original BM3D algorithm consists in a Wiener filtering of the 3D transform group where the first de-noised estimate obtained at the previous step is used as oracle. Here, we propose to adapt this step by using the high quality patches from the epitome as oracle for the Wiener filtering. Note that this step is performed on the de-noised estimate obtained at the previous step and not directly on the noisy frames.

We first search for the K-NN of the current patch $\hat{x}$ among the first estimate patches colocated with the epitome patches from the two closest key-frames. The K-NN patches and their corresponding high quality patches from the epitome are stacked in 3D groups that we note $\mathbf{G}_{\hat{x}}$ and $\mathbf{G}_x^{Wien}$. Note that these groups are different from the previous step groups $\mathbf{G}_{\hat{x}}^{HT}$ and $\mathbf{G}_x^{HT}$ respectively because the K-NN patches are different. We also compute a third 3D group containing the corresponding noise patches:

$$\mathbf{G}_n = \mathbf{G}_x^{Wien} - \mathbf{G}_{\hat{x}} \tag{3.27}$$

We then apply a 3D transform T_{Wien} on both groups. We can then compute the Wiener filter coefficients:

$$\mathbf{M}_{Wien} = \frac{|T_{Wien}(\mathbf{G}_x^{Wien})|^2}{|T_{Wien}(\mathbf{G}_x^{Wien})|^2 + |T_{Wien}(\mathbf{G}_n)|^2} \tag{3.28}$$

To de-noise the current patch $\hat{x}$, we replace in $\mathbf{G}_{\hat{x}}$ the closest NN of $\hat{x}$ by the patch $\hat{x}$ itself, to obtain a 3D group noted $\mathbf{G}'_{\hat{x}}$. We can then apply the transform T_{Wien} to $\mathbf{G}'_{\hat{x}}$ followed by the Wiener filtering, and finally apply the inverse transform to obtain the de-noised group $\mathbf{G}_{\hat{x}}^{Wien}$:

$$\mathbf{G}_{\hat{x}}^{Wien} = T_{Wien}^{-1}(\mathbf{M}_{Wien} \cdot T_{Wien}(\mathbf{G}'_{\hat{x}})) \tag{3.29}$$

The final de-noised patch $\hat{x}_{Wien}$ is then extracted from $\mathbf{G}_{\hat{x}}^{Wien}$ at the same position as the one of $\hat{x}$ in $\mathbf{G}'_{\hat{x}}$. This step is illustrated in Fig. 3.20.

3.3.4.3 Epitome-based Local Linear Mapping

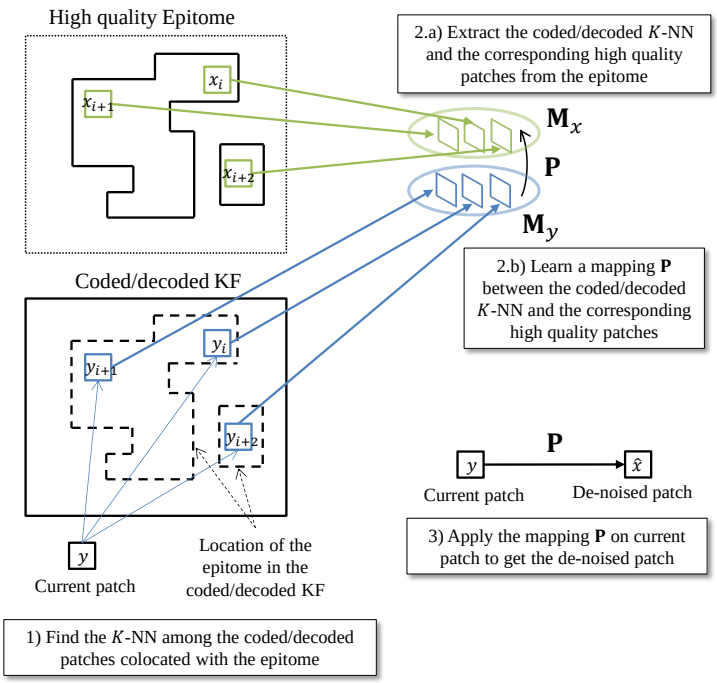


Figure 3.21: Epitome based de-noising of coded/decoded patches using Local Linear Mapping.

In this section, we proposed a novel method based on linear mappings that we call "epitome-based Local Linear Mapping (LLM)". Let y be the current patch to be de-noised. We note $y_i, i = 1 \dots K$ the K-NN of y. The corresponding high quality patches from the epitome are denoted $x_i, i = 1 \dots K$. Let $\mathbf{M}_y$ and $\mathbf{M}_x$ be the matrices containing in their columns the vectorized y_i and x_i respectively. We can then learn a function, more precisely here a linear mapping, estimating $\mathbf{M}_x$ from $\mathbf{M}_y$.

Considering multivariate linear regression, the problem is of searching for the function $\mathbf{P}$ minimizing:

$$\mathbf{E} = ||\mathbf{M}_x^T - \mathbf{M}_y^T \mathbf{P}^T||^2 \tag{3.30}$$

which is of the form $||\mathbf{Y} - \mathbf{XB}||^2$ (corresponding to the linear regression model $\mathbf{Y} = \mathbf{XB} + \mathbf{E}$). The minimization of Eq. (3.30) gives the least squares estimator

$$\mathbf{P} = \mathbf{M}_x \mathbf{M}_y^T (\mathbf{M}_y \mathbf{M}_y^T)^{-1} \tag{3.31}$$

We finally obtain the de-noised patch by applying the learned mapping to the current patch:

$$\hat{x} = \mathbf{P}y \tag{3.32}$$

The method is illustrated in Fig. 3.21.

3.3.5 Simulation and results

In this section, the epitome-based methods described in the previous section are applied in the proposed schemes described in section 3.3.1.

Four test sequences of resolution 352×288 are used: City, Foreman, Macleans, and Mobile. For each sequence, one GOP of 8 frames is processed. In the case of the first application in a single layer scheme, the epitome of the first key frame of the following GOP is also used. The sequences are encoded with the HEVC test model HM (ver 15.0) [82] using the Main profile in Random Access, with 4 values for the Quantization Parameter, $QP = 22, 27, 32, 37$.

For the second and third applications, SNR and spatial scalablity, we compare our performance with the SHVC scalable scheme. For SNR scalability, the sequences encoded with HEVC as described above are used as base layer, and the enhancement layer is encoded with the SHVC test model SHM (ver. 9.0) [83], with $QP = 18, 23, 28, 33$. For spatial scalability, the base layer is down-sampled with a factor 2, and encoded with the HEVC test model HM (ver 15.0) using the Main profile in Random Access, with 4 values for the Quantization Parameter, $QP = 22, 27, 32, 37$. The enhancement layer is encoded with the SHVC test model SHM (ver. 9.0), with $QP = 22, 27, 32, 37$.

3.3.5.1 De-noising with source epitome

In this section we evaluate the epitome-based de-noising performances for the first application. In order to obtain the full potential of the method, the epitome is considered known without any loss. In practice, this means the epitome would be encoded using lossless coding.

Average results for the GOP of the test sequences are given in Table 3.8. Overall, the proposed methods are more efficient at low bit-rates, *i.e.* for stronger quantization noise. The proposed methods also benefit from bigger epitomes (lower ε_M), since more information on the source signal is available. We can see that the epitome-based BM3D

and LLM methods perform in general better than the epitome-based NLM and LLE methods. In facts, for the former, the de-noising rule learning is directly made between the source and the noisy patches, while for the latter, it is only learned on the noisy patches, and *then* applied to the source patches.

Table 3.8: De-noising performances (PSNR in dB) of epitome-based methods for coding noise averaged over the test sequences. The bold values indicate a PSNR above the corresponding PSNR of the decoded sequence.

		Original methods		Epitome-based methods $\varepsilon_M = 15.0$			
QP	Coded / Decoded	NLM	BM3D	NLM	LLE	BM3D	LLM
22	40.354	36.390	40.328	32.460	32.351	39.669	34.422
27	36.920	34.348	**36.941**	32.064	32.049	36.575	33.375
32	33.788	32.425	**33.859**	31.460	31.479	**33.802**	32.078
37	30.793	30.226	**30.875**	30.489	30.528	**31.126**	**30.929**

	Epitome-based methods							
	$\varepsilon_M = 10.0$				$\varepsilon_M = 7.0$			
QP	NLM	LLE	BM3D	LLM	NLM	LLE	BM3D	LLM
22	33.532	33.394	40.110	36.345	34.041	33.897	40.331	36.915
27	33.075	33.045	**36.981**	35.281	33.553	33.525	**37.200**	35.804
32	32.405	32.426	**34.185**	**33.951**	32.863	32.894	**34.378**	**34.453**
37	**31.378**	**31.418**	**31.440**	**31.951**	**31.809**	**31.869**	**31.587**	**32.886**

The previous results are averaged for the frames of the GOP, but do not necessarily reflect the temporal performances. Results indicate that the de-noising performances are not homogeneous for all the frames of the GOP. In facts, the PSNR gain is much more important for the key frames, where the high quality epitomes are known. An example of such behavior is given in the graph of Fig. 3.22, which corresponds to the de-noising performances for the City sequence encoded at $QP = 27$, with an epitome obtained with $\varepsilon_M = 7.0$ We can see that the epitome-based BM3D and LLM yield an important PSNR gain for the key frames (indexes 279 and 287). However, for the intermediates frames, the epitome-based BM3D has similar performances to the original BM3D, and the epitome-based LLM PSNR is sometimes below the PSNR of the decoded frames.

Tests were also performed using "random epitomes", which are constructed by randomly selecting patches in the KFs, with the same number of patches as in the actual epitome. Results showed that the "random epitomes" perform poorly compared to the actual epitomes, which demonstrates the interest of the concept of epitome to obtain representative textures for a GOP, and their use in the proposed methods. An example of such result is given in Table 3.9. An example of "random epitomes" is given in comparison with the actual epitomes for the key frames of the City sequence in Fig. 3.23.

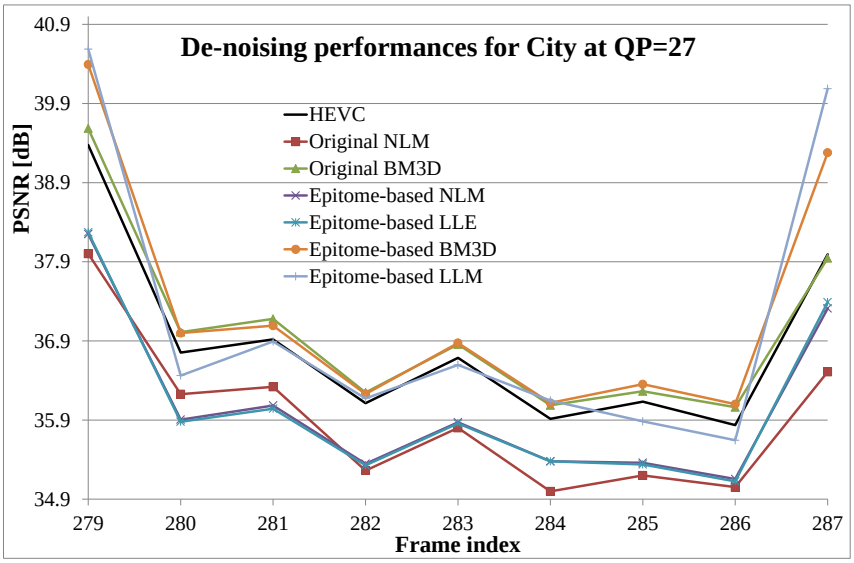


Figure 3.22: De-noising performances on the first GOP of the City sequence encoded with HEVC with $QP = 27$.

Table 3.9: Comparison of de-noising performances (PSNR in dB) between the epitome and a "random epitome" ($\varepsilon_M = 7.0$) averaged over the test sequences encoded at $QP = 37$

	Epitome	Random epitome
NLM	31.809	29.045
LLE	31.869	28.836
BM3D	31.587	31.204
LLM	32.886	29.856

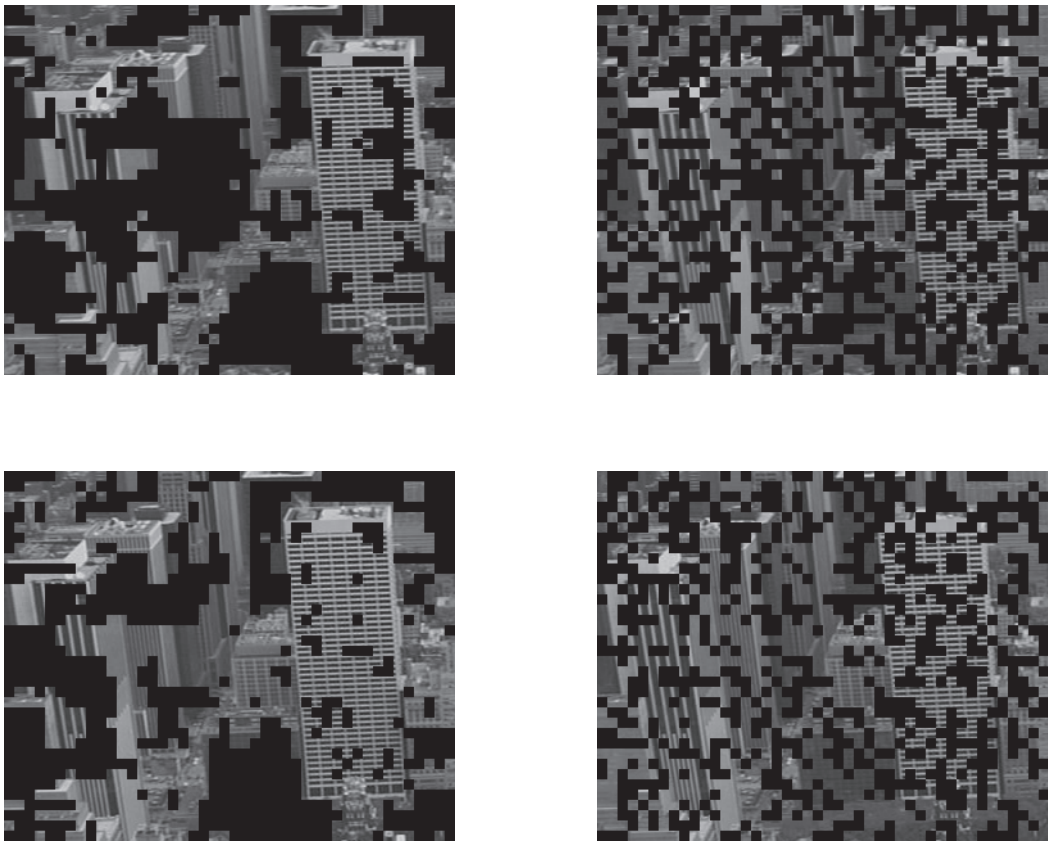


Figure 3.23: Epitomes (left) extracted from the first (top) and last (bottom) frames of the City (crop) GOP. The epitome was obtained with the threshold $\varepsilon_M = 7.0$, and the epitomes account for 53.6616 % and 59.596 % of the original frames size respectively. On the right are shown "random epitomes" of the same size.

3.3.5.2 RD performances in a single layer coding scheme

The results given in Table 3.8 correspond to the case where the epitomes are lossless encoded, which allows to evaluate the de-noising performances. However, in terms of RD performances, the lossless encoding of the epitomes might no be suitable, as the bit-rate associated to the epitomes becomes prohibitive. Thus in this section, in addition to the lossless case, we consider epitomes encoded as described in section 3.3.3. Different QP values are used depending on the QP used to encode the GOP, summarized in Table 3.10.

Table 3.10: Choice of the quantization parameter to encode the epitome.

	QP BL			
	22	27	32	37
QP EL	lossless			
	6	8	9	11
	12	16	18	22
	18	23	28	33

An example of RD curves is given for the City sequence in Fig. 3.24. Results in Table 3.8 show that the epitome-based NLM and LLE methods reach very similar performances. Thus in this section, we only give the results for the epitome-based LLE method. Due to the numerous parameters, many points corresponding to our methods are obtained, and represented as scatter plots in Fig. 3.24. To limit the number of points, we only kept the one whose PSNR values are above the corresponding PSNR values of the decoded sequence (similar to the bold numbers in Table 3.8. The best RD performances obtained with the epitome-based de-noising methods (purple curve in the figure) are obtained with the biggest epitomes ($\varepsilon_M = 7.0$) encoded with the highest of the proposed QP values. However, we can see that even the best RD performances are outperformed by HEVC.

The RD performances shown in the figures above are not as good as expected, as the de-noising improvement of the proposed methods are not sufficient to justify the additional rate cost. This limited average PSNR improvement is partly due to the fact that the proposed methods are mainly efficient for the key frames from which the epitomes are extracted, but have little to no de-noising effect on the intermediate frames. A typical example of such phenomenon is shown in Fig. 3.25 for the Macleans sequence, encoded at $QP = 22$, de-noised with the epitome-based BM3D ($\varepsilon_M = 7.0$). This can be explained by the GOP structure used in the common test conditions of HEVC, which defines different QP values for the frames of the GOP. This can be seen in Fig. 3.25, where the PSNR varies depending on the frame index, which also means that the noise properties vary as well. Consequently, the texture information contain in the epitome of the key frames might not be adapted to efficiently de-noised the intermediate frames. In addition, quantization artifacts removal is a very challenging problem, as many of the assumptions usually made for additive white Gaussian noise (AWGN) are violated.

In fact, we show in Fig. 3.26 the performances of the epitome-based de-noising methods ($\epsilon_M = 7.0$) for the City sequence corrupted with AWGN with $\sigma_n = 10$. We

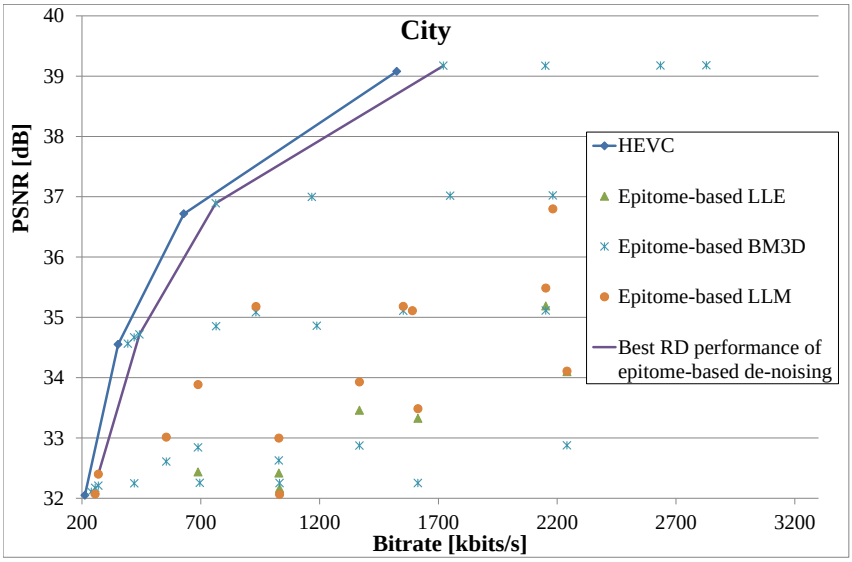


Figure 3.24: RD performances for City when encoding epitomes of several sizes using different QP values in a single layer scheme.

can see that, even if the de-noising is more effective on the key frames, the intermediate are efficiently de-noised as well.

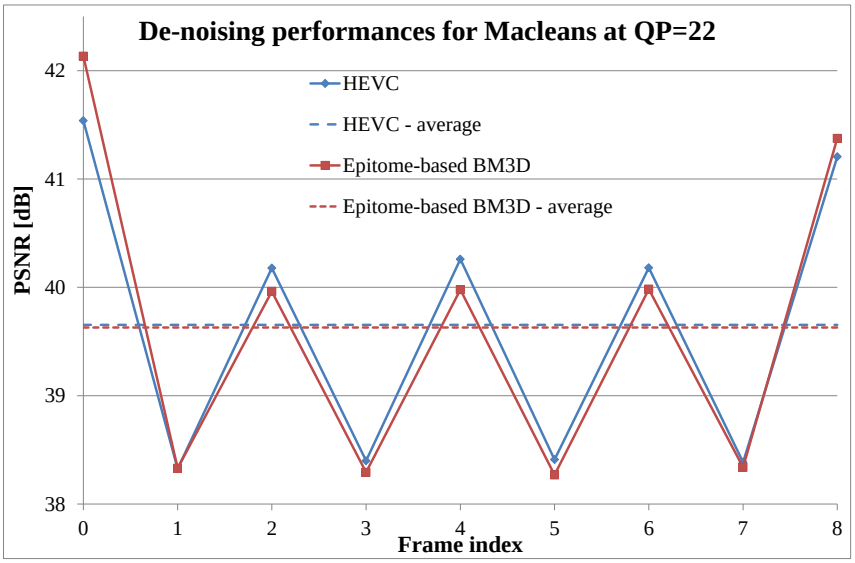


Figure 3.25: De-noising performances on the first GOP of the Macleans sequence encoded with HEVC with $QP = 22$.

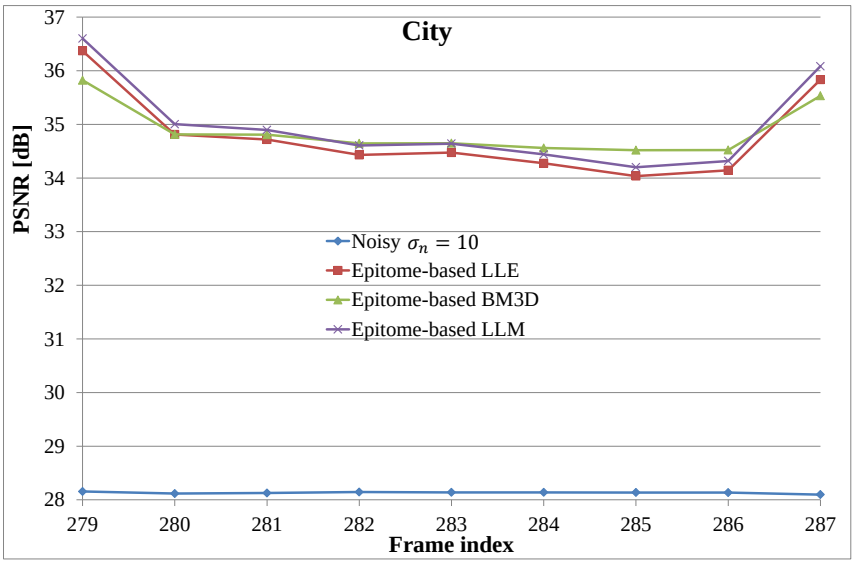


Figure 3.26: City sequence corrupted with AWGN ($\sigma_n = 10$), de-noised with the epitome-based methods. The information contained in the KF epitomes was sufficient to de-noise the remaining frames of the GOP.

3.3.5.3 RD performances in a SNR scalable coding scheme

For this application, we compare our results to the enhancement layer (EL) of SHVC, encoded with $QP = 18, 23, 28, 33$. The epitomes are encoded with the same QP values. As for the first applications, the epitome-based de-noising methods used are LLE, BM3D and LLM.

The RD curves for the four sequences are given in Fig. 3.27. Here, since an epitome is transmitted for each frame, the limitations of the first application in terms of de-noising are overcome.

We can see that our methods, notably the epitome-based BM3D, can outperform SHVC.

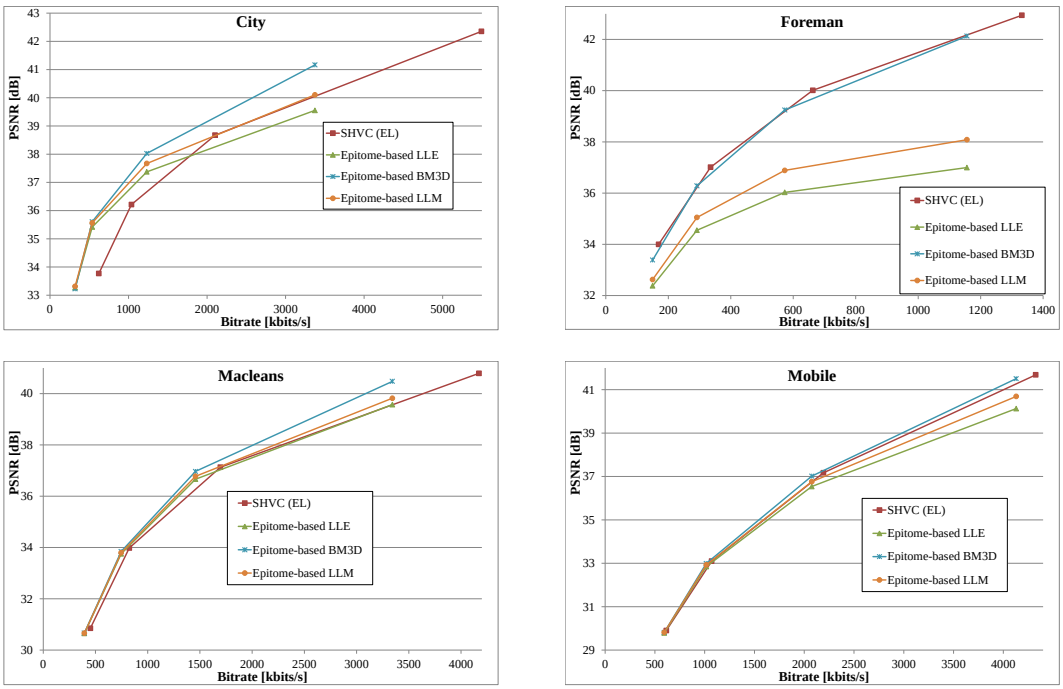


Figure 3.27: RD performances of the proposed methods in a SNR scalable scheme.

3.3.5.4 RD performances in a spatial scalable coding scheme

For this application, we compare our results to the enhancement layer (EL) of SHVC, encoded with $QP = 22, 27, 32, 37$, with a base layer down-sampled with a factor 2×2. The epitomes are encoded with the same QP values.

The RD curves for the four sequences are given in Fig. 3.28.

Here, the proposed methods jointly perform de-noising and super-resolution, and we can see that they can still outperform SHVC. However, contrary to the previous application, the best performing method is not necessarily the epitome-based BM3D, but rather the epitome-based LLM.

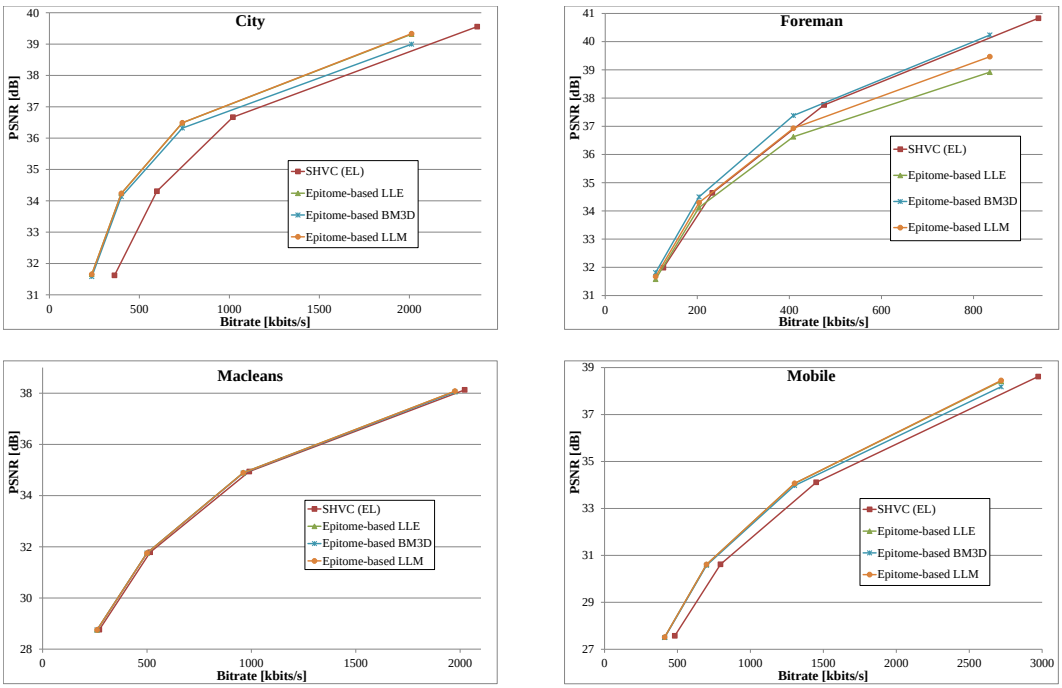


Figure 3.28: RD performances of the proposed methods in a spatial scalable scheme.

3.4 Conclusions and perspectives

In this chapter, we first presented efficient epitome generation methods, based on list or cluster approaches. By grouping non-overlapping blocks before performing exhaustive matching searches among overlapping blocks, the memory occupation and the processing time are reduced.

Second, the epitome generation method was used in an epitome-based quantization noise removal scheme. The proposed out-of-the-loop framework aims at improving the performances of a video compression scheme by sending high-quality epitomes to the decoder, where the they are used to de-noise the non-epitome patches of the video sequence.

The epitome patches being representative of the textural content of the video sequence, the non-epitome patches can be de-noised using similar patches from the epitome. In the proposed epitome-based de-noising methods, a patch is thus de-noised by first looking for its K-NN among the coded/decoded patches colocated with the epitome. We then propose three method to exploit the pairs of coded/decoded K-NN patches and their corresponding high quality patches from the epitome. The epitome-based local linear combination (LLC) computes weights for each nearest neighbor, and then linearly combines the corresponding high quality patches using the previous weights to obtain the de-noised patch. The weights can be obtained similarly to the NLM algorithm, or using an LLE-based approach. The epitome-based BM3D adapts the original BM3D

algorithm by inferring the thresholds for the hard thresholding and Wiener filter coefficients from the high quality patches. Finally, the epitome-based local linear mapping (LLM) method learns a linear mapping projection between the K-NN coded/decoded patches and their corresponding high quality patches, and then apply the projection on the patch being de-noised.

Three applications of the scheme were considered. We first evaluated the performances when epitomes are sent only for key frames of the video sequence, in order to improve a single layer scheme. By limiting the number of epitomes transmitted to the decoder, we want to limit the associated bit-rate while improving the overall quality of the decoded sequence, and thus the encoding performances. However, the results showed that, except for the key frames, the de-noising performances were limited, and consequently the RD performances as well.

In a second application, the epitomes are considered as an enhancement layer in a SNR scalable scheme, and an epitome is generated for each frame of the video sequence. Finally, the proposed methods are used as an enhancement layer for spatial scalability. The results show that for both scalable applications, the proposed methods could outperform the standard SHVC.

We believe that several approaches can be considered to further improve the performances of the proposed scheme.

First, we could consider applying a first pass "blind" de-noising algorithm to the decoded sequence. The term "blind" is used to characterize a method for which no side information needs to be transmitted to the decoder. Any state-of-the-art method, such as the one presented in section 3.1.2, is eligible. The first pass de-noising is likely to improve the K-NN search precision for our epitome-based de-noising method, which is expected to further improve the de-noising performances of our methods.

A second aspect would be to optimize the epitome generation algorithm. First, by taking into account the temporal redundancies, the epitome could be further compacted, which is expected to reduce the bit-rate associated with the epitome while maintaining similar de-noising performances. A direct implementation from existing algorithms would be to generate an epitome of the frame epitomes. In another approach, a sprite [84][85] of the GOP would be created, and an epitome of the sprite would be then generated. Such improvement would be especially interesting for the second application, where epitomes are currently generated independently for each frame. Second, the epitome model used here is generic, and has been used for different applications. However, an application-driven epitomic model could improve the performances of said application. In particular, an optimized epitome generation technique was presented in [86][2] for a LLE-based multi-patches super-resolution application. The results show that the epitomes obtained are more compact than the ones used here for a similar reconstruction quality.

Finally, the scheme could be extended to other scalable applications. In the experiments, we showed that the spatial and SNR scalability were efficient, but extensions

[2]Co-authored publication

to color gamut scalability, or compression of hygh dynamic range (HDR) videos from a low dynamic range base layer, are straightforward.

Chapter 4

Clustering-based quantization noise removal

In chapter 3, we proposed an epitome-based quantization noise removal method to improve coding performances of existing compression schemes. The out-of-the-loop proposed scheme was proven efficient for a scalable application, however, the single layer application was outperformed by the standard compression scheme because the PSNR gains could not compensate the cost of the epitomes in terms of bit-rate.

In this chapter, we propose a comparable out-of-the-loop quantization noise removal scheme for video compression. However, we adopt here a different perspective: instead of deriving a de-noising method between the noisy patches and their corresponding high quality patches in the epitome at the decoder side, we derive the parameters of the de-noising method at the encoder side between the noisy patches and their corresponding source signal. The derived parameters are then sent to the decoder to perform the de-noising. To take into account the redundancies of the signal, we perform clustering on the patches, and learn parameters for each cluster. The Local Linear Mapping (LLM) based multi-patches technique presented in the previous chapter is especially suited for the scheme presented in this chapter. In fact, a linear mapping can be learned for each cluster, and transmitted in a matrix form.

In comparison, the Local Linear Combination presented in the previous chapter could not be adapted here. The BM3D may be adapted to the new scheme, but would require the transmission of many parameters, namely all the 3D transform thresholds and Wiener filter coefficients.

Thus, on one hand, we expect the additional bit-rate to be limited, in particular compared to the bit-rate associated to the epitomes. On the other hand, the linear mappings are learned between noisy patches and the corresponding *source* patches, which we expect to be efficient in terms of de-noising. This idea is also corroborated by the proven efficiency of de-noising methods based on clustering [71][72][67].

This chapter is organized as follow. Section 4.1 reviews background on clustering and classification methods. Section 4.2 describes the proposed scheme based on clustering for quantization noise removal. Section 4.3 introduces an optimal clustering algorithm

for the clustering-based linear mapping de-noising. Section 4.4 presents an adaptation of the proposed scheme for super-resolution in a scalable compression scheme. Finally, we propose in section 4.5 a classification-based scheme which aims at reaching a compromise between the clustering approach described in section 4.2 and the optimal clustering presented in 4.3.

4.1 Background

Note that the contributions of this chapter also rely on the de-noising background, described in section 3.1.2. In this section, we present two machine learning tasks, un-supervised and supervised learning.

4.1.1 Un-supervised learning

Un-supervised learning, also called clustering, aims at partitioning a set of points of a p-dimensional space into several subsets, called clusters, so that points in a same cluster are similar according to a pre-defined criterion, while points belonging to different cluster are dissimilar.

In this section, we consider n data points to be clustered, noted $\mathbf{x}_i, i = 1 \dots n$. The number K of clusters is a pre-defined parameter.

Below, we present several popular clustering algorithms.

4.1.1.1 K-means algorithm

The K-means algorithm is among the most known clustering algorithm and was first described in [30]. This algorithm assigns a data point to the cluster with the nearest mean. Formally, the K-means algorithm solves the following equation:

$$\min_{\mathbf{C}} \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \boldsymbol{\mu}_i||^2 \tag{4.1}$$

where $\boldsymbol{\mu}_i$ is the mean of the cluster C_i and $\mathbf{C} = \{C_i\}_{i=1 \dots K}$.

The standard algorithm to solve this problem is given in Algorithm 2, where the initial set of centroids $\boldsymbol{\mu}_1^{(1)} \dots \boldsymbol{\mu}_K^{(1)}$ is considered known. This initial set can be obtained by different methods. A popular method randomly select K vectors from the data set and used them as initial centroids. Another method perform the K-means algorithm on a small subset of the data with a random initialization. The outcome is then used to initialize the centroids to cluster the full data set.

This two steps algorithm can be seen as a variant of the Expectation-Maximization algorithm, presented in the following section.

4.1.1.2 Gaussian Mixture Model (GMM)

The Gaussian Mixture Model (GMM) assumes that the data points are drawn from a mixture of K Gaussian distributions, which form the latent clusters.

Algorithm 2 K-means algorithm

Input: clusters number K, $p \times n$ data matrix $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$
Output: cluster set $\mathbf{C} = \{C_i | i = 1 \ldots K\}$
Repeat the two following steps until convergence:
Assignment step:

$$\forall i = 1 \ldots K, C_i^{(t)} = \{\mathbf{x} : ||\mathbf{x} - \boldsymbol{\mu}_i^{(t)}||^2 \leq ||\mathbf{x} - \boldsymbol{\mu}_j^{(t)}||^2 \ \forall j, 1 \leq j \leq K\}$$

Update step:

$$\forall i = 1 \ldots K, \boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{\mathbf{x}_j \in C_i^{(t)}} \mathbf{x}_j$$

Let us first formally define the problem. The underlying density $p(\mathbf{x})$ can be formally expressed as:

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|z_k, \theta_k) \tag{4.2}$$

where $p_k(\mathbf{x}|z_k, \theta_k)$ is a mixture component, with $\mathbf{z} = (z_1, \ldots, z_K)$ is a vector of binary indicator, with only one component equal to 1, and the others are 0. For each point $\mathbf{x}_i$, $\mathbf{z}_i$ is a random variable which represents the mixture component which generated $\mathbf{x}_i$. Each component is a multivariate Gaussian density, expressed as:

$$p_k(\mathbf{x}|\theta_k) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma_k}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)} \tag{4.3}$$

and $\theta_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$.

The mixture weights $\alpha_k = p(z_k)$ represent the probability that a random point $\mathbf{x}$ was generated by the component k, and $\sum_{k=1}^{K} \alpha_k = 1$.

The complete set of parameters for the mixture model is $\Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}$.

Given the set of points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$, the problem in hand is to estimate the $n \times K$ latent variables matrix $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1\ldots n}$ and the model parameters Θ. The estimates are obtained using the Expectation - Maximization (EM) algorithm [87], a generic iterative method to find maximum likelihood estimates of the latent variables and parameters of a model. The EM algorithm to estimate the GMM is given in Algorithm 3, where the initial parameters $\Theta^{(0)}$ are considered known. The initial parameters are usually obtained by randomly selecting K points from the data set to initialize the means, and set the covariance of the whole data set for each of the K covariance matrices. The mixture weight α_k are set homogeneously with the value $\frac{1}{K}$.

Note that the EM algorithm does not produce directly the latent variables matrix $\mathbf{Z}$, but instead a $n \times K$ "membership probability" matrix $\mathbf{W}$. The latent variable $\mathbf{z}_i$ can be obtained from $\mathbf{W}$ by taking the i-th row and set the maximum value to 1 and the other values to 0.

Algorithm 3 EM algorithm for GMM

Input: clusters number K, $p \times n$ data matrix $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$
Output: $n \times K$ "membership probability" matrix $\mathbf{W}$, parameters Θ
Repeat the two following steps until convergence:
Expectation step:
For each point $\mathbf{x}_i$, compute its membership probability with respect to each cluster k, given the parameters $\Theta^{(t)}$:

$$\mathbf{W}_{i,k} = p(z_{ik} = 1|\mathbf{x}_i, \Theta^{(t)}) = \frac{p_k(\mathbf{x}_i|z_k, \theta_k^{(t)}) \cdot \alpha_k^{(t)}}{\sum_{l=1}^{K} p_l(\mathbf{x}_i|z_l, \theta_l^{(t)}) \cdot \alpha_l^{(t)}}$$

Maximization step:
Let $n_k = \sum_{i=1}^{n} \mathbf{W}_{i,k}$. Update the parameters $\Theta^{(t+1)}$:

$$\alpha_k^{(t+1)} = \frac{n_k}{n},\ k = 1, \ldots, K$$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{n_k} \sum_{i=1}^{n} \mathbf{W}_{i,k} \cdot \mathbf{x}_i,\ k = 1, \ldots, K$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{n_k} \sum_{i=1}^{n} \mathbf{W}_{i,k} \cdot (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)})^T,\ k = 1, \ldots, K$$

4.1.1.3 Spectral clustering

Spectral clustering is one of the most popular modern clustering algorithms [88], and usually outperforms the K-means algorithm. The main idea of spectral clustering is to defines a similarity graph, in which each vertex represents a data point, and each edge between two vertices is weighted with the similarity between these two vertices. The clusters are obtained by grouping the vertices of the graph which have strong connections, *i.e.* strong weights on their edges.

Formally, the spectrum (eigenvalues) of the similarity matrix is used to perform dimensionality reduction, to finally perform clustering (e.g. K-means) in the reduced-dimensions space. A detailed algorithm is given in Algorithm 4.

4.1.2 Supervised learning

Supervised learning is a field of machine learning which aims at learning a predictive model linking input data to a target signal. Here we are interested in the specific task of classification, where the signal to be learned consists in class labels, contrary to regression which consists in learning a continuous signal. In this case, the predictive model is also called a classifier.

The usual workflow of supervised learning consists in a first training phase, where the model is learned, followed by a prediction phase, where new data points will be assigned to a class using the predictive model.

We consider that the data points, noted $\mathbf{x}$, belong to a p dimensional space. Their corresponding class labels will be noted y. The corresponding class labels belong to a

Algorithm 4 Spectral clustering

Input: clusters number K, $p \times n$ data matrix $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots n}$
Output: cluster set $\mathbf{C} = \{C_i | i = 1 \ldots K\}$
Compute the $n \times n$ affinity matrix $\mathbf{A}$ such that:

$$\mathbf{A}_{i,j} = e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}} \text{ for } i \neq j, \text{ and } \mathbf{A}_{i,i} = 0$$

Define the diagonal matrix $\mathbf{D}$, such that $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$, and compute the graph Laplacian matrix:

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

Compute the K first eigenvectors of $\mathbf{L}$, $u_1, \ldots, u_K$ and form the $n \times K$ matrix $\mathbf{U}$ whose columns are the eigenvectors $u_1, \ldots, u_K$
Form the matrix $\mathbf{V}$ from $\mathbf{U}$ by re-normalizing its rows:

$$\mathbf{V}_{i,j} = \frac{\mathbf{U}_{i,j}}{\sum_j (\mathbf{U}_{i,j}^2)^{1/2}}$$

Considering each row of $\mathbf{V}$ as a point in a K-dimensional space, cluster them into K clusters $\mathbf{C}' = \{C'_i | i = 1 \ldots K\}$ using the K-means algorithm
Assign a point $\mathbf{x}_i$ to a cluster C_j if and only the i-th row of $\mathbf{V}$ was assigned to the cluster C'_j

limited set of integer values S_c. A training set D_T contains n pair of input point / label:

$$D_T = \{(\mathbf{x}_i, y_i) \text{ s.t. } \mathbf{x}_i \in \mathbb{R}^p, y_i \in S_c, i = 1 \ldots n\} \tag{4.4}$$

In this section, we present two popular methods for supervised learning: Support Vector Machine (SVM) and decision trees. Note that we are only interested in presenting the classifier models associated with these two methods, and the algorithm using these models to predict the class of new data points. However, we will not explore the algorithms used to learn these classifiers.

4.1.2.1 Support Vector Machine

The Support Vector Machine (SVM) was introduced in [31] for binary classification.

Linear SVM Considering that the training data are labeled into two classes, $S_c = \{-1, 1\}$, the SVM aims at constructing an hyperplane which will separate the two classes with a maximum margin (see Fig. 4.1).

An hyperplane is formally described by the set of points $\mathbf{x}$ satisfying the following equation:

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \tag{4.5}$$

where $\mathbf{w}$ is a vector orthogonal to the hyperplane and b is a scalar.

The SVM searches for the maximum-margin hyperplane described by $(\mathbf{w}, b)$, which satisfies the following binary classification constraints:

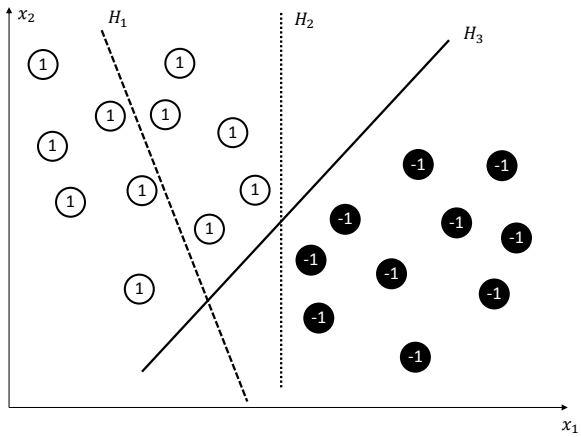


Figure 4.1: Binary labeled data points in a 2D space separated by hyperplanes. The hyperplane H_1 can not separate the two classes. Both H_2 and H_3 can separate the classes, but H_3 separates them with the maximum margin.

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \text{ if } y_i = 1, \forall(\mathbf{x}_i, y_i) \in D_T \\ \mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \text{ if } y_i = -1, \forall(\mathbf{x}_i, y_i) \in D_T \end{aligned} \tag{4.6}$$

which can be summarized as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall(\mathbf{x}_i, y_i) \in D_T \tag{4.7}$$

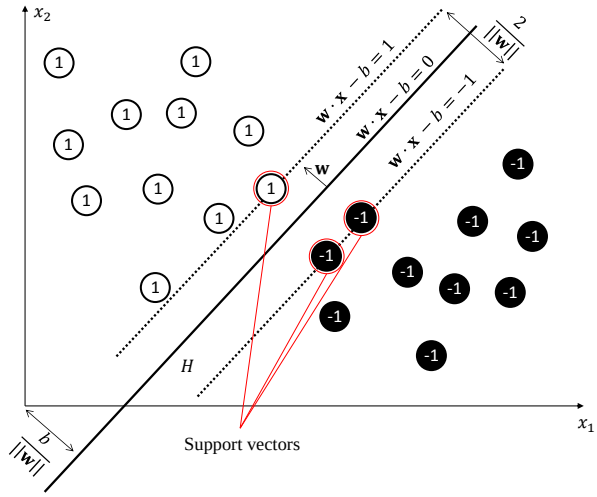


Figure 4.2: Example of linear SVM.

This condition was modified into the so-called soft margin method, which allows for

mislabeled data. The method introduces non-negative slack variables ξ_i which measures the degree of misclassification of a point $\mathbf{x}_i$:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1 - \xi_i, \forall(\mathbf{x}_i, y_i) \in D_T \tag{4.8}$$

The solutions for this optimization problem are well-known and will not be discussed here. Eventually, the vector $\mathbf{w}$ can be expressed as a combination of its support vectors:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{4.9}$$

where the $\mathbf{x}_i$ corresponding to non-zero α_i are the support vector. The support vectors lie on the margin and satisfy $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) = 1$. The scalar b can then be obtained from a support vector as:

$$b = \mathbf{w} \cdot \mathbf{x}_i - y_i \tag{4.10}$$

Once such hyperplane has been found, a new data point $\mathbf{x}'$ is assigned a label y' using the following rule:

$$\begin{cases} y' = 1 & \text{if } \mathbf{w} \cdot \mathbf{x}' - b \geq 1 \\ y' = -1 & \text{if } \mathbf{w} \cdot \mathbf{x}' - b \leq -1 \end{cases} \tag{4.11}$$

Alternatively, the computation of the dot product $\mathbf{w} \cdot \mathbf{x}'$ can be expressed using the support vectors:

$$\mathbf{w} \cdot \mathbf{x}' = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x}' \tag{4.12}$$

Non-linear SVM For many problems, the classes are not necessarily linearly separable in the input space. In [32], the authors propose a non-linear version of SVM based on the kernel trick.

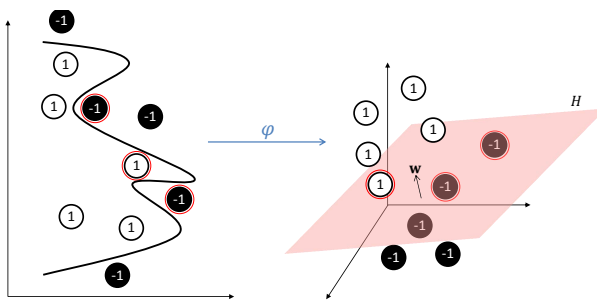


Figure 4.3: Example of non-linear SVM.

Conceptually, the finite-dimensional input space is mapped into a higher-dimensional feature space using a transform φ. In the feature space, the classes are expected to be

linearly separable, and thus the SVM can be applied efficiently (see Fig. 4.3). Using the kernel trick, no computation is explicitly performed in the feature space. The kernel k is related to the transform φ by the following equation:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j) \tag{4.13}$$

In other words, every dot product in the feature space is replace by a non-linear kernel function. Widely used kernels include the polynomial kernel, $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$, sigmoid functions $k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \cdot \mathbf{x}_j + c)$, or Gaussian radial basis functions $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2})$.

The vector $\mathbf{w}$ describing the hyperplane in the feature space is expressed as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \varphi(\mathbf{x}_i) \tag{4.14}$$

corresponding to equation 4.9. However, $\mathbf{w}$ is never computed explicitly, thanks to the kernel trick. In fact, a new point $\mathbf{x}'$ is classified by computing $\mathbf{w} \cdot \varphi(\mathbf{x}')$, which can be expressed as follow:

$$\mathbf{w} \cdot \varphi(\mathbf{x}') = \sum_{i=1}^{n} \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}') = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}') \tag{4.15}$$

Similarly, the scalar b can be computed as:

$$b = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j) - y_j \tag{4.16}$$

Extension to multiclass Although SVM was initially designed for binary classification, it can be extended to the multiclass problem. The main approach consist in reducing the multiclass problem in multiple binary classification problems. In the "one-vs-all' approach, binary SVM classifiers are built between one class and the all the remaining classes. In the "one-vs-one' approach, a binary SVM classifier is built between every pair of classes.

4.1.2.2 Decision Tree

Decision trees are methods which build prediction models from data [33]. The models are obtained by recursively partitioning the training data space and fitting a simple prediction model within each partition [89]. This partitioning can be represented as a tree, called a decision tree. In this section, we focus on describing the use of decision trees for classification problems, however, we will not discuss the well known algorithm for obtaining such trees.

Here, we will only consider binary tree for classification, *i.e.* for each node of the tree, a binary condition is evaluated on a component of the input vector. More precisely,

at a node i, the component c^i of the input vector is compared to some threshold value t^i, and we can express the condition as:

$$x_{c^i} < t^i \tag{4.17}$$

Each leaf of the tree represents a label.

To classify a new input point $\mathbf{x}'$, the conditions at the nodes are recursively evaluated, going from a node to its left child if the condition is validated, to the right one otherwise. The recursive process is repeated until a path is defined from the root to a leaf. The label associated with the leaf is then assigned to the input point.

In the example of Fig. 4.4, the training points are partitioned into two classes, labeled 1 and -1. Although this classification example can not be solved with a linear SVM, it can be simply modeled by the classification tree shown in Fig. 4.5. The new un-labeled point shown as a red question mark in Fig. 4.4 defines the path shown in red in Fig. 4.5, and is assigned the label -1.

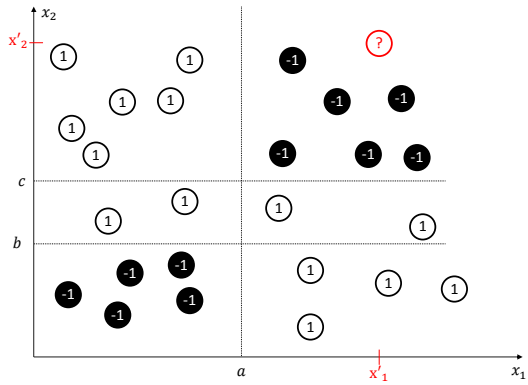


Figure 4.4: Binary labeled training data points in a 2D space, which can not be separated by an hyperplane.

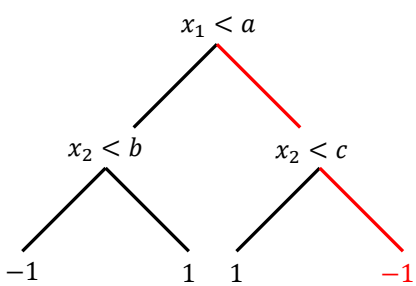


Figure 4.5: Classification tree corresponding to the training data shown in Fig. 4.4. The branches shown in red represent the path used to classify the points $(\mathbf{x}'_1, \mathbf{x}'_2)$ shown in Fig. 4.4.

Bagging. Bagging is the contraction of Bootstrap Aggregating, an ensemble method designed to improve the performances of machine learning algorithms [34]. This meta algorithm is mainly used to reduce the overfitting issue of decision trees, but can be applied to any classification method.

The main idea of ensemble methods is to create several training subset from the training data. Formally, m training subsets D_T^i of size n' are generated by sampling uniformly and with replacement (bootstrap sample) from the main training set D_T. It is then possible to learn a classifier model from each training subset. When classifying a new test point, m predicted labels are thus obtain. The label which is the more represented among these m labels is assigned to the test point.

4.2 Clustering-based linear mapping learning for quantization noise removal

In this section, we propose a quantization noise removal method using multiple linear mappings, learned at the encoder side between the decoded sequence and the source sequence. As in the previous chapter, we rely on the grouping of the coded/decoded patches. Here we choose to perform clustering, and a linear mapping is then learned for each cluster.

4.2.1 Main idea and preliminary results with additive noise

To demonstrate the potential of the proposed approach in terms of de-noising, we first tested it on images corrupted with Additive White Gaussian Noise (AWGN) or Signal Dependent Noise (SDN) (see section 3.1.2).

We briefly describe our method below, which will be more formally explained in the following section. The noisy image is divided into non-overlapping blocks of size $M \times N$. In a first step, the blocks are clustered into K partitions using the K-means algorithm (see section 4.1.1.1). For each cluster, a linear mapping is learned between the noisy blocks and the corresponding source blocks (see Eqs. 3.30 and 3.31 in section 3.3.4.3). The linear mapping is then applied on all the noisy blocks of the cluster to obtain the de-noised ones.

The tests were performed on single images (first frame of the test sequences described in Table 4.5), using 4×4 non-overlapping blocks and $K = 10$ clusters. For the AWGN, four values of the standard deviation were used, $\sigma = 10, 20, 30, 40$. For the SDN, four sets of noise parameters were also used, $\gamma = 0.5, \sigma_u = 1.5, \sigma_w = 10$, $\gamma = 0.5, \sigma_u = 3, \sigma_w = 10$, $\gamma = 0.6, \sigma_u = 1.5, \sigma_w = 5$, $\gamma = 0.7, \sigma_u = 0.5, \sigma_w = 5$. The proposed method is compared to the NLM and BM3D algorithms.

Table 4.1: De-noising performances of K-means ($K = 10$) clustering-based linear mapping learning compared to NLM and BM3D for AWGN

	PSNR (dB)			
σ	Noisy	K-means LM	NLM	BM3D
10	28.1393	33.5882	34.1586	36.3768
20	22.2159	29.8034	30.0121	33.2105
30	18.8622	27.6393	27.0592	31.2358
40	16.5635	26.0502	24.8694	29.5960

In Tables 4.1 and 4.2, we give the results for the AWGN and the SDN respectively, averaged over all the test images. The complete results are given in Tables C.1 and C.2, in annex (section C.1.1). We can see that our method is competitive with the NLM, even though it can not reach the performances of BM3D. Especially for high noise level of AWGN or SDN, our methods clearly outperforms NLM. To the best of our knowledge, methods using linear mapping have not been studied for de-noising. Given these promising preliminary results, we propose in the next section a scheme to address the more challenging task of quantization noise removal using this technique. In

Table 4.2: De-noising performances of K-means ($K = 10$) clustering-based linear mapping learning compared to NLM and BM3D for SDN

			PSNR (dB)			
γ	σ_u	σ_w	Noisy	K-means LM	NLM	BM3D
0.5	1.5	10	23.1242	30.3219	25.7377	33.2564
0.5	3	10	18.4487	27.3471	22.6082	30.0192
0.6	1.5	5	20.5998	28.7132	27.8899	30.9156
0.57	0.5	5	25.1191	31.5671	32.5274	33.7646

addition, the proposed following application needs to take into account the transmission of the linear mappings to the decoder, which was not considered here.

4.2.2 Proposed scheme for quantization noise removal

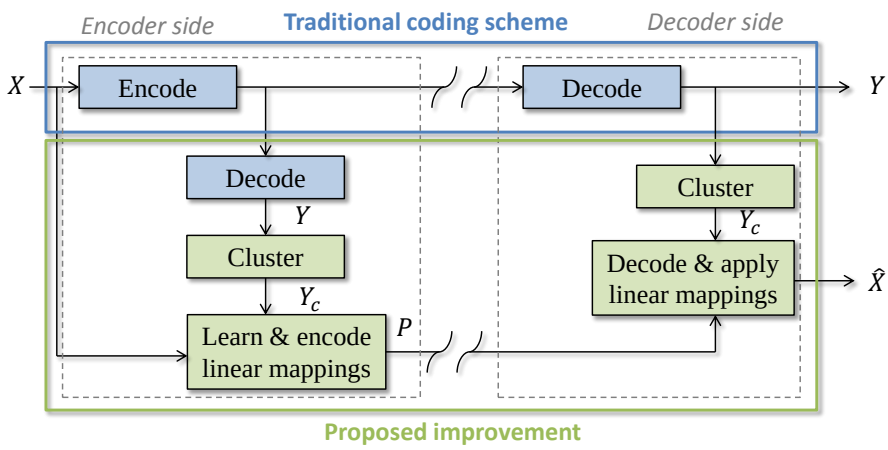


Figure 4.6: Proposed de-noising scheme for coding noise based on clustering and linear mappings.

In this section we describe the proposed compression scheme improvement using clustering-based linear mapping learning for de-noising. We note $\mathbf{M}_d$ and $\mathbf{M}_s$ the matrices containing in their columns the vectorized $M \times N$ patches extracted from the coded/decoded sequence Y and the source sequence X respectively.

The main idea of the proposed method is represented in Fig. 4.6, and consists in the following steps:

- At the encoder side:
  - cluster the coded/decoded patches $\mathbf{M}_d$ (see section 4.2)
  - for each cluster c, learn a linear mapping $\mathbf{P}_c$ between the coded/decoded sequence patches $\mathbf{M}_d^c$ and the source patches $\mathbf{M}_s^c$:

$$\mathbf{P}_c = \mathbf{M}_s^c \mathbf{M}_d^{cT} (\mathbf{M}_d^c \mathbf{M}_d^{cT})^{-1} \tag{4.18}$$

- encode the corresponding linear mappings (in matrix form) and transmit them to the decoder (see section 4.2.4)

- At the decoder side:
  - decode the linear mappings
  - cluster the coded/decoded sequence $\mathbf{M}_d$, as it is performed at the encoder side
  - for each cluster c, apply the corresponding linear mapping $\mathbf{P}_c$ to the decoded sequence patches $\mathbf{M}_d^c$ to obtain the de-noised patches $\mathbf{M}_r^c$:

$$\mathbf{M}_r^c = \mathbf{P}_c \mathbf{M}_d^c \tag{4.19}$$

In the following sections, we first analyze the clustering process in section 4.2.3. We then detail the linear mappings encoding in section 4.2.4. In section 4.2.5 we propose improvements for the algorithm. Finally we present the results in section 4.2.6.

4.2.3 Clustering

To perform the clustering, we divide the frames in $M \times N$ non-overlapping patches. Empirical results show that the de-noising performances are better for small patches. Thus, 4×4 patches perform better than 8×8 patches, as their local structures are better estimated. However, the patches should be big enough to contain structures. For example, 2×2 patches performances are worst than 4×4 patches. In practice, 4×4 patches were used to obtain our results.

4.2.3.1 Choice of a clustering algorithm

Any clustering algorithm is eligible for the proposed method, as long as it can be performed similarly at the encoder and the decoder side. We choose the popular K-means algorithm, described in section 4.1.1.1. In order to obtain the same clustering at the encoder and the decoder sides, the same initialization should be used on both sides (which excludes the widely used random initialization). The choice of a number of cluster K is discussed in details in section 4.2.3.2.

Note that other clustering methods can be used, such as Gaussian Mixture Models (GMM), described in section 4.1.1.2, or the more recent spectral clustering, described in 4.1.1.3. However, tests showed that even if the clusters obtained are not exactly the same, the final de-noising performances are similar. In fact, all these clustering methods tend to regroup in each cluster patches which are close in terms of Euclidean distance.

To illustrate this behavior, the three clustering methods were performed on a 200×200 crop of the first frame of the Kimono sequence encoded with HEVC at $QP = 37$. The image was divided in 4×4 patches, and K was set to 10. The source image and the compressed version along with the different clusters obtained are shown in Fig. 4.7. The clustering results obtained are *overall* similar, with obvious differences. However, the de-noising performances are in the end very similar, as the de-noised PSNR are

38.4586 dB, 38.3860 dB and 38.4297 dB for the K-means, GMM and spectral clustering respectively. The K-means algorithm was thus preferred for its simplicity and reduced complexity compared to GMM or spectral clustering.

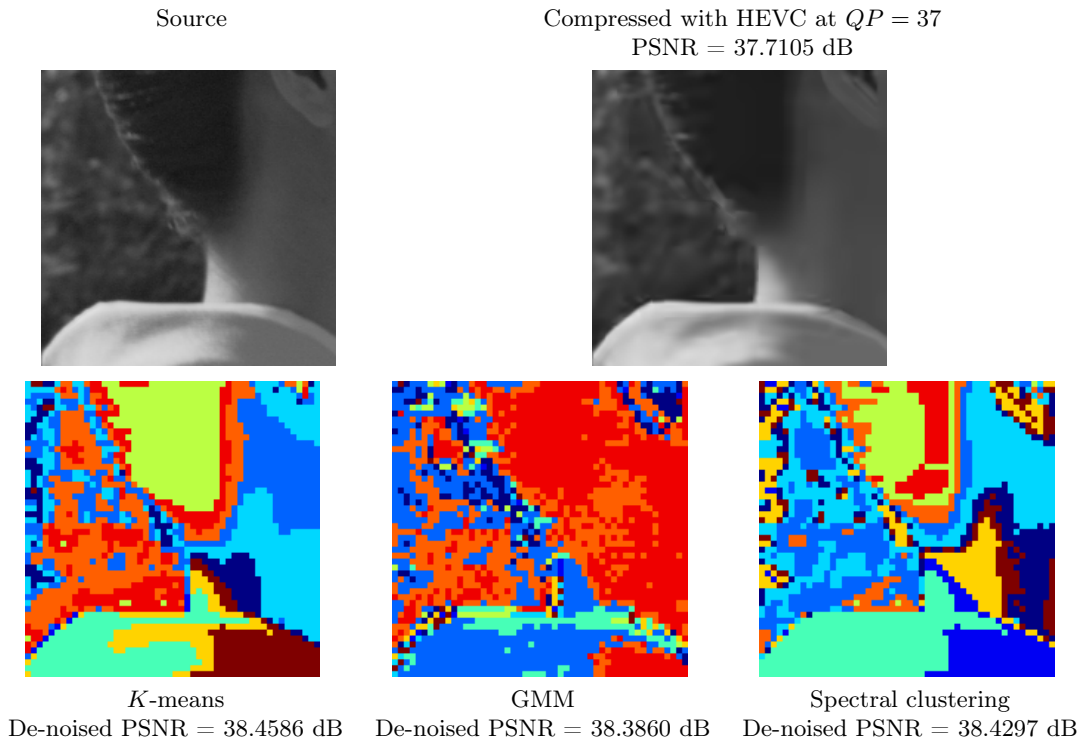


Figure 4.7: 200×200 crop of the first frame of Kimono, and the comparison between the different clustering methods with $K = 10$. (Each color corresponds to a cluster label.)

4.2.3.2 Choice of a number of clusters K

Generally speaking, the parameters should not only be tuned according to the de-noising performance, but the overall RD performance. Concerning the patch size, no compromise needs to be made, as small patches both improve the de-noising performances while reducing the linear mapping rate cost (see section 4.2.4). On the contrary, the number of clusters K should be chosen carefully. In fact, increasing K improves the de-noising performances, but also the number of linear mappings to send to the decoder, and their associated rate cost. Ultimately, learning one mapping per patch can produce almost perfect de-noising, but then a $M^2 \times N^2$ linear mapping should be sent to the decoder for each $M \times N$ patch. This is obviously not acceptable in terms of RD performance. Experimental results show that 10 is a good order of magnitude for K. Such value also limits the complexity of the clustering.

The RD performances presented in section 4.2.6.1 were obtained with a fixed K set

to 10. The corresponding results presented in Table 4.11 show that this value might be too high at low bit-rates, as it increases the bit-rate for a small increase in PSNR. For example, the bit-rate allocated to the linear mappings for the Kimono and Tennis sequences at $QP = 37$ accounts for about 7% of the whole bit-rate. On the contrary, at high bit-rates, the cost of the linear mappings can become negligible, e.g. for the Terrace and Ducks sequence, where it only accounts for less than 0.1% of the whole bit-rate. In this case, K could be increased to improve the de-noising performances.

Manually tuning the parameter K for each sequence and each QP to obtain the best performance is a tedious task. To address this problem, we present in this section an adaptive method for selecting K based on an RDO criterion.

Instead of explicitly partitioning the data into K clusters, we use a recursive binary partitioning. The procedure is detailed below:

- At initialization, the full data set is considered as a cluster.
- Each cluster is then recursively split in 2 clusters if the RDO criterion (explained below) is satisfied, creating a binary tree structure.
- The procedure stops when no RDO criterion is satisfied, or a certain maximum tree depth is reached.

The RDO criterion used to decide to further split or not a cluster is designed to balance the de-noising performances and the linear mappings rate cost. Given a cluster C_n and its corresponding mapping $\mathbf{P}_n$, we can allocate a RD cost to it computed as:

$$RD_n = D_n + \lambda R_n \tag{4.20}$$

The distortion D_n is computed as the Sum of Squared Error (SSE) between the patches of C_n de-noised with $\mathbf{P}_n$ and the corresponding source patches. The rate R_n is estimated as the number of bits of the encoded mapping $\mathbf{P}_n$ (see section 4.2.4). The lagrangian parameter is computed as in the test model of HEVC:

$$\lambda = QP_{factor} \times 2^{((QP-12)/3)} \tag{4.21}$$

where QP is the quantization parameter and QP_{factor} is an adjustment factor, set to 1 in our experiments. Thus, to decide if C_n is further divided in two clusters C_{n+1} and C_{n+2}, we just verify the condition $RD_{n+1} + RD_{n+2} < RD_n$. Note that this method is independent from the clustering algorithm, and could be performed with any clustering algorithm presented in section 4.2.3. An example of binary tree structure obtained after the recursive partitioning is given in Fig. 4.8.

To perform the exact same partitioning at the decoder, we then need to transmit the binary tree structure. If we note the maximum binary tree depth d, then the maximum number of bits needed to describe the tree structure is:

$$R_{max}^{BT} = \sum_{i=0}^{d-1} 2^i \tag{4.22}$$

In our experiments we set $d = 4$, so the maximum number of clusters is $K_{max} = 16$ and $R^{BT}_{max} = 15$ bits. Thus, the bit-rate allocated to the binary tree structure is negligible compared to the bit-rate of a full GOP.

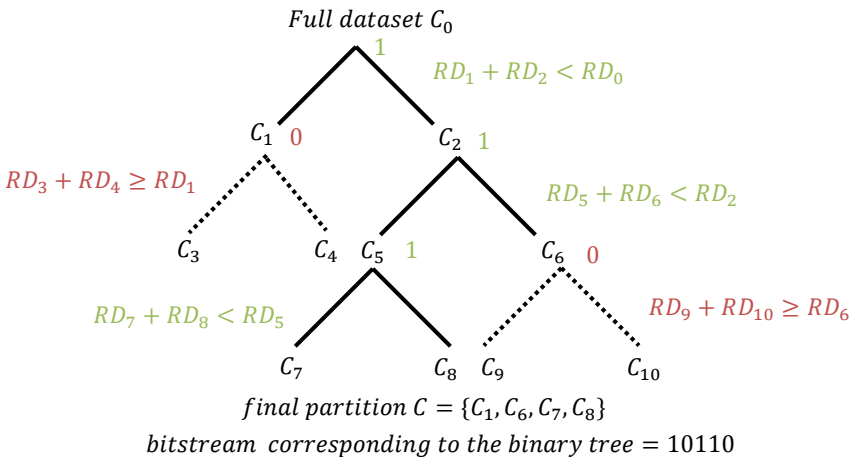


Figure 4.8: Example of a cluster partition based on a binary tree structure and an RDO criterion. The clusters are recursively split to reach 4 clusters. The tree structure is described by only 5 bits.

The full results obtained with the adaptive RDO selection of K are detailed in section 4.2.6.2, and show its efficiency in terms of RD performance.

4.2.4 Linear mappings encoding

Once the clustering and linear mapping learning steps are performed, the linear mappings need to be encoded and sent to the decoder.

For $M \times N$ patches, the size of a linear mapping matrix $\mathbf{P}$ is $(M*N)\times(M*N)$. Thus, in order to limit the rate cost of the linear mappings encoding, the size of the patches should be small, which is also profitable for the de-noising performances. Typically, in section 4.2.6, 4×4 patches are used. As explained in section 4.2.3.2, the number of clusters K should be carefully selected, as increasing K improves the de-noising performances but also increases the rate associated to the linear mappings. When using the method described in section 4.2.3.2, the K value ranges from 2 to 16.

For such values, the linear mapping matrices usually exhibit strong redundancies, both inside each matrix and between the different matrices (e.g. in Figs. 4.9 and 4.10). Therefore we choose to stack the linear mapping matrices in a 3-dimensional group, and encode them as an image sequence using a video codec.

The linear mappings matrices consist originally in floating points values, thus they need to be quantized to integer values before encoding. To avoid a loss in terms of de-noising performance due to the quantization, the matrices are quantized on 16 bits integer values (e.g. in Table 4.3). To illustrate this claim, we performed the method on the first frame of the Kimono sequence encoded at $QP = 22$ and $QP = 37$ and using the

Table 4.3: Impact of the quantization of the linear mapping matrices on the de-noising performances. The de-noising is performed on the first frame of the Kimono sequence, clustered with the K-means algorithm, $K = 10$.

QP	coded/ decoded PSNR (dB)	de-noised PSNR (dB)					Linear mappings rate (in bits) for quantization on 16 bits
		no quantization	quantized on 8 bits	quantized on 12 bits	quantized on 16 bits		
22	43.2763	43.3768	43.1790	43.3761	43.3768		37824
37	37.9512	38.0677	38.0173	38.0674	38.0677		40136

K-means algorithm ($K = 10$). In Table 4.3, we show the PSNR values of the de-noised frame with or without quantization of the linear mappings. We can see that when the matrices are quantized on 16 bits, there are no effects on the de-noising performances.

The encoding of the matrices is then performed with the Range Extension (RExt) of HEVC [90][91], which supports 16 bits input. The quantization parameter of HEVC was chosen very low ($QP = -30$) to limit the loss on the de-noising performances. Finally, in order to recover the floating point matrices, the minimum and maximum values of the original floating points matrices need to be sent to the decoder without loss. In Table 4.3 we show the rate obtained after encoding of the matrices.

The rate of the linear mappings is slightly higher when the frame is encoded at $QP = 37$ because the matrices are less correlated than at $QP = 22$, as it can be seen in Figs. 4.9 and 4.10.

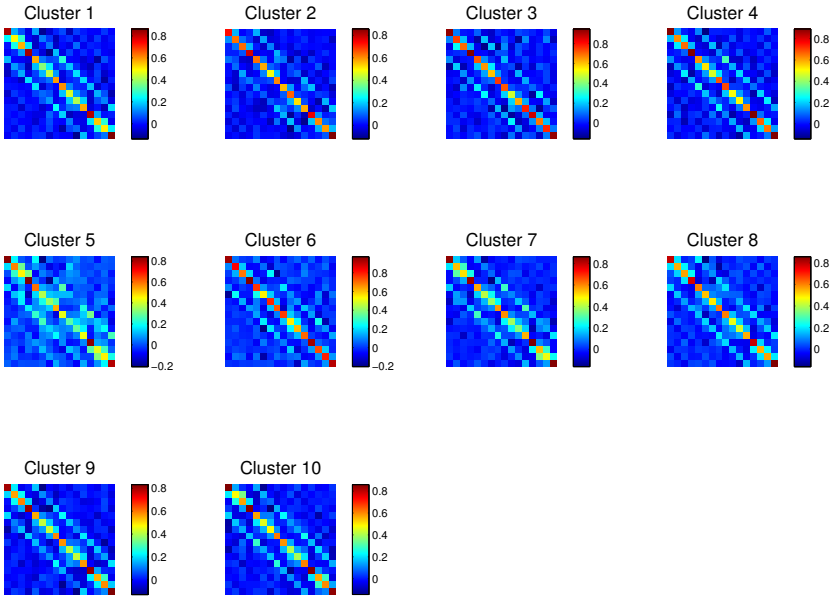


Figure 4.9: Linear mapping matrices obtained on the first frame of the Kimono sequence encoded with HEVC at QP=22. The clusters were obtained with the K-means algorithm, with $K = 10$. With this configuration, the source and coded/decoded patches are similar, thus the energy of the mappings is mainly concentrated on the diagonal.

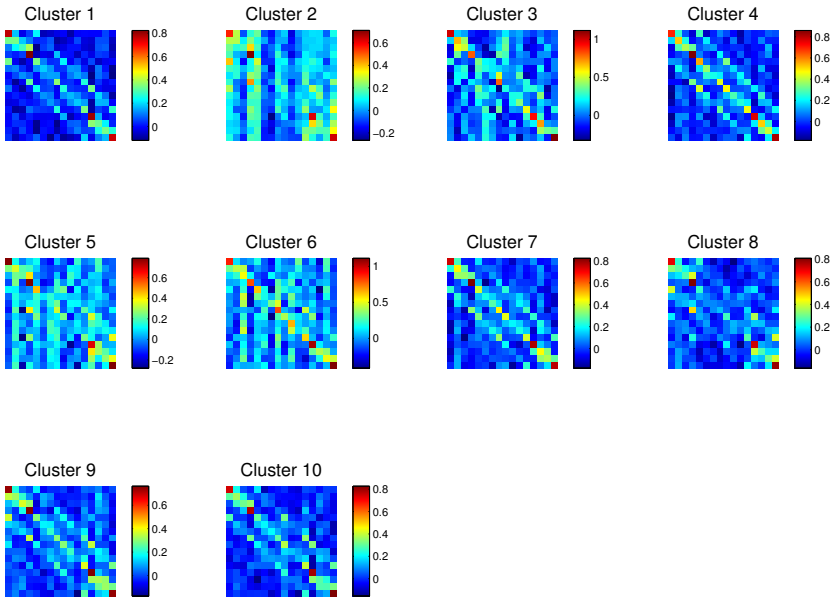


Figure 4.10: Linear mapping matrices obtained on the first frame of the Kimono sequence encoded with HEVC at QP=37. The clusters were obtained with the K-means algorithm, with $K = 10$. With this configuration, the coded/decoded patches are more degraded than in Fig. 4.9, thus the mappings are less structured.

4.2.5 Two steps scheme with first pass blind de-noising

Several methods in de-noising use a first de-noising pass in order to improve their performances. First, the BM3D algorithm itself is a two steps algorithm [29], where the first de-noising output, obtained from the sparse 3D transform spectrum shrinkage, is used as input for a Wiener filter. More recently, [92] proposed a two steps algorithm, where the noisy input is first de-noised with the NL-Bayes algorithm [93]. This first pass de-noised image is used as a clean guide for a dual-domain de-noising algorithm [94]. In [67], the authors tackle the less studied Signal Dependent Noise (SDN) removal problem. Their clustering-based method estimates the noise level for each cluster (which is supposed constant). Each cluster is then de-noised by applying an AWGN de-noising algorithm with its corresponding noise level. The performance is improved by first applying a BM3D algorithm using the average noise level.

In this section, we propose a similar improvement of our method by first applying a "blind" de-noising algorithm. Our method is then applied on this first de-noised sequence. This idea is represented in Fig. 4.11. At the encoder side the linear mapping learning is made between the source and this first estimate. Thus, at the decoder side, the same de-noising has to be performed. We qualify this first pass de-noising as "blind" because no parameters should be transmitted between the encoder and the decoder side, so that there is no bit-rate increase. In practice, some parameters that have a negligible rate cost are send.

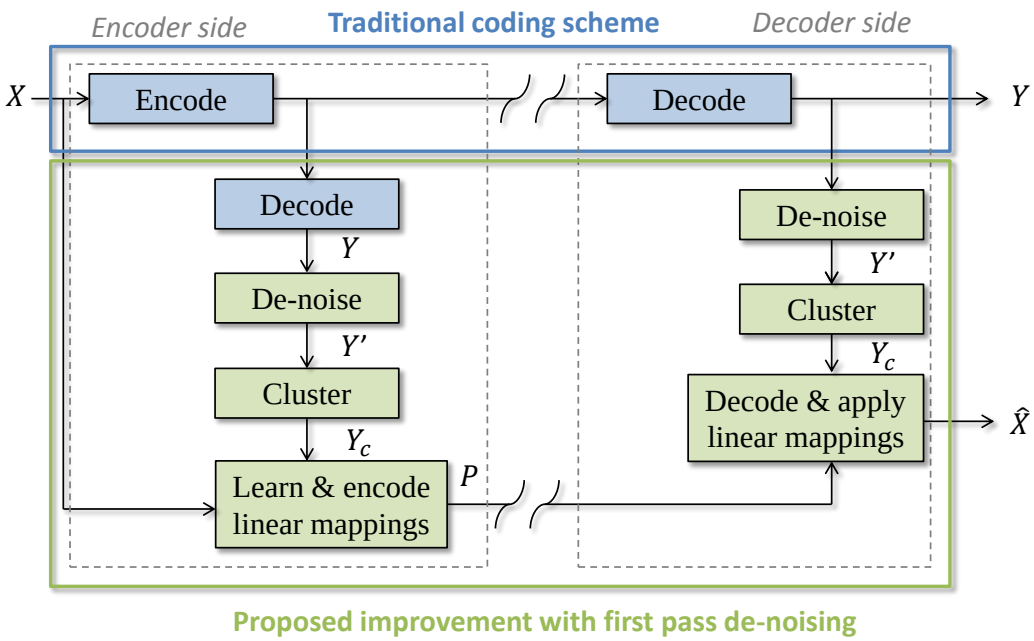


Figure 4.11: Proposed de-noising scheme for coding noise based on clustering and linear mappings with first pass "blind" de-noising.

The first pass de-noising is performed using the Video BM3D (VBM3D) algorithm [78]. The noise standard deviation σ_n is estimated as the square root of the MSE computed on a GOP. This parameter can only be computed at the encoder side and has to be sent to the decoder. However, its rate cost is negligible compare to the bit-rate of the whole GOP. The other VBM3D parameters, shown in Table 4.4, are fixed and considered known at both encoder and decoder side. On top of this first de-noising pass, we apply our method using K-means clustering, with the adaptive K selection method described in section 4.2.3.2.

The results given in section 4.2.6.3 show that a significant improvement is brought by the first pass de-noising compared to the proposed clustering-based method. This also proves that our de-noising method relying on linear mapping can be complementary to the de-noising method relying on shrinkage in a transform domain.

Table 4.4: Fixed parameters for the BM3D algorithm. See section 3.1.2.4 for details on the parameters.

Parameters for Step 1 (HT)	T_{HT}	2D-Bior 1.5 + 1D-Haar
	λ	$2.7\sigma_n$
	ε_{BM}^{HT}	3000
Parameters for Step 2 ($Wien$)	T_{Wien}	2D-DCT + 1D-Haar
	ε_{BM}^{Wien}	1500
Common parameters	K	8
	N	8
	N_{SW}	7
	s	8
	N_{FR}	4

4.2.6 Simulations and results

The sequences used in the experiment are presented in Table 4.5. The test sequences are processed per Group Of Pictures (GOP), and a GOP contains a number of frames equal to the frame rate (*i.e.*, we have one GOP per second).

Table 4.5: Test sequences

Sequence	**Resolution** $W \times H$	**Frames count** T	**Frame rate**
City	1280×720	600	60
Park Scene	1920×1080	240	24
Tennis	1920×1080	240	24
Kimono	1920×1080	240	24
Cactus	1920×1080	500	50
Terrace	1920×1080	600	60
Basket	1920×1080	500	50
Ducks	1920×1080	500	50
People On Street	2560×1600	150	30
Traffic	2560×1600	150	30

The sequences are encoded with the HEVC test model HM (ver 15.0) [82] using the Main profile in Random Access, with 4 values for the Quantization Parameter, $QP = 22, 27, 32, 37$. For all the experiments, the linear mappings are encoded for each GOP as described in section 4.2.4 using the range extension of HEVC (HM 15.0 RExt 8.1) [91] with a fixed $QP = -30$.

4.2.6.1 RD performances using the K-means clustering

In this experiment, the GOPs are divided in non-overlapping patches of size 4×4, which are clustered using the K-means algorithm with $K = 10$ clusters. Rate-distortion performances for each GOP are presented in Table 4.6, computed using the Bjontegaard

metric [54] with respect to the sequences encoded with HEVC.

In average over all the sequences and all the GOPs, the bit-rate is reduced by 1.662 %. We can see that except for 3 sequences (ParkScene, Tennis, Kimono), we obtain a bit-rate saving of about 1-2 %, and we reach about 8 % for the Terrace sequence. For the fifth GOP of this sequence, we even reach about 12 % bit-rate reduction.

Table 4.6: RD performances per GOP of K-means clustering with $K = 10$ (Bjontegaard bit-rate gain with respect to HEVC)

GOP	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
1	-2.309	0.123	-0.060	-0.130	-1.157	-5.377	-0.828	-2.319	-2.544	-1.293	-1.589
2	-1.755	0.155	0.092	-0.132	-1.224	-7.143	-0.429	-1.810	-2.554	-1.231	-1.603
3	-1.958	0.091	0.182	-0.196	-0.882	-8.617	-0.870	-1.016	-2.540	-1.221	-1.703
4	-1.742	0.184	0.102	-0.243	-0.807	-11.690	-0.744	-1.164	-2.601	-1.177	-1.988
5	-0.961	0.301	0.104	-0.191	-1.055	-12.155	-0.649	-1.197	-2.584	-1.309	-1.97
6	-1.900	0.473	1.598	0.352	-1.198	-10.051	-0.433	-1.613			-1.597
7	-1.589	0.481	1.451	1.216	-1.111	-8.493	-0.505	-2.010			-1.320
8	-1.976	0.502	1.054	1.422	-1.343	-7.534	-0.709	-2.364			-1.368
9	-1.844	0.589	0.839	1.898	-1.028	-6.734	-0.742	-2.490			-1.189
10	-1.922	0.972	0.773	2.541	-0.914	-6.788	-0.486	-2.757			-1.073
Average	-1.806	0.381	0.080	0.426	-1.079	-8.282	-0.652	-1.877	-2.558	-1.257	-1.662

4.2.6.2 RD performances using binary tree RDO for K selection

In this section, the number of clusters K is adaptive and selected using the binary tree RDO method described in section 4.2.3.2. Experiments were performed using the K-means clustering and the same sequences and parameters as in section 4.2.6.1. RD performances are shown in Table 4.7. Comparison with performances using a fixed $K = 10$ presented in Table 4.6 show that we now obtain better performance for most of the sequences. We report in Table 4.8 the selected values of K, averaged over all the GOPs. We can see that the K values adapt to the sequence, reaching higher values for sequences with a higher resolution and/or frame rate. The K values depend on QP, and lower values are selected at low bit-rate (high QP value).

Table 4.7: RD performances per GOP of K-means clustering with RDO adaptive K selection (Bjontegaard bit-rate gain with respect to HEVC)

GOP	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
1	-2.763	-0.692	-0.900	-1.317	-1.125	-5.253	-1.104	-2.393	-2.570	-1.439	-1.956
2	-2.309	-0.672	-0.720	-1.348	-1.285	-6.830	-0.405	-1.819	-2.576	-1.387	-1.935
3	-2.447	-0.632	-0.775	-1.358	-0.892	-8.855	-0.977	-1.078	-2.589	-1.352	-2.096
4	-1.657	-0.536	-0.824	-1.308	-0.979	-11.579	-0.622	-1.245	-2.612	-1.351	-2.271
5	-1.844	-0.517	-0.717	-0.888	-1.112	-11.947	-0.761	-1.266	-2.631	-1.528	-2.321
6	-2.217	-0.476	-1.077	-0.853	-1.370	-9.900	-0.766	-1.707			-2.296
7	-1.898	-0.479	-1.907	-0.585	-1.248	-8.203	-0.679	-2.130			-2.141
8	-1.861	-0.396	-2.389	-0.582	-1.295	-7.233	-1.089	-2.469			-2.164
9	-2.515	-0.309	-1.773	-0.428	-1.188	-6.354	-0.880	-2.577			-2.003
10	-2.743	0.017	-2.518	0.286	-0.999	-6.471	-0.691	-2.836			-1.994
Average	-2.236	-0.472	-1.415	-0.938	-1.153	-8.083	-0.795	-1.957	-2.600	-1.425	-2.107

Table 4.8: K values for K-means with RDO adaptive K selection (Averaged over all GOPs)

QP	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
22	10.8	8.5	7.6	8.2	14.9	15.4	12.0	15.9	16.0	14.0	12.33
27	11.3	6.8	5.7	6.4	11.7	15.0	9.7	15.8	16.0	12.6	11.10
32	7.4	4.0	3.4	4.5	9.0	14.1	7.8	15.8	16.0	7.6	8.96
37	5.9	4.0	2.8	4.0	7.1	13.1	5.8	15.5	15.8	5.8	7.98

4.2.6.3 RD performances with first pass blind de-noising

Here the video is first de-noised with the VBM3D algorithm, with the parameters described in section 4.2.5. Results are given for 1 GOP. In Table 4.9, we see that substantial bit-rate reduction are achieved when combining the proposed method with the BM3D.

From Tables 4.8 and 4.10 we can see that the number of clusters selected is higher when using the first pass de-noising. In fact, in the RDO decision, the improvement brought by the first pass de-noising reduces the distortion, and thus allows for a higher bit-rate cost, *i.e.* more linear mappings can be sent to the decoder.

Table 4.9: RD performances of K-means clustering with RDO adaptive K selection and VBM3D de-noising (Bjontegaard bit-rate gain with respect to HEVC)

	K-means clustering	VBM3D	VBM3D and K-means clustering
City	-2.763	-1.498	-3.237
Park Scene	-0.692	-0.882	-1.072
Tennis	-0.900	-7.837	-7.766
Kimono	-1.317	-4.266	-4.317
Cactus	-1.125	-4.037	-6.188
Terrace	-5.253	-1.145	-7.240
Basket	-1.104	-2.977	-3.803
Ducks	-2.393	-1.602	-4.127
People On Street	-2.570	-7.881	-9.627
Traffic	-1.439	-4.326	-5.428
Average	-1.956	-3.645	-5.281

Table 4.10: K values for K-means with RDO adaptive K selection and VBM3D de-noising

QP	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
22	15.0	12.0	9.0	7.0	16.0	16.0	11.0	16.0	16.0	15.0	13.30
27	15.0	8.0	7.0	6.0	13.0	16.0	11.0	16.0	16.0	12.0	12.00
32	10.0	6.0	6.0	6.0	12.0	16.0	9.0	16.0	15.0	13.0	10.90
37	9.0	4.0	5.0	4.0	11.0	15.0	5.0	16.0	15.0	6.0	9.00

4.3 Optimal clustering for de-noising based on linear mappings

Results from the previous section show that the proposed method using K-means clustering achieves convincing bit-rate reduction compared to HEVC. We give in Table 4.11 the de-noising performances corresponding to the results using K-means clustering with $K = 10$ (see section 4.2.6.1). We can see that the PSNR improvement is unexpectedly limited. In this section, we first analyze the limitations of the K-means clustering for de-noising using linear mappings. Second, we propose a new clustering algorithm that optimizes the de-noising performances. Note that with the proposed clustering algorithm, we define an upper theoretical bound on the de-noising performances. However, its application as such in the proposed compression scheme is not straightforward, as it relies on the knowledge of the source patches.

Table 4.11: De-noising performances of K-means clustering with $K = 10$ (ΔPNSR)

QP	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
22	0.125	0.044	0.104	0.089	0.042	0.088	0.028	0.028	0.170	0.077	0.080
27	0.118	0.047	0.092	0.081	0.048	0.141	0.034	0.048	0.136	0.083	0.083
32	0.099	0.048	0.080	0.071	0.052	0.158	0.042	0.076	0.119	0.080	0.083
37	0.083	0.047	0.078	0.069	0.058	0.188	0.049	0.110	0.115	0.076	0.087

4.3.1 K-means limitations for linear mappings learning

First, we notice that the linear mapping used to de-noise a patch implicitly estimates the residue. Let x_s be a source patch, x_d the corresponding decoded patch and q the patch residue, we have the following relation:

$$x_s = x_d + q \tag{4.23}$$

We call $\mathbf{P}$ the linear mapping used to estimate the source patch:

$$\hat{x_s} = \mathbf{P}x_d \tag{4.24}$$

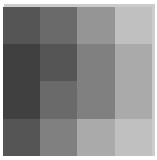
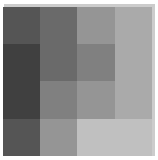
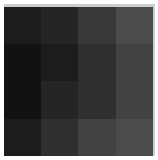
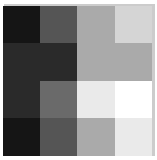
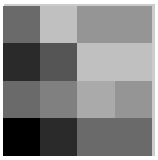
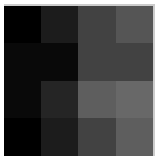
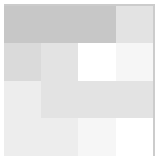
The residue can then be directly estimated using the linear mapping $\mathbf{P'} = \mathbf{P} - \mathbf{I}$, where $\mathbf{I}$ is the identity matrix with the same size as $\mathbf{P}$:

$$\mathbf{P'}x_d = \mathbf{P}x_d - x_d = \hat{x_s} - x_d = \hat{q} \tag{4.25}$$

This (trivial) observation suggests that the use of linear mappings for de-noising will perform better for clusters which contain similar patches residue. When using the

K-means algorithm, an implicit assumption is made that the similar decoded patches (belonging to the same cluster) correspond to similar source patches, and thus similar patches residue. However, this hypothesis does not hold for quantization noise, where similar decoded patches can correspond to different patches residue, and vice versa. For example, in Table 4.12, the two noisy patches on the left are similar, but their corresponding source patches (hence the patches residue) are different. On the contrary, the two dissimilar noisy patches on the right would be efficiently de-noised using the same linear mapping, because they correspond to similar patches residue.

Table 4.12: Left: Comparison of patches in the same cluster (using K-means). Although the decoded patches are similar, different linear mapping should be used to efficiently approximate their corresponding source patches. Right: Comparison of patches in the different clusters. Although the decoded patches are dissimilar, an unique linear mapping would efficiently de-noise the patches.

	Patch 1, cluster 1	Patch 2, cluster 1		Patch 1, cluster 1	Patch 2, cluster 2
Decoded					
Source					

Another way to represent the clusters is to visualize the patches as points in the Euclidean space. For this purpose we work on the first full frame of the Kimono sequence, encoded with HEVC at $QP = 37$. Fig. 4.12 shows the coded/decoded frame, which PSNR is 37.95 dB. The proposed de-noising method using the K-means algorithm ($K = 10$) performed on the 4×4 decoded patches reaches a PSNR of 38.07 dB, which corresponds to a PSNR gain of 0.12 dB. The patches are points in a 16 dimensions Euclidean space. In order to obtain a visual representation, we project the patches on 2 dimensions using the Linear Discriminant Analysis (LDA). The LDA was chosen over the popular Principal Component Analysis (PCA) because of its better ability to discriminate between the different classes [95].

In Table 4.13, we show different visual representations of the clusters obtained with the K-means algorithm performed in the decoded patches. In the first row, each patch of the first frame of Kimono is colored depending on its cluster label. In the second row, we show the 2D representation of the patch Euclidean space. On the left, each point represents a decoded patch projected with the LDA and colored depending on its cluster label. On the right, the same representation is shown, but *applied on the patches residue*. From these representations, we can verify that the K-means groups

similar patches (close in the Euclidean space) together. However, we can see that the clusters completely overlap in the residue space, which means that similar patches residue belong to different clusters, while different patches residue belong to the same cluster, as exhibited in Table 4.12.

Figure 4.12: First frame of the Kimono sequence encoded with HEVC at QP=37. PSNR = 37.95 dB.

To further explore this idea, we perform the K-means clustering (with $K = 10$) directly on the patches residue of the same previous decoded frame of Kimono. The proposed de-noising methods applied with these clusters reaches a PSNR of 40.68 dB, which corresponds to a PSNR gain of 2.73 dB. This result supports the idea that the linear mapping learning is much more efficient for clusters with homogeneous residue patches. In Table 4.14, we show the different visual representations of the clusters, as in Table 4.13. We can see that in the Euclidean spaces, we obtain the inverse representations of Table 4.13. The clusters are well separated in the residue space, but completely overlap in the decoded patches space. Thus, contrary to the K-means performed on the decoded patches, similar decoded patches belong to different clusters, while dissimilar decoded patches belong to the same cluster. This observation appears clearly in the representation of the first row of Table 4.14.

We can see that, in terms of de-noising, significant improvement can be reached compared to the K-means clustering of the decoded patches. Thus, in the next section, we propose a clustering algorithm designed to maximize the de-noising gains.

4.3.2 Optimal clustering for linear mapping learning

We proposed here a new clustering method denoted optimal clustering. For each cluster, we learn a linear mapping, and minimize the overall reconstruction error between the source patches and the coded/decoded patches de-noised with the linear mappings.

Formally, the problem corresponding to the optimal clustering is formulated as follows:

Table 4.13: Visual representations of the K-means clustering performed on the decoded patches, $K = 10$. (Each color corresponds to a cluster label.)

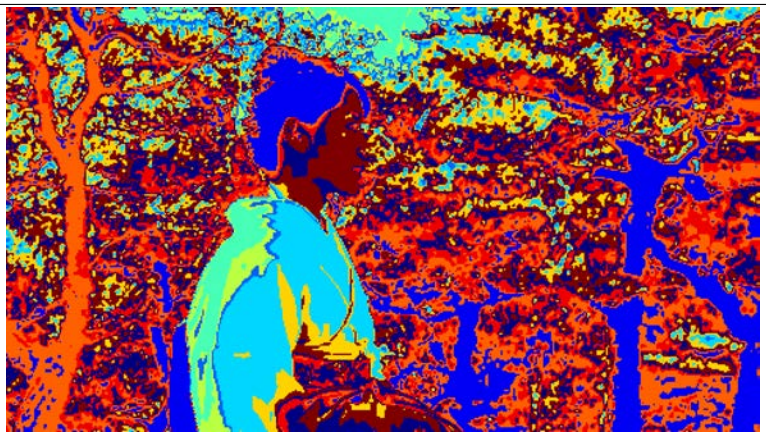
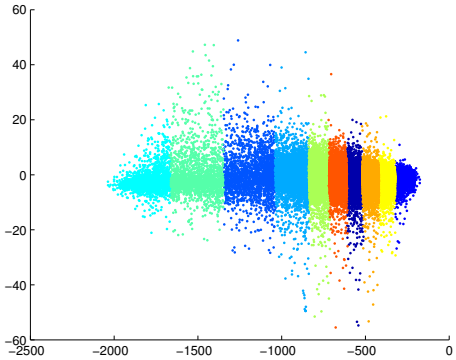
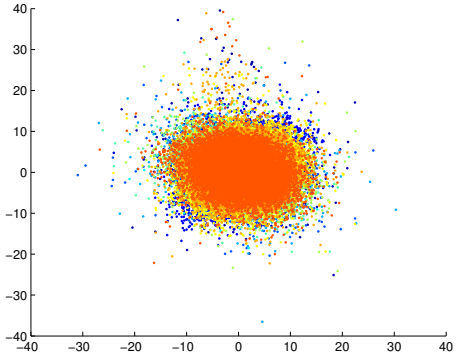
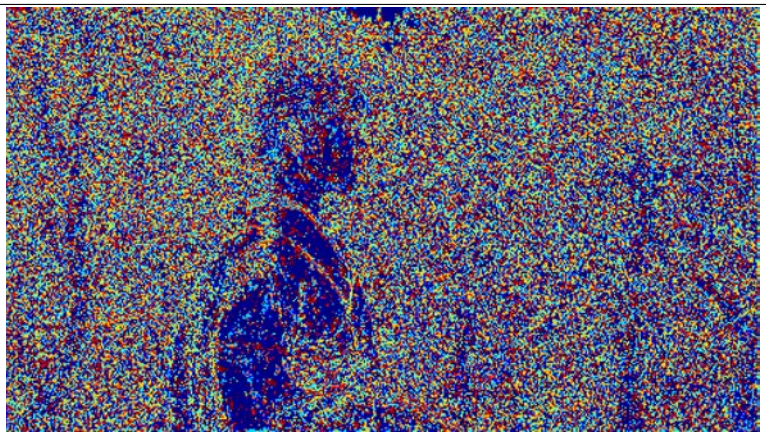
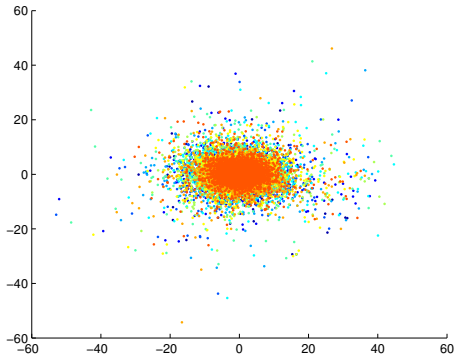
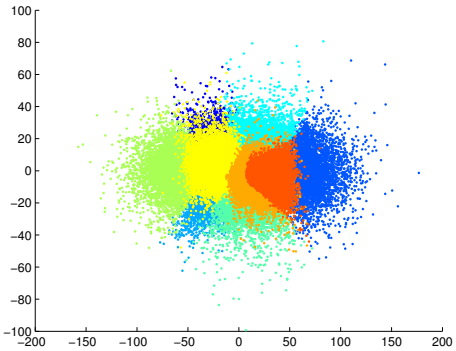
K-means clustering on decoded patches. De-noised PSNR = 38.07 dB.			
			
LDA projection of the coded/decoded patches.		LDA projection of the residue patches.	

Table 4.14: Visual representations of the K-means clustering performed on the residue patches, $K = 10$. (Each color corresponds to a cluster label.)

K-means clustering on residue patches. De-noised PSNR = 40.68 dB.			
			
LDA projection of the coded/decoded patches.		LDA projection of the residue patches.	

$$\min_{C} \sum_{i=1\ldots K} \sum_{x_d, x_s \in C_i} ||x_s - \mathbf{P}_i x_d||^2 \tag{4.26}$$

with $C = \{C_i\}_{i=1\ldots K}$. As for the K-means clustering, the problem of Eq. (4.26) can be solved using a greedy algorithm, described in Algorithm 5.

The initial set of linear mappings $\mathbf{P}_1^{(1)} \ldots \mathbf{P}_K^{(1)}$ is supposed known, e.g. obtained by first performing the K-means clustering on the coded/decoded patches and learning a linear mapping for each cluster.

Algorithm 5 Optimal clustering for linear mapping learning

Input: cluster number K, $d \times n$ coded/decoded patches matrix $\mathbf{M}_d = \{x_d\}$, corresponding source patches matrix $\mathbf{M}_s = \{x_s\}$

Output: cluster set $C = \{C_i, 1 \leq i \leq K\}$, corresponding mapping matrices $\mathbf{P} = \{\mathbf{P}_i, 1 \leq i \leq K\}$

Repeat the two following steps until convergence:

Assignment step:

$$\forall i = 1 \ldots K, C_i^{(t)} = \{x_d, x_s : ||x_s - \mathbf{P}_i^{(t)} x_d||^2 \leq ||x_s - \mathbf{P}_j^{(t)} x_d||^2 \forall j, 1 \leq j \leq K\}$$

Update step:

$$\forall i = 1 \ldots K, \mathbf{P}_i^{(t+1)} = \mathbf{M}_s^i {\mathbf{M}_d^i}^T (\mathbf{M}_d^i {\mathbf{M}_d^i}^T)^{-1}$$

In order to estimate the potential of the optimal clustering in terms of de-noising, preliminary experiments were performed on AWGN and SDN, with the same parameters as in 4.2.1. Here, the results are averaged over all test images. The complete results are given in Tables C.3 and C.4, in annex (section C.1.1). The results given in Tables 4.15 and 4.16 for AWGN and SDN respectively show a dramatic increase of the PSNR for the images de-noised with optimal clustering-based linear mapping learning compared to the K-means clustering (see corresponding Tables 4.1 and 4.2). We can see that the new clustering method is now competitive with BM3D for AWGN and even outperforms BM3D for SDN.

Table 4.15: De-noising performances of optimal clustering-based linear mapping learning and BM3D for AWGN

	PSNR (dB)		
σ	Noisy	Optimal clustering LM	BM3D
10	28.1393	36.2814	36.3768
20	22.2159	33.3756	33.2105
30	18.8622	31.8393	31.2358
40	16.5635	30.8744	29.5960

When performed on the first frame of the Kimono sequence encoded with HEVC at $QP = 37$, the de-noised frame PSNR reaches 41.26 dB, which corresponds to a PSNR gain of 3.31 dB. We can see that the optimal cluster does outperform the clustering of the residue patches in terms of de-noising performance. In Table 4.17, we show the different visual representations of the clusters, as in Tables 4.13 and 4.14. Although it

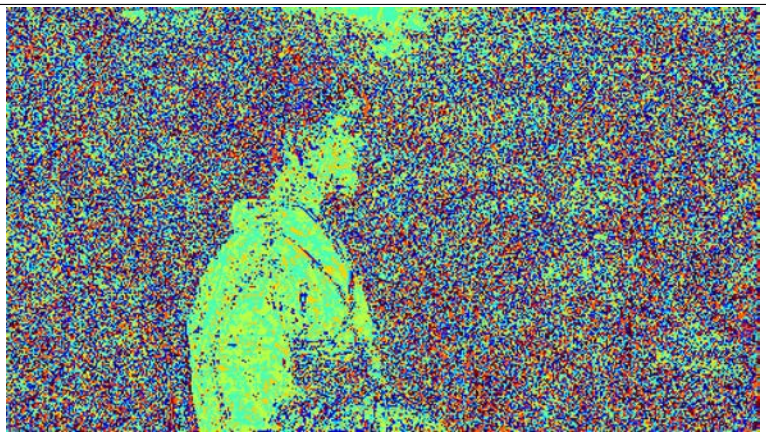
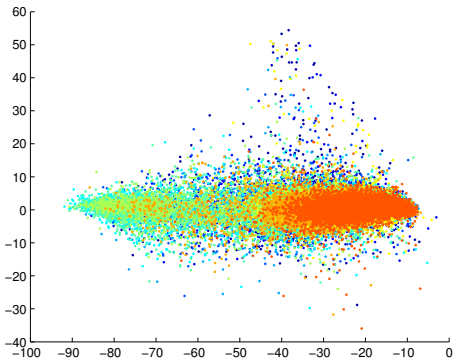
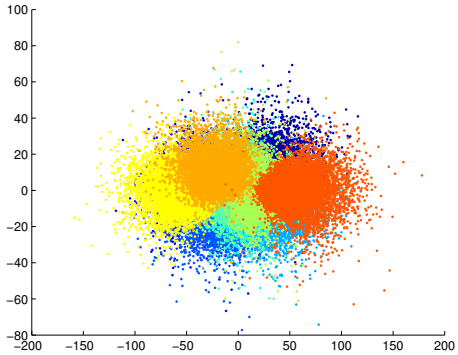
Table 4.16: De-noising performances of optimal clustering-based linear mapping learning and BM3D for SDN

			PSNR (dB)		
γ	σ_u	σ_w	Noisy	Optimal clustering LM	BM3D
0.5	1.5	10	23.1242	33.8326	33.2564
0.5	3	10	18.4487	31.7479	30.0192
0.6	1.5	5	20.5998	32.7183	30.9156
0.57	0.5	5	25.1191	34.9877	33.7646

does not appear as clearly as in Table 4.14, the same conclusions can be drawn from the visual representations: the clusters are well separated in the residue space, but overlap in the decoded patches space. Once again, it means that similar decoded patches will belong to different clusters, while dissimilar decoded patches will belong to the same cluster. This observation also appears clearly in the representation of the first row of Table 4.17, which is very similar to the one of Table 4.14.

Although very efficient in terms of de-noising, the optimal clustering and its approximation by clustering the patches residue can not be used as such in the proposed scheme (described in Fig. 4.6). In fact, this method relies on the knowledge of the source patches, which is only available at the encoder side. Furthermore, from the previous observation, it seems difficult to approach the optimal clustering by directly performing clustering on the coded/decoded patches (whatever clustering algorithm is used). Even if information is sent about the clusters shape (such as the centroid and the covariance matrix), it is clear that there is too much overlap between the optimal clusters in the decoded patches space to effectively separate them. Thus, to benefit from the optimal clustering at the decoder side, one should signal to the decoder the cluster label for each patch. The signaling of the cluster labels is studied in the next section.

Table 4.17: Visual representations of the optimal clustering, $K = 10$. (Each color corresponds to a cluster label.)

Optimal clustering. De-noised PSNR = 41.26 dB.			
			
LDA projection of the coded/decoded patches.		LDA projection of the residue patches.	

4.3.3 Potential RD performances using optimal clustering

In this section, we present the potential RD performances obtained when using the optimal clustering described in section 4.3. We also give the actual performances when taking into account the raw cost of the direct transmission cluster indexes. This cost is estimated using the following equation:

$$R_{indexes} = \frac{W}{M} \times \frac{H}{N} \times T \times \log_2(K) \tag{4.27}$$

where $W \times H$ is the spatial resolution of the sequence, T is the number of frames, $M \times N$ is the patch size, and K is the number of clusters. The only way to decrease this cost is to increase the patch size $M \times N$ and/or decrease the number of clusters K, which is known to disadvantage the de-noising performances. Thus, we adapted these parameters depending on the QP used to encode the sequence. In Table 4.18, we give these parameters along with the de-noising performances. Compared to the results using the K-means algorithm given in Table 4.11, the PSNR gains show the efficiency of the optimal clustering for the removal of quantization noise.

In order to visualize the RD performances, we show the RD curves for the "City" and "Ducks" sequences, which are representative of the worst and best cases respectively, in Fig. 4.13. The curves for the remaining test sequences can be seen in annex in section C.1.2. The gray dash lines represent the cluster indexes bit-rates depending on the different patch sizes and cluster numbers (which are function of the QP). Note that the rate cost of the linear mappings also depends on these parameters, and thus varies as a function of QP. This sometimes results in unexpected RD curve shapes when the cluster indexes bit-rate is not taken into account (green curves). For the same reason, the Bjontegaard measures can not be computed.

The results show that potential huge improvement can be achieved using the optimal clustering in terms of RD performance. However, we can see that the direct transmission (without coding) of the cluster indexes is obviously unrealistic, and yields far worst performances than the standard HEVC. Note that, the indexes exhibit no correlation, neither spatially nor temporally, which excludes compression using predictive model, such as differential pulse-code modulation (DPCM). A statistical analysis of the indexes shows that the histogram is flat, which excludes the use of arithmetic or entropy coding (e.g. Huffman coding).

Table 4.18: De-noising performances of optimal clustering (ΔPNSR)

QP	K	$M \times N$	City	Park Scene	Tennis	Kimono	Cactus	Terrace	Basket	Ducks	People On Street	Traffic	Average
22	16	4×4	1.36	1.08	1.49	1.28	0.79	0.85	0.91	0.79	1.58	1.35	1.15
27	8	4×4	1.20	1.04	1.76	1.55	0.74	0.80	0.90	0.85	1.45	1.28	1.16
32	16	8×8	0.76	0.67	1.63	1.38	0.62	0.59	0.80	0.62	1.08	0.88	0.90
37	4	8×8	0.49	0.46	1.34	1.09	0.48	0.49	0.66	0.51	0.76	0.62	0.69

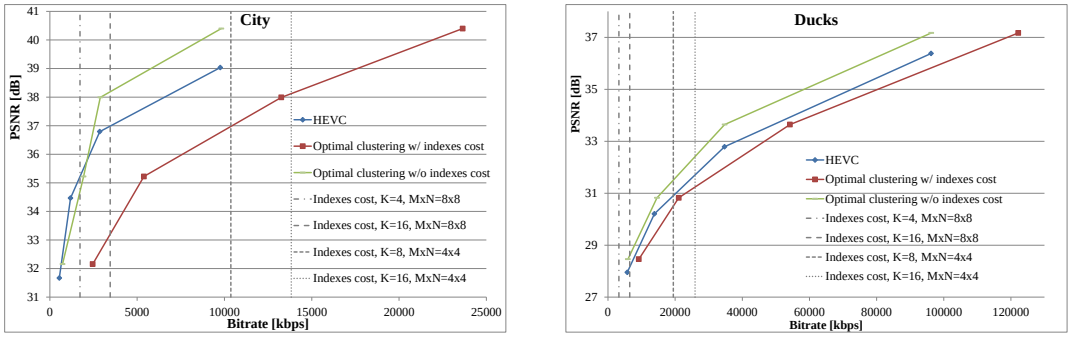


Figure 4.13: RD performances using the optimal clustering, with and without the cluster indexes rate-cost. The performance upper bound of the optimal clustering (green curve) are clearly above the performance of HEVC, especially at high bit-rate. However, when taking into account the cluster indexes raw bit-rate, the performance drop below the HEVC performances.

4.3.4 Conclusion

It is well known in de-noising that exploiting the self-similarities of an image, by grouping or clustering patches, is an efficient tool. The underlying assumption, for white Gaussian noise as well as signal-dependent noise, is that grouped patches will promote sparsity and share a similar noise level. Therefore, a pruning of the coefficients of a sparse representation below this noise level provides significant de-noising performance. We showed in this section that the proposed linear mapping based method also reaches high PSNR gains on clusters which gather patches with similar noise properties. However, we showed experimentally that the assumption that similar noisy patches will share a similar noise level does not hold in the case of compression noise.

The study of the optimal clustering shows that, even if the K-means algorithm yields some interesting results in terms of RD performances, significant improvements can be expected. Even though its use is not mature in the proposed compression scheme (see Fig. 4.6), it sets an upper bound on the RD performances. We show in section 4.5 a new perspective based on classification which could allow to reach a compromise

between the K-means and the optimal clustering on the RD performances.

4.4 Application to super-resolution for scalable video compression

In this section we described a clustering-based linear mapping learning method for the super-resolution that we introduce in a scalable compression scheme. The method is adapted from the de-noising method described in the previous chapter. Here, after clustering, the linear mappings are learned between the low-resolution and high-resolution patches, and then applied to perform super-resolution. If the low-resolution sequence is also corrupted with noise, our method jointly performs super-resolution and de-noising, which makes it suitable for the up-sampling of a decoded base layer sequence in a scalable compression scheme such as SHVC.

4.4.1 Preliminary super-resolution results

We first evaluate the potential of the method purely in terms of super-resolution performances, without encoding the sequences.

Below, we briefly explain how the method can be adapted for super-resolution. More details will be given in the next section. Here, the clustering is performed on the low-resolution (LR) sequence up-sampled to the size of the HR sequence using a simple up-sampling filter. The linear mappings are then learned between the up-sampled LR patches and the corresponding high-resolution (HR) patches for each cluster. The process is thus very similar to the de-noising method proposed in section 4.2, except that here the linear mapping only performs high frequencies reconstruction.

The tests were performed on single images (first frame of the test sequences described in Table 4.5), the LR images were obtained from the source images by down-sampling with a factor 2 using the SHVC filter [47]. The LR image was up-sampled using the SHVC up-sampling filter, so that LR and HR patches share the same 4×4 size. The number of clusters was fixed to $K = 10$.

As for the de-noising application, the optimal clustering formulation (see section 4.3) can be used to define an upper bound on the performances. The tests were performed with K-means and the optimal clustering methods for comparison, and results are given in Table 4.19. We can see that the proposed method improves the performances compared to the SHVC up-sampling filter, especially using the optimal clustering.

Given these promising preliminary results, we propose in the next section a scheme integrating this method in a scalable compression framework. The results shown in section 4.4.5 also take into account the transmission of the linear mappings to the decoder, which was not considered here.

4.4.2 Proposed scheme for salable compression

In this section, we describe the proposed compression scheme improvement using clustering-based linear mapping learning for super-resolution. The method is applied in a scalable

Table 4.19: Super-resolution performances of clustering-based linear mapping on non-encoded low resolutions images

Sequence	SHVC Filter	K-means	Optimal clustering
City	30.9836	32.3000	35.3865
Park Scene	36.3585	36.9171	38.8824
Tennis	40.9845	41.3829	43.0397
Kimono	44.4866	44.6471	45.7041
Cactus	34.1444	34.7399	36.9079
Terrace	28.7704	29.6783	32.3759
Basket	34.9184	35.6637	39.2549
Ducks	33.4748	34.0972	35.9034
People On Street	38.0490	39.1701	41.6081
Traffic	38.4106	39.1342	41.7834
Average	36.0581	36.7731	39.0847

scheme such as SHVC, in which the LR sequence is encoded as a base layer (BL), and the coded/decoded BL is then used to predict the HR sequence, also denoted enhancement layer (EL). Note that SHVC scalable features also include temporal resolution, SNR, bit depth and color gamut [47]. Here, we only focus on the scalability of the spatial resolution. Below, we first describe the proposed method to super-resolve the BL. Then, we explain how the method can be introduced in a scalable coding scheme.

4.4.2.1 Super-resolution of the base layer

The different steps of the proposed super-resolution method are the same as for the de-noising, with an additional step to up-sample the coded/decoded LR sequence. We use here the same formalism as in section 4.2.2. The matrices $\mathbf{M}_d$ and $\mathbf{M}_s$ contain in their columns the vectorized $M \times N$ patches from the up-sampled LR coded/decoded sequence Y_{HR} and the HR source sequence X respectively.

The main idea of the proposed method is represented in Fig. 4.14, and consists in the following steps:

- At the encoder side:
  - cluster the up-sampled coded/decoded patches $\mathbf{M}_d$ (see section 4.4.3)
  - for each cluster c, learn a linear mapping $\mathbf{P}_c$ between the up-sampled coded/decoded sequence patches $\mathbf{M}_d^c$ and the source patches $\mathbf{M}_s^c$:

$$\mathbf{P}_c = \mathbf{M}_s^c \mathbf{M}_d^{cT} (\mathbf{M}_d^c \mathbf{M}_d^{cT})^{-1} \tag{4.28}$$

  - encode the corresponding linear mappings (in matrix form) and transmit them to the decoder (see section 4.4.4)
- At the decoder side:
  - decode the linear mappings

– cluster the up-sampled coded/decoded sequence $\mathbf{M}_d$, as it is performed at the encoder side
– for each cluster c, apply the corresponding linear mapping $\mathbf{P}_c$ to the decoded sequence patches $\mathbf{M}_d^c$ to obtain the super-resolved patches $\mathbf{M}_r^c$:

$$\mathbf{M}_r^c = \mathbf{P}_c \mathbf{M}_d^c \tag{4.29}$$

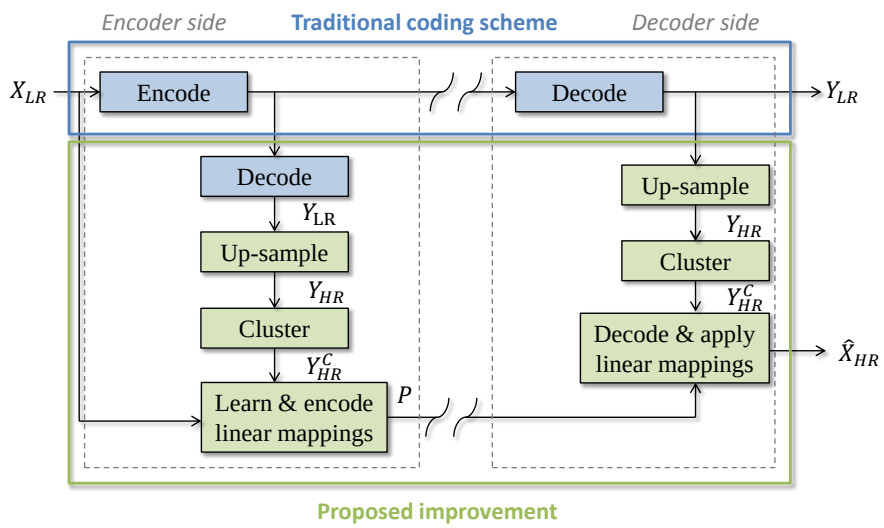


Figure 4.14: Proposed super-resolution scheme for low resolution coded/decoded sequences based on clustering and linear mappings.

Note that here, applying the linear mappings does not only super-resolve the up-sampled coded/decoded patches, but also jointly de-noises them.

4.4.2.2 Super-resolution in a scalable scheme

Traditional scalable coding schemes, such as SHVC, first encode the BL, which is then used to encode the EL. More precisely, the coded/decoded BL is first processed during a so-called inter-layer processing step. The outcome of the inter-layer process can then be used for the prediction during the EL coding. Note that the inter-layer prediction comes in addition to the standard prediction modes. Here, the inter-layer processing only consists in up-sampling the coded/decoded BL (see section 1.2.3). A simple example of such scheme is presented in Fig. 4.15.

To improve the compression performances of the scalable scheme, we propose to replace the up-sampling filter by our super-resolution method, as presented in Fig. 4.16.

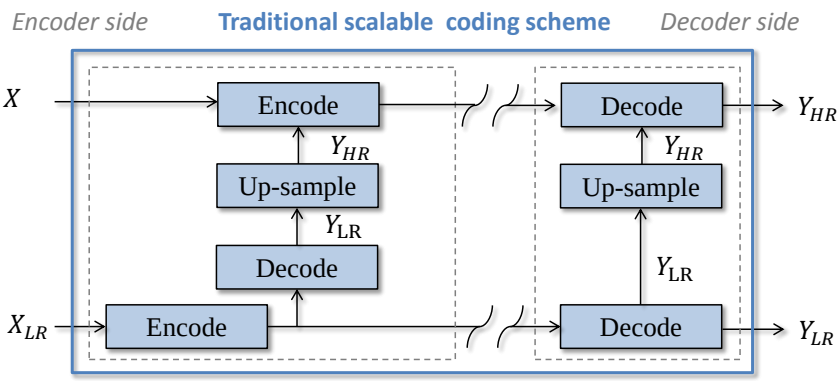


Figure 4.15: Traditional scalable scheme for video compression (e.g. SHVC).

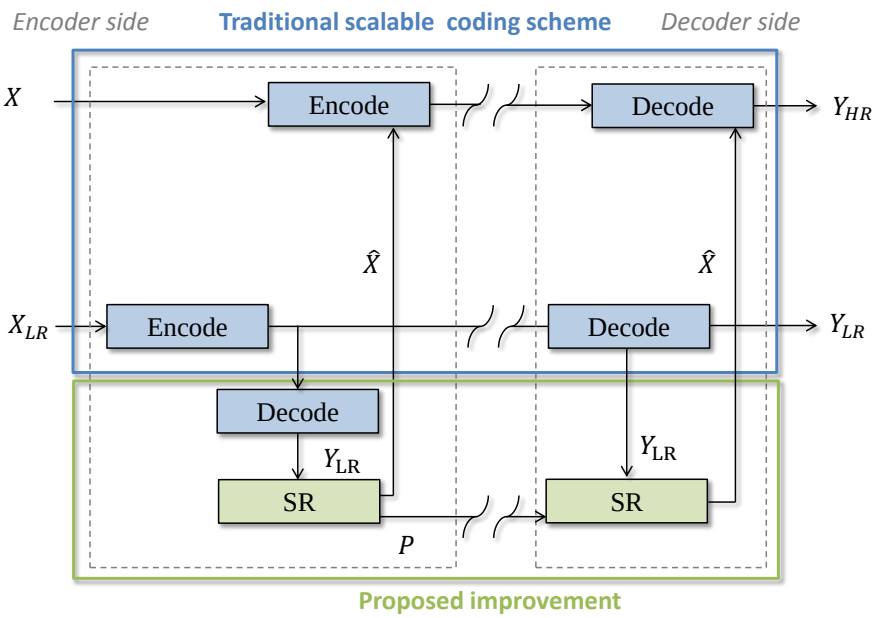


Figure 4.16: Proposed improvement of the scalable compression scheme with super-resolution.

4.4.3 Clustering

The clustering is performed either with the K-means algorithm or the optimal clustering, as preliminary results from section 4.4.1 showed their efficiency. Note that the optimal clustering represents an upper bound on the performances, but as in the previous chapter, would require the transmission of the cluster indexes to be used at the decoder side. The results in section 4.4.5 are obtained with a fixed $K = 10$ clusters, but the method described in section 4.2.3.2 to adaptively select K is still valid.

4.4.4 Linear mappings encoding

The proposed method for super-resolution does not imply any changes to encode the linear mappings compared to the de-noising method. Therefore, the linear mappings are encoded as described in section 4.2.4.

4.4.5 Simulations and results

The test sequences are presented in Table 4.5. Here, only the first GOP of the sequences was processed.

The BL sequences are obtained by down-sampling the source sequences by a factor 2 using the SHVC filter. The BL sequences are then encoded with the HEVC test model HM (ver 15.0) [82] using the Main profile in Random Access, with 4 values for the Quantization Parameter, $QP = 22, 27, 32, 37$. The decoded BL sequences are up-sampled with the SHVC filter or with our super-resolution method. The up-sampled versions (SHVC filter or super-resolution) of the BL encoded at $QP = 22$ is then used to encode the EL with the SHVC test model SHM (ver. 9.0), with $QP = 24, 29, 34, 39$. Note that the two up-sampled BL sequences are used as external base layer thanks to the hybrid codec scalability feature of SHVC.

For all the experiments, the linear mappings are encoded as described in section 4.2.4 using the range extension of HEVC [90][91] (HM 15.0 RExt 8.1) with a fixed $QP = -30$.

First, the super-resolution was performed using the K-means clustering. The RD performances of both the BL and the EL are given in Table 4.20, evaluated with the Bjontegaard measures compared to the SHVC filter. We can see that even if significant bit-rate reduction can be achieved with the super-resolved BL compared to the SHVC filter (up to 40% for the Terrace sequence), these gains are not reported on the EL. In fact, the inter-layer reference frames used for the prediction of the EL are in competition with the other reference frames previously decoded from the EL. The latter are usually of better quality than the former, and the gains obtained with the super-resolution are not enough to make a difference when encoding the EL.

In the second experiment, the super-resolution was performed with the optimal clustering. Note that here the cost of the cluster indexes was not taken into account, thus the results we give are an upper bound. The RD performances of both the BL and the EL are given in Table 4.21. Here, we can see that the gains brought by the optimal

Table 4.20: RD performances computed on the BL and EL using K-means clustering with $K = 10$ for super-resolution (Bjontegaard measures with respect to the up-sampled BL with SHVC filter and corresponding SHVC EL respectively)

	BL gain		EL gain	
	bit-rate % gain	PSNR gain	bit-rate % gain	PSNR gain
City	-20.36	0.501	-0.50	0.004
Park Scene	0.91	-0.007	-0.40	-0.093
Tennis	2.03	-0.059	0.92	-0.049
Kimono	2.87	-0.088	-0.02	0.062
Cactus	-7.74	0.174	0.56	-0.050
Terrace	-39.24	0.560	-0.20	0.090
Basket	-18.53	0.374	2.34	-0.114
Ducks	-8.66	0.194	-0.59	-0.016
Average	-11.09	0.206	0.26	-0.021

clustering on the BL are huge, and therefore have an impact on the coding of the EL, which also reaches significant bit-rate reduction compared to the SHVC filter.

This shows that the proposed scheme (Fig. 4.16) is valid, and as for the de-noising, the upper bound on the performances set by the optimal clustering is substantial. However, the K-means clustering could only improve the performance of the BL.

Table 4.21: RD performances computed on the BL and EL using optimal clustering with $K = 10$ for super-resolution (Bjontegaard measures with respect to the up-sampled BL with SHVC filter and corresponding SHVC EL respectively)

	BL gain		EL gain	
	bit-rate % gain	PSNR gain	bit-rate % gain	PSNR gain
City	-80.66	2.864	-15.70	0.307
Park Scene	-64.29	2.025	-23.87	0.102
Tennis	-62.52	2.584	-19.90	0.934
Kimono	-73.98	3.638	-18.08	1.291
Cactus	-68.02	2.135	-16.99	0.520
Terrace	-96.03	2.887	-11.50	0.470
Basket	-90.49	3.338	-16.64	0.564
Ducks	-73.94	2.318	-16.88	1.682
Average	-76.24	2.724	-17.45	0.734

Below, in order to better visualize the above conclusions, we show in Fig 4.17 the RD curves for the City sequence, with the K-means and the optimal clustering. We can see that when using the K-means, the gain obtained on the BL is slightly reported on the EL at low bit-rates. However at high bit-rates, the quality of the BL is not sufficient (PSNR below 32) to provide an inter-layer prediction competitive with the inter-frame prediction (PSNR above 38). Thus, no difference is visible between the EL encoded with the BL up-sampled with the SHVC filter and the EL encoded with the BL up-sampled with the super-resolution.

The RD curves of the remaining sequences can be seen in annex (section C.2), in Fig. C.2

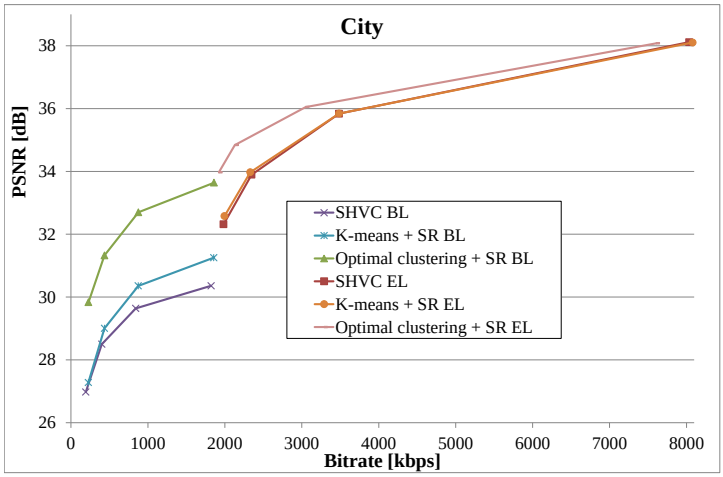


Figure 4.17: RD performance of the City sequence using the K-means and the optimal clustering for super-resolution. The obvious improvement in the performance on the BL are not reported on the EL for the K-means.

4.5 From clustering to classification

The optimal clustering algorithm described in section 4.3 sets an upper bound on the de-noising and corresponding RD performances, which shows that the results using the K-means clustering presented in section 4.2.6 can be further improved. Unfortunately, in practice, this upper bound is not reached, because the optimal clustering algorithm relies on the knowledge of the source patches, which are not available at the decoder side. Experiments showed that the direct transmission of the cluster indexes to the decoder is not worthwhile in a RD sense, especially since this information is very difficult to compress.

The analysis of section 4.3.1 indicates that recovering the optimal clusters using only clustering (un-supervised learning) of the coded/decoded patches is not worth considering. One could consider introducing in the clustering process priors relative to a quantization noise model, which would provide an explicit relation between the decoded signal and the source one. To the best of our knowledge, the quantization noise in modern video codecs, such as HEVC, is too complex to be modeled.

However, the problem can be formulated as a supervised learning (classification) problem. Such machine learning techniques are known to be efficient to obtain a model which links input and output data for which an explicit mathematical equation is unknown. In our context, we can use the labels of the optimal clustering to train a classifier *on data which are available at the decoder side*. Such classifier can then be used at the decoder side to recover the optimal clusters labels.

Two solutions can be considered to train the classifier:

- The classifier is trained at the encoder side, where all the optimal clusters labels are known, *using only training data which are also available at the decoder side*.

The classifier model has then to be sent to the decoder.

- A subset of the optimal clusters labels is sent to the decoder. The classifier is then directly trained at the decoder side on this subset of data. It is then used on the full data set to predict the optimal clusters labels.

These two schemes are represented in Figs. 4.18 and 4.19 respectively.

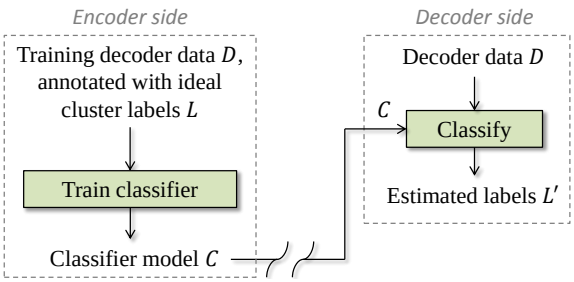


Figure 4.18: Classification-based scheme to recover the optimal cluster labels at the decoder, with transmission of the classifier model.

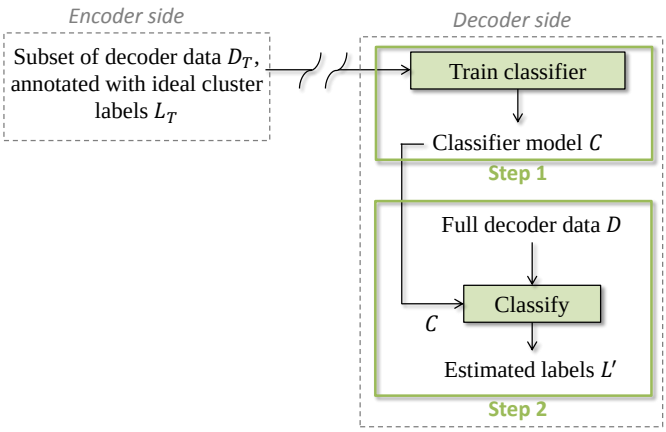


Figure 4.19: Classification-based scheme to recover the optimal cluster labels at the decoder, with transmission of a subset of the training data.

The feature vectors on which we perform the classification can obviously contain the (vectorized) coded/decoded patch. However, additional information available at the decoder can help to improve the classifier performance, by adding new dimensions in order to maximize inter-class separability. Among others, the feature vector could for example contain:

- The coded/decoded patch
- The prediction patch

- The transform
- The quantized transform coefficients
- The prediction mode: intra or inter
- In case the prediction is intra: the directional mode
- In case the prediction is inter: the motion vector
- The frame type (I-P-B)
- The quantization parameter
- The bit-rate associated to the patch

The choice of the features for classification is usually critical, and one should be careful when selecting them. For example, selecting the first three features in the above list might be redundant, as the decoded patches are reconstructed from the prediction and the inverse transformed quantized coefficients. Instead, one could select only the coded/decoded patches and their corresponding prediction.

Note that there is no guarantee that we can recover perfectly all the cluster labels without knowledge of the source patches. In the next section, we evaluate several classification algorithms performances, notably depending on the features choice.

4.5.1 Preliminary results: classification performances

In this section, we aim to evaluate the ability of several classification algorithms to recover the optimal clusters labels.

We consider the scheme shown in Fig. 4.18, where the classifier is learned on the full data set. Even with this configuration, the optimal clusters labels might not be completely recovered. Thus, after the classification step, the linear mappings are re-evaluated.

The optimal cluster labels are obtained using the optimal clustering algorithm with $K = 10$. The test images are coded with a modified version of a JPEG encoder in which a prediction is performed using the intra coding modes of H.264. Three quality factor (QF) values are used: 75 (high quality), 50, and 25 (low quality). The block size in the codec is modified to use 4×4 blocks. This simple codec architecture was preferred over more complex H.264 or HEVC codecs to have easier access to the decoder data.

Thus, the following features can be used in the classification algorithm:

- The coded/decoded block
- The prediction block
- The H.264 intra directional mode
- The bit-rate associated to the block

When all the features are combined, the feature vector has a dimension of 34: 4×4 vectorized coded/decoded block, 4×4 vectorized prediction block, one directional mode (scalar) and one bit-rate (scalar).

In order to reduce the complexity, the tests are performed on nine images of resolution 200×200, cropped from the first frame of the following test sequences: City, Park Scene, Kimono, Cactus, Terrace, Basketball, Ducks, People On Street, and Traffic (shown in Fig. C.3 in annex, section C.3.1). Although the resolution is small, the images cover a wide range of textures. Nevertheless, the results are consistent over the different images, so we only give the average results.

Below, we give the average classification results over all the test images, using 4 popular classification algorithms: linear SVM, non-linear Gaussian SVM, decision tree and bagged decision tree, described in section 4.1.2. Note that in this section we are only concerned with the classification and following de-noising performances of the algorithms, and do not take into account the cost of the models in terms of bit-rate.

4.5.1.1 Choice of the features

We first evaluate the classification performances depending on the selected features and the corresponding de-noising performances evaluated as the PSNR gain over the coded/decoded image with $QF = 50$. The results are shown in Fig. 4.20. The de-noising performances obtained with the K-means algorithm are indicated by a red line, the performances of the optimal clustering with a green line (upper bound).

Not surprisingly, the H.264 intra modes and the block bit-rates alone do not provide enough information to discriminate the classes, and perform poorly with all the algorithms (always under 20 % of the right labels recovered). However, when using the prediction or the coded/decoded blocks, a significant amount of the right labels can be recovered. The performance can be even improved when combining the different features.

The de-noising performance corresponds directly to the classification performance. The best performing algorithms are the Gaussian SVM, which get close to the optimal performances, and the bagged trees, which clearly outperform the K-means performances. We can see that a PSNR gain is always obtained. However, these gains are not always better than the gain of the K-means algorithm. These results should be considered carefully, since the de-noising gains do not directly translate into RD gains. In fact, the best de-noising performances usually correspond to complex classifier models, as shown in the next section.

The corresponding results for $QF = 25$ and $QF = 75$ are very similar and given in annex, in section C.3.2.

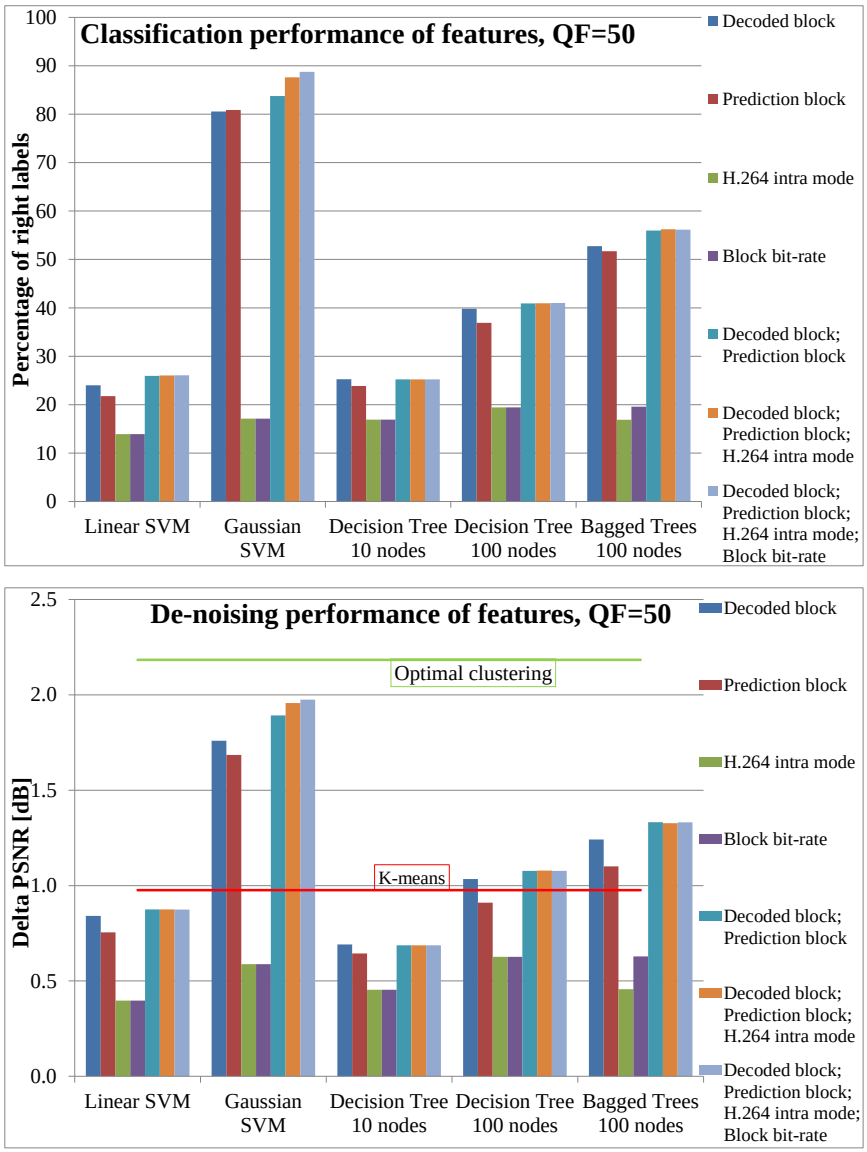


Figure 4.20: Average classification (top) and de-noising (bottom) performances with different features and combination of features, with images encoded at $QF = 50$.

4.5.1.2 Dimensionality reduction using PCA

In Fig. 4.21, we show the classification performances obtained before and after performing a PCA on the combined features and the corresponding de-noising performances, with $QF = 50$. The PCA dimension is gradually reduced.

In general, compared to the raw features, the PCA (with all 34 dimensions) can slightly improve the performances, both in terms of classification and de-noising. More interestingly, the dimension of the PCA can be reduced from 34 to 10 with very little loss in the performances. Reducing the dimension can be advantageous because it can reduce the complexity as well as the rate-cost of the classification model (see section 4.5.2). When the dimension is reduced to 3, there can be a slight loss in the performances, acceptable for most classification algorithms. However, when reduced to 1, we notice a dramatic loss in the performances which is not acceptable.

The best performing algorithms are still the Gaussian SVM and the bagged trees.

The corresponding results for $QF = 25$ and $QF = 75$ are very similar and given in annex, in section C.3.2.

From these results, we can conclude that:

- Non negligible proportion of the optimal clusters labels can be retrieved using only data available at the decoder side (*i.e.* without any knowledge of the source signal). Significant de-noising gains can thus be obtained, leading to a compromise between the K-means clustering performances and the upper bound of the optimal clustering.
- Combining the different features available results in better performances.
- Reducing the dimension of the combined features, e.g. using PCA, results in very little loss in the performances. This can be interesting in order to obtain compact classifier models, as explained in the following section.

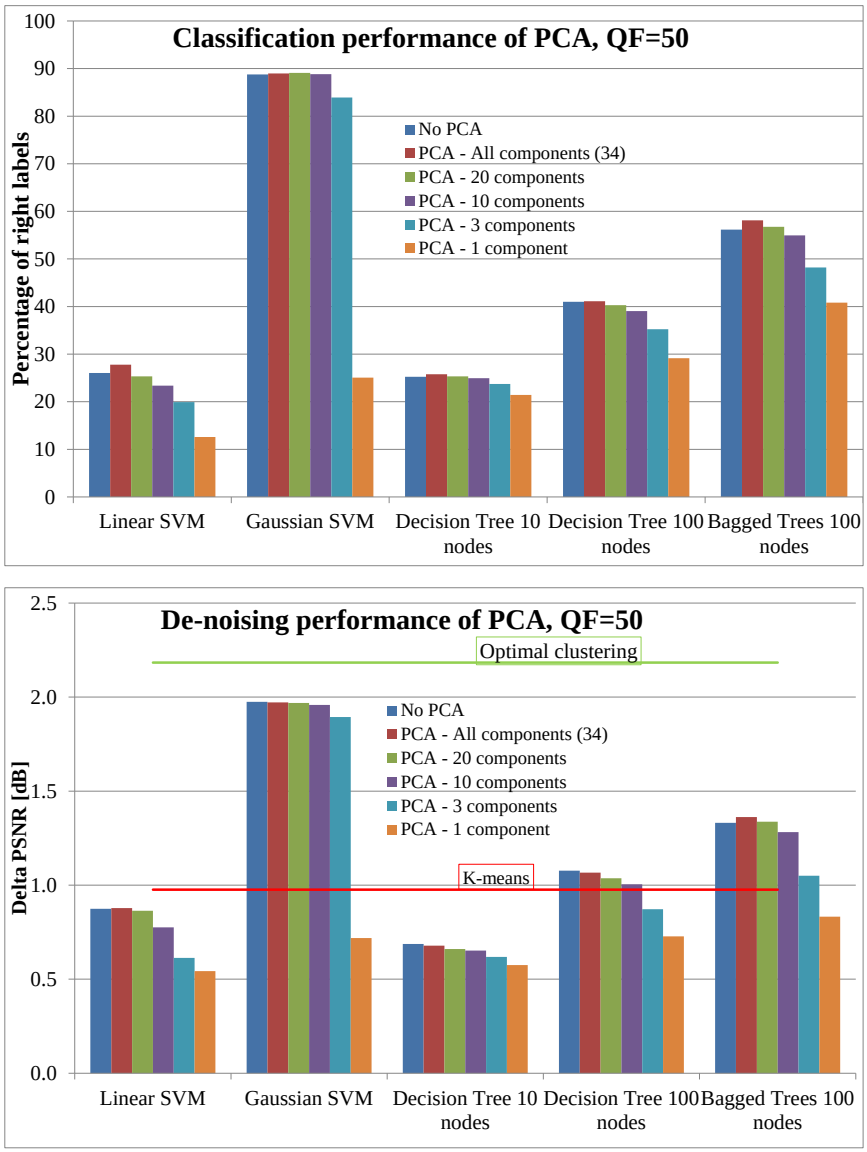


Figure 4.21: Average classification (top) and de-noising (bottom) performances with different dimensions of a PCA performed on the combination of features, with images encoded at $QF = 50$.

4.5.2 Application in the compression context

4.5.2.1 Transmission of a classifier model

The results above demonstrate that the proposed scheme based on classification can be useful to recover the optimal cluster labels. These results correspond to the scheme of Fig. 4.18, where the classifier model C has to be sent to the decoder. Here, we analyze the different classifier models, given their description in section 4.1.2, and give an estimation of their rate cost. Results assessing the de-noising performances depending on the models complexity are also given. These results are obtained with $K = 2$ classes.

Support Vector Machine (SVM) The results from the previous section show that the classification performances of the linear SVM are limited. Thus, in this section, we only consider the case of the non-linear Gaussian SVM.

The classifier model for the SVM consists in a set of hyperplanes, which have to be sent to the decoder in our context. However, in the case of the non-linear SVM, the hyperplanes can not be computed explicitly. Instead, the classification relies on the knowledge of the support vectors and their labels (see Eq. 4.15 in section 4.1.2.1). The support vectors belong to the training set, and are in our case feature vectors, which will have in practice to be signaled to the decoder, along with their class labels. To estimate the corresponding rate, we can consider that the value 0 is associated with the non-support vectors, while the class label is associated with the support vectors. The distribution of these values is not expected to be homogeneous, we can thus consider compressing this additional information using entropy or arithmetic coding.

As shown in Fig. 4.22, increasing the number of support vectors increases the de-noising performances, but will require a larger cost in terms of bit-rate. The number next to each point of the curves indicates the percentage of the right optimal labels recovered. The results for $QF = 25$ and $QF = 75$ are similar to the one below, and given in annex in section C.3.3.

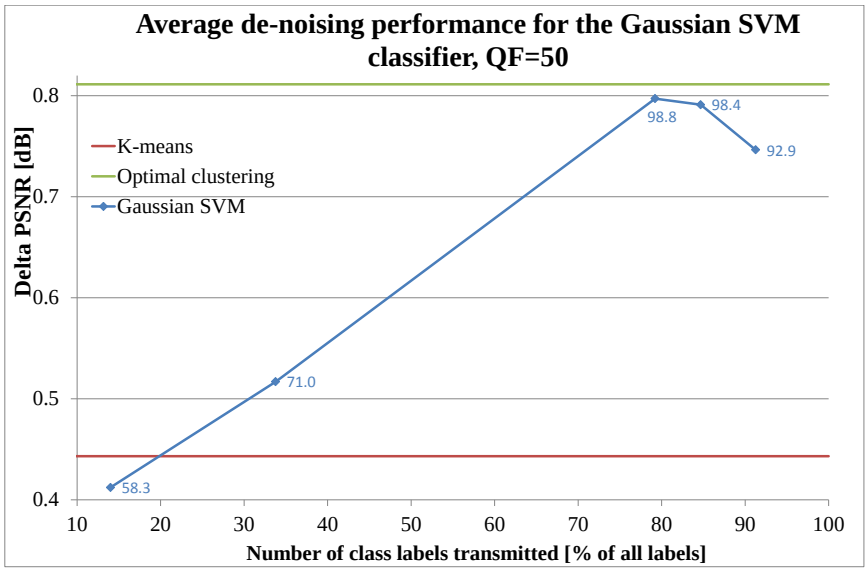


Figure 4.22: Average performances of the Gaussian SVM depending on the percentage of support vectors, $QF = 50$.

Decision Tree Here, the classifier model to be sent to the decoder is the decision tree itself. In the case bagging is performed, several trees are transmitted.

As for the Gaussian SVM, increasing the complexity of the model increases the de-noising performances as well as the bit-rate. Here, the complexity of the model is defined by the number of nodes in the tree. For each node, we need to send the index of the component being evaluated, as well as the threshold value against which the component is compared (see Eq. 4.17 in section 4.1.2.2).

If we note d the dimension of the feature space, we can roughly evaluate the rate of a node as $R_N = \log_2(d) + 16$, where the left side of the addition represents the rate of the component and the right side the rate of the threshold, estimated as a 16 bits floating point value. In addition, the values of the class labels for each leaf need to be transmitted. If we note N_L the number of leaves, this rate can be evaluated as $R_L = N_L * \log_2(K)$. This particular rate could be even compressed using entropy or arithmetic coding. If we note N_N the number of nodes in a tree, and N_T the number of trees to be transmitted, the total rate associated to this classifier model can be evaluated as:

$$R_{DT} = N_T * (N_N * R_N + R_L) \tag{4.30}$$

In Fig. 4.23, we compare the performances of different decision trees models. For all models, a PCA is performed on the features vectors. For the decision tree, we first consider using all the dimensions, $d = 34$, and then reduce the dimension to 1. We then vary the number of nodes, $N_N = 1, 5, 20, 50, 100, 200, 1000$, which gives the different points of the curves. For the bagged decision trees, the dimension is reduced to 1, and

we consider different number of nodes, $N_N = 1, 200, 1000$. We then vary the number of trees to obtain the points of the curves $N_T = 1, 5, 20, 50, 100, 200$. The number next to each points of the curves indicate the percentage of the right optimal labels recovered.

We can see that it is more advantageous in a RD sense to send one complex tree rather than several (bagged) simpler trees.

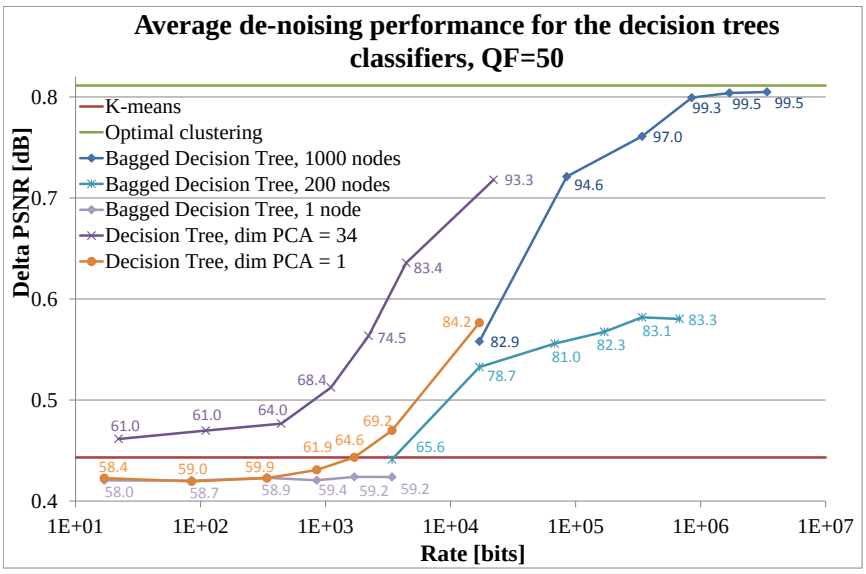


Figure 4.23: Average RD performances of the decision trees, $QF = 50$.

The results for $QF = 25$ and $QF = 75$ are similar to the one above, and given in annex in section C.3.3.

4.5.2.2 Learning on a subset of data

In the second scheme of Fig. 4.19, we propose to send only a subset of the optimal class labels, so that the classifier is learned at the decoder side.

The subset of blocks for which the optimal class labels are transmitted can be selected by computing epitomes of each frame. At the decoder side, the classifier is learned on the feature vectors corresponding to the blocks of the epitomes, and then used to obtain the labels of the non-epitome blocks. The scheme is very similar to the Gaussian SVM described above, where the feature vectors corresponding to the epitome blocks act as support vectors.

To estimate the corresponding rate, we can consider that the value 0 is a associated with the non-epitome blocks, while the class label is associated with the epitome blocks. The distribution of these values is not expected to be homogeneous, we can thus compress this additional information using entropy or arithmetic coding.

In Fig. 4.24, we show the performances when transmitting the class labels of epitomes of different sizes. The epitomes are obtained using the method described in section 3.2, with $\varepsilon_M = 2.0, 3.0, 5.0, 7.0, 10.0, 15.0$, and $\varepsilon_A = 0.5$. We used a bagged decision

trees classifier, with a PCA performed on the feature vectors, $d = 34$, $N_N = 100$ and $N_T = 1000$. The number next to each points of the curves indicate the percentage of the right optimal labels recovered. For comparison, we show the corresponding results using "random epitomes" of the same size. We can see that the choice of the epitome is more relevant that the random epitome. As expected, bigger epitomes allow better de-noising performances, but will require a larger cost in terms of bit-rate.

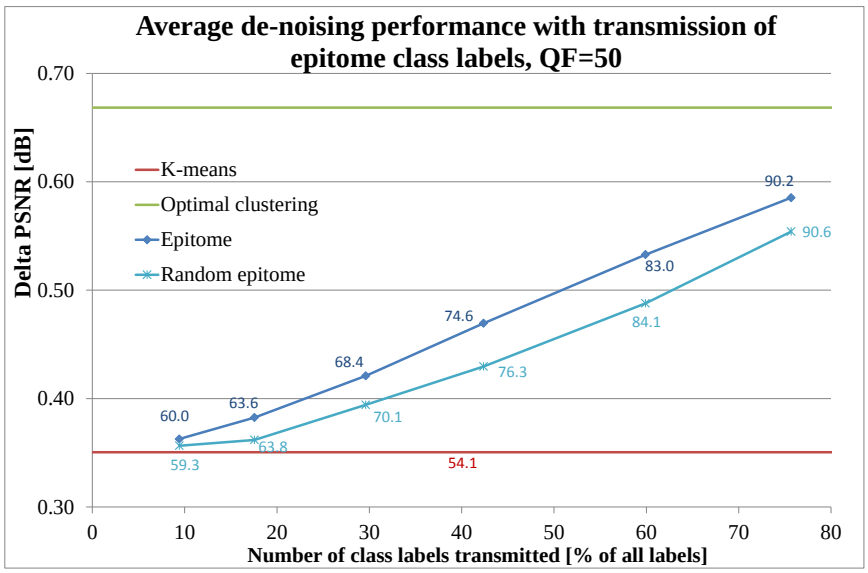


Figure 4.24: Average performances of the bagged decision tree depending on the size of the epitome, $QF = 50$.

The results for $QF = 25$ and $QF = 75$ are similar to the one above, and given in annex in section C.3.3.

4.5.2.3 RD performances

De-noising of a single layer scheme. Preliminary tests to evaluate the RD performances were performed using the same test conditions as in section 4.5.1, but using here sequences of 10 frames. Here, the scheme described in section 4.5.2.2 is used, where a subset of the optimal class labels is transmitted for blocks of the frames epitomes. The epitomes are obtained using the method described in section 3.2, with $\varepsilon_M = 2.0$, and $\varepsilon_A = 0.5$. The classifier model, learned at the decoder side, is a bagged decision trees classifier, with a PCA performed on the feature vectors, with $d = 34$, $N_N = 100$ and $N_T = 1000$.

Results are shown for the Ducks sequence in Fig. 4.25. We can see that the even though the classification allows to recover a PSNR close to the optimal clustering, the cost of the subset of labels transmitted to the decoder is still too high to outperform the K-means clustering in terms of RD performances.

Additional results for the other test sequences are given in annex in section C.3.4.1.

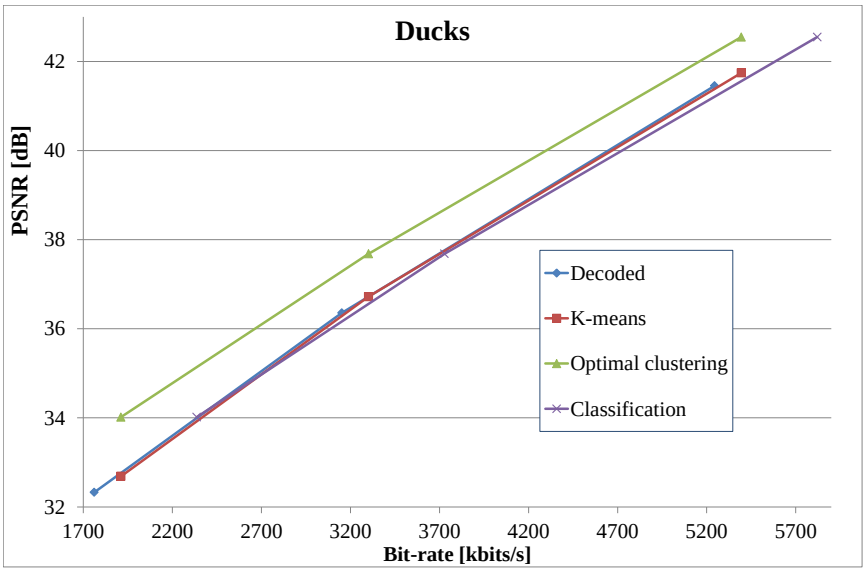


Figure 4.25: RD performances for the Ducks sequence with recovering of the optimal class labels using classification.

Super-resolution in a scalable scheme. We evaluate here the RD performance in a spatial scalable scheme, using the same sequence as in the previous section. The base layer is first down-sampled with a factor 2, and encoded using the same modified JPEG encoder as in section 4.5.1, using $QF = 80, 60, 40, 20$. The enhancement layer is encoded using a modified JPEG encoder where the up-sampled of super-resolved decoded base layer is used as prediction, with the same QF values as the base layer. A subset of the optimal class labels is transmitted as described in section 4.5.2.2. The epitomes are obtained using the method described in section 3.2, with $\varepsilon_M = 2.0$, and $\varepsilon_A = 0.5$. The classifier model, learned at the decoder side, is a bagged decision trees classifier, with a PCA performed on the feature vectors, with $d = 34$, $N_N = 100$ and $N_T = 1000$.

Results are shown for the Basketball and Traffic sequence in Fig. 4.26. We can see that in this case the proposed classification-based scheme can benefit both the base layer and the enhancement layer in terms of RD performances. However, these results do not necessarily hold for the other sequences, which are given in annex in section C.3.4.2.

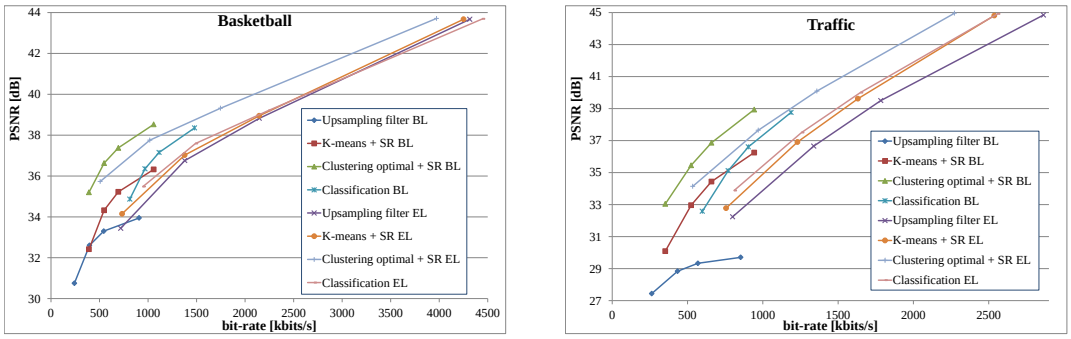


Figure 4.26: RD performances with recovering of the optimal class labels using classification for super-resolution in a scalable scheme

4.5.3 Conclusion

We propose in this section a strategy to obtain a compromise between the performances of the K-means and the optimal clustering algorithms. The preliminary classification results show an interesting potential, and the RD results showed that it is possible to outperform the performances of the K-means clustering in a scalable scheme. Nevertheless, we believe that several means could be considered to further improve the approach.

Indeed, only two types of classification algorithms were considered, SVM and decision trees. However, many classification models could be investigated, such as naive Bayes [96], discriminant analysis [97][98][99], nearest neighbors [100] or deep neural networks [101] classifiers. In our context, we could investigate classifiers with compact models which could perform similarly to the studied models while reducing the rate cost of the model if it is transmitted to the decoder.

Alternatively, we could also investigate classifiers with higher classification performances for a given subset of optimal class labels. The RD performances could thus be improved by a better de-noising efficiency for a similar bit-rate.

4.6 Conclusion and perspectives

In this chapter, we proposed a clustering-based scheme with linear mappings learning for quantization noise removal. The proposed out-of-the-loop framework aims at improving the performances of a video compression scheme by sending to the decoder linear mappings learned at the encoder between the clustered coded/decoded patches and the corresponding source patches. At the decoded side, the same clustering as the encoder side is performed, and the corresponding linear mappings received are used to de-noise the decoded video sequence.

First, the K-means algorithm is used to partition the video sequence. A binary tree partitioning technique based on a RDO criterion is also proposed to adaptively select the number of clusters.

Results show that the proposed scheme can improve the coding efficiency of HEVC, and is especially efficient when combined with a first pass de-noising algorithm.

We also proposed an optimal clustering algorithm which sets an upper bound on the performances, and show that further improvement can be reached.

The clustering-based scheme is then used for super-resolution in a scalable video compression context. When using the K-means algorithm, we observed that the gains brought to the base layer are not reported on the enhancement layer. Experiments performed with the optimal clustering showed that the proposed scalable scheme could outperform SHVC, but requires an important improvement of the base layer.

Finally, a classification-based scheme is presented which aims at reaching a compromise between the K-means and the optimal clustering in terms of RD performances. Some promising preliminary results are obtained, and the experiments showed that it is possible to outperform the K-means in terms of RD performances in a simple JPEG-based spatial scalable scheme.

Several approaches can be considered to further extend the proposed scheme.

First, all the experiments were performed on the luminance channel, nevertheless, the proposed scheme could be applied to color sequences as well. The proposed method can be adapted to color sequences simply by stacking all the color components of the vectorized patches in the matrices $\mathbf{M}_d$ and $\mathbf{M}_s$. In addition to the extension of the method to color components, the proposed method could be applied in a scalable scheme to color gamut scalability, e.g. to predict a 4:4:4 or RGB enhancement layer from a 4:2:0 base layer. The adaptation of the proposed scalable scheme to the encoding of a high dynamic range enhancement layer from a low dynamic range base layer is also straightforward.

Second, we observed that the optimal clustering approach was not efficient in terms of RD performances when taking into account the bit-rate of the cluster indexes. A promising alternative scheme based on classification is proposed in order to reach a compromise between the K-means performances and the optimal performances. The technique could be further improved by considering more efficient or more compact classifiers, notably in order to be applied in modern scalable coding schemes.

Finally, we believe that our method would be particularly suited for cloud-based compression. Cloud-based compression has gained interest recently [102], as well as applications such as de-noising based on external images [103][104], or super-resolution [105][106]. By introducing our method in such framework, e.g. using the scheme of Fig. 4.27, we can assume that the linear mapping matrices are available in an external database, which means that no bit-rate overhead is needed. Thus, we could purely benefit from the de-noising improvement without any additional cost in terms of bit-rate.

The learning of the linear mappings, performed in an off-line training phase, thus becomes critical. During the on-line processing phase, the decoded patch being processed would query the database to retrieve an appropriate linear mapping, with which it could then be de-noised.

For example, if the linear mappings are learned on clusters found with the K-means

algorithm, the centroids of the clusters would be stored in the database along with the corresponding linear mappings. When a patch is processed, one would search for the closest centroid in the database, and use the associated linear mapping to de-noise the current patch. Such scheme has been for example considered for super-resolution in [107].

The scheme of Fig. 4.27 is also especially suitable to be combined with the proposed classification-based approach, where the classifiers would be learned during the off-line training phase and then stored in the database to be later used on-line.

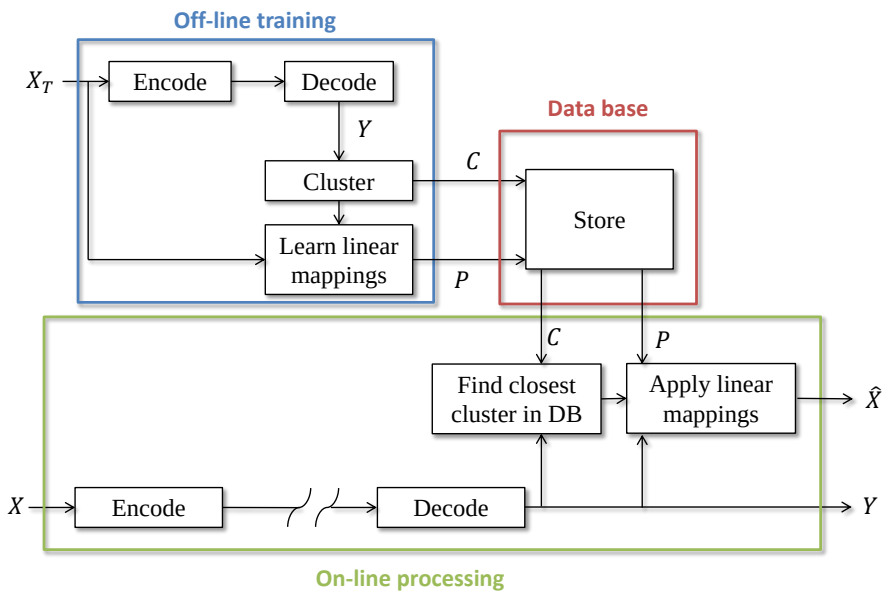


Figure 4.27: Off-line linear mappings learning for quantization noise removal.

Conclusion

In this thesis, we explored novel approaches in order to improve existing video compression schemes. One of the main principle of video compression is to reduce the temporal and spatial redundancies occurring in natural images/videos. Video compression standards exploit this principle through Inter and Intra prediction tools respectively, which we described in details in Chapter 1. Inter prediction thus rely on motion estimation/compensation while Intra prediction propagate the information from the spatially neighboring pixels previously decoded. The goal of the work presented in this manuscript was to improve the existing compression schemes by further exploiting the intrinsic redundancies in natural images and videos, notably through the use of multi-patches methods.

This first lead us to improve the current standard prediction tools, using multi-patches methods based on Locally Linear Embedding (LLE). In Chapter 2, we presented methods which consist in a first step in approximating the current template, by linearly combining K templates found in the previously decoded part of the video. The linear weights are obtained with the LLE, and then applied on the blocks adjacent to the templates in order to obtain the predictor of the current block. Different strategies are proposed to find the K patches necessary to perform the prediction. First, we proposed to search for the K nearest neighbors (K-NN) of the template only. This technique can be reproduced at the decoder side, and thus we spare the syntax usually associated with motion estimation (list index, reference frame index, motion vector). Then, we proposed to search for the K-NN of the full patch comprising the current block and template, which enforces the correlation between the templates and their adjacent blocks, which in return improves the quality of the predictor. Such methods were already proven efficient for the reduction of spatial redundancies through Intra prediction, and we showed that they can be valuable to improve Inter prediction as well. Moreover, we demonstrated that combining the proposed methods for both Intra and Inter prediction greatly benefits the coding performances.

Independently from the standard video compression techniques emerged the concept of epitome, which is defined as a condensed representation of an image or video containing the essence of its textural properties. Due to its proximity to the compression principle of reducing the self-similarities within an image or video signal, several in-loop approaches exploiting epitomes have been proposed. In particular, one method combined the epitomic concept and the LLE-based multi-patches methods discussed above. At the same time, we observed that many applications such as de-noising and

super-resolution exploit redundancies in natural images and videos. We thus decided to propose an epitome-based methods for compression with an out-of-the-loop perspective, where the epitome containing high quality patches is used at the decoder to improve the decoded video, using multi-patches image restoration methods such as de-noising or super-resolution. This work is presented in Chapter 3, where we first described an efficient method for epitome generation, based on list or cluster approaches. We then implemented this epitome generation method in an epitome-based quantization noise removal scheme, where high-quality epitomes are sent to the decoder and used to de-noise the non-epitome patches of the video sequence. Three epitome-based de-noising methods are proposed, which first consists in looking for the K-NN of the patch being processed among the coded/decoded patches colocated with the epitome. The pairs of coded/decoded K-NN patches and their corresponding high quality patches from the epitome are then exploited to perform the de-noising. A local linear combination (LLC) method is proposed, which linearly combines the K high quality patches with weights computed from the corresponding noisy patches using the Non Local Mean (NLM) or LLE approach. An epitome-based method adapted from the BM3D algorithm is also introduced, which infer the thresholds for the hard thresholding and Wiener filter coefficients from the high quality patches. Finally, an epitome-based local linear mapping (LLM) method is described, where a linear mapping projection is learned between the K-NN coded/decoded patches and their corresponding high quality patches, and then applied on the patch being de-noised. The scheme was shown efficient to improve the performances of the scalable video compression scheme SHVC.

However, the scheme described above was not able to outperform the performances of the single layer codec HEVC. We thus proposed in Chapter 4 an out-of-the-loop scheme based on a different paradigm: instead of transmitting an additional layer containing high quality patches, we learn at the encoder side linear mappings between the coded/decoded patches and their corresponding source patches, and send the projection matrices to the decoder. In order to take into account the self-similarities of the signal, the patches are clustered, and a linear mapping is learned for each cluster. The same clustering is then performed at the decoder side, and the corresponding linear mappings are then applied on the patches of the cluster to obtain a de-noised version. The results show that the proposed scheme outperforms HEVC. In addition, we proposed an optimal clustering formulation, which defines an upper bound on the performances, and demonstrates that potential further improvement could be reached. We also proposed to adapt the scheme for scalable video compression, notably using super-resolution for spatial scalability. However, if the improvement on the base layer are indubitable compared to SHVC, they are not reported on the enhancement layer. We observed that potential improvement can be obtained on the enhancement layer when using the optimal clustering. Finally, a novel classification-based scheme is presented which aims at reaching a compromise between the K-means and the optimal clustering in terms of RD performances. The results obtained are promising, and show that the K-means can be outperformed in terms of RD performances in a JPEG-based spatial scalable scheme.

Thus, in this thesis, we focused on exploiting the natural self-similarities of an image or video signal to improve compression performances. First, we improved the

existing prediction tools which already exploit the texture redundancies of a video. Then, we introduced in an out-of-the-loop compression framework different methods exploiting texturing redundancies, originally designed for epitome generation, clustering, de-noising or super-resolution. We proposed novel and disruptive approaches, and we demonstrated their efficiency for different applications, either single layer or scalable compression schemes. Furthermore, we believe that this work can be extended to additional applications such as high dynamic range (HDR) image compression, or cloud-based compression, as discussed below.

Future work

Several approaches can be considered to further extend the work of this thesis.

First, we have considered this work as an upstream investigation, with a focus on the improvement of the RD performances, which in return requires a high processing complexity. For all the contributions, the complexity mainly comes from the self-similarities search, such as the nearest neighbors search, for which efficient and fast methods exist [14][15]. This process is also highly parallelizable, and could be for example implemented on GPU [19]. Similarly, fast clustering methods could be considered [108]. Finally, the LLE or NLM weights computations could be fasten with approximate methods [18].

Second, although we proposed in Chapter 3 an epitome generation method optimized in terms of complexity, the performances of the compression scheme proposed in this chapter could be improved by adapting the epitomic model so that it takes into account the temporal redundancies. A direct approach would be to create an epitome of the frame epitomes. Alternatively, a sprite of the GOP could be generated [84][85], and the epitome would be extracted from the sprite. Furthermore, the epitomic model used in this work are optimized to reconstruct the original image. We could therefore adapt the model so that it takes into account the final application, which is here multi-patches de-noising or super-resolution. Such method has been proposed in [86] for a LLE-based multi-patches super-resolution application.

Third, the optimal clustering algorithm proposed in Chapter 4 remains at the time of writing a theoretical upper bound on the performances. Indeed, we observed that the optimal clustering could not be used directly in the proposed scheme, as it requires the transmission of the cluster indexes to the decoder, which is not efficient in terms of RD performances. However, given this upper bound, it is clear that the results that we obtained with K-means clustering can be improved. We postulate that a compromise could be reached between the K-means and the optimal clustering performances by using an alternative scheme based on classification. In this scheme, only a subset of the optimal cluster indexes would be signaled to the decoder, where a classifier model could be learned and then used to derive the unknown cluster indexes. Alternatively, the classifier could be learned at the encoder side and transmitted to the decoder.

Finally, the proposed schemes could be adapted for other applications. For example, the methods were only tested on the luminance channel, but could be adapted to color videos as well. Notably in scalable schemes, the proposed methods could be extended to

color gamut scalablity, and bit-depth scalability, in particular for the encoding of a high dynamic range enhancement layer from a low dynamic range base layer. In addition to HDR compression, the proposed scheme could be adapted to other trending topics such as cloud-based compression [102], or plenoptic imaging compression [109][110][111], which contain many redundancies.

Author's publications

Articles

Journal paper

M. Alain, C. Guillemot, D. Thoreau, and P. Guillotel, "Inter Prediction Methods based on Linear Embedding for Video Compression," *Signal Processing: Image Communication*, vol. 37, pp. 47-57, sep 2015.

Conference papers

M. Alain, S. Cherigui, C. Guillemot, D. Thoreau, and P. Guillotel, "Locally linear embedding methods for inter image coding," in *IEEE International Conference on Image Processing (ICIP)*, pp. 1904-1908, 2013.

M. Alain, C. Guillemot, D. Thoreau, and P. Guillotel, "Clustering-based Methods for Fast Epitome Generation," in *European Signal Processing Conference (EUSIPCO)*, (Lisbon), pp. 211-215, IEEE, 2014.

[57] S. Cherigui, **M. Alain**, C. Guillemot, D. Thoreau, and P. Guillotel, "Epitome inpainting with in-loop residue coding for image compression," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2014, pp. 5581-5585, 2014.

[86] M. Turkan, **M. Alain**, D. Thoreau, P. Guillotel, and C. Guillemot, "Epitomic image factorization via neighbor-embedding," in *IEEE International Conference on Image Processing (ICIP)*, 2015.

Patents

M. Alain, D. Thoreau, P. Guillotel and C. Guillemot, "Method and apparatus for constructing an epitome from an image", EP2903288/US2015215629, 2015

C. Guillemot, **M. Alain**, D. Thoreau and P. Guillotel, "Method and apparatus for building an estimate of an original image from a low-quality version of the original image and an epitome", WO2015067518, 2015

D. Thoreau, S. Cherigui, **M. Alain**, P. Guillotel and C. Guillemot, “Inter-image prediction method and device and corresponding coding method and apparatus”, EP2901697/WO2014048946, 2015

S. Cherigui, **M. Alain**, C. Guillemot, D. Thoreau and P. Guillotel, “Inter-image prediction method and device and corresponding coding method and apparatus”, EP2782344/US2014314330, 2014

D. Thoreau, S. Cherigui, **M. Alain**, P. Guillotel and C. Guillemot, “Methods for encoding and decoding a picture and corresponding devices”, EP2938074/US2015312590, 2015

M. Le Pendu, D. Thoreau, **M. Alain**, and C. Guillemot, “Method for encoding and decoding an image block based on dynamic range extension, encoder and decoder ”, EP2958329/US2015365701, 2015

Author or co-author of 14 patents not yet disclosed

Appendix A

Appendix of chapter 2

Figure A.1: Frames extracted from the test sequences. From top to bottom, left to right: Foreman, Rushes, Matrix, City and Spincalendar

Appendix B

Appendix of chapter 3

B.1 Comparison of the epitomes obtained with the different self-similarities search methods

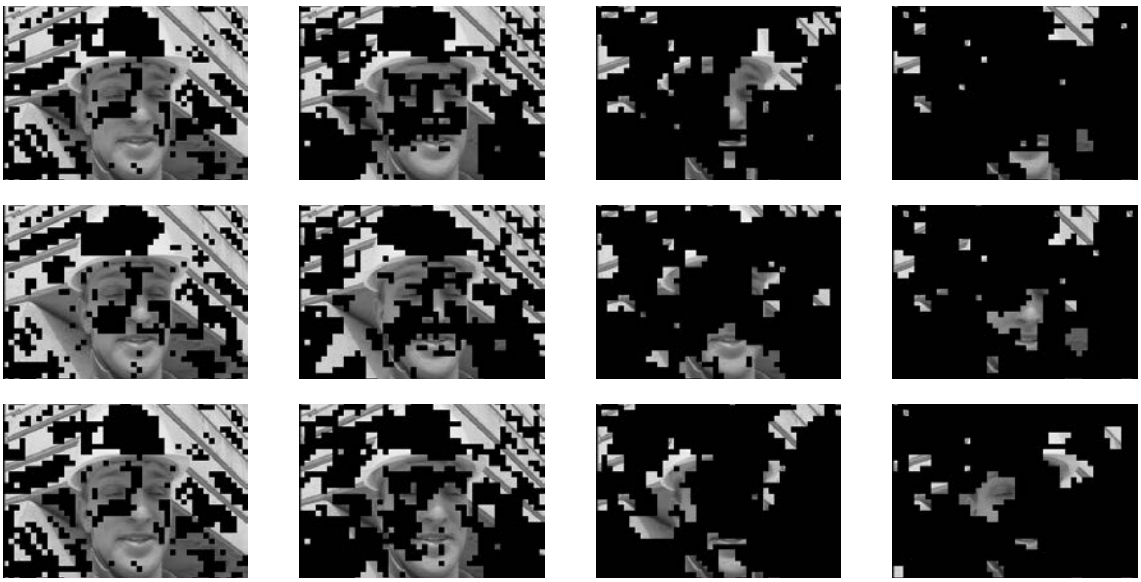


Figure B.1: Epitomes extracted from Foreman. The epitome was obtained using the full search method (top), the list-based method (middle) and the threshold-based clustering (bottom) for the self-similarities search with, from left to right, the thresholds $\varepsilon_M = 3.0, 5.0, 10.0, 15.0$ and the parameter $\alpha_A = 0.5$.

Figure B.2: Reconstructed images corresponding to the epitomes from the previous Figure.

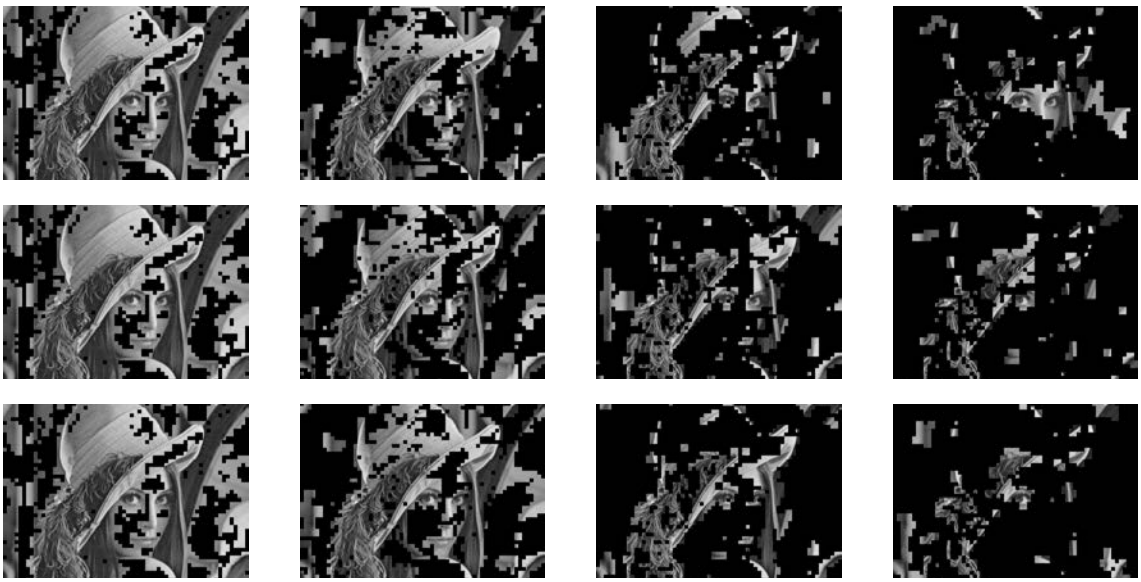


Figure B.3: Epitomes extracted from Lena. The epitome was obtained using the full search method (top), the list-based method (middle) and the threshold-based clustering (bottom) for the self-similarities search with, from left to right, the thresholds $\varepsilon_M = 3.0, 5.0, 10.0, 15.0$ and the parameter $\alpha_A = 0.5$.

Figure B.4: Reconstructed images corresponding to the epitomes from the previous Figure.

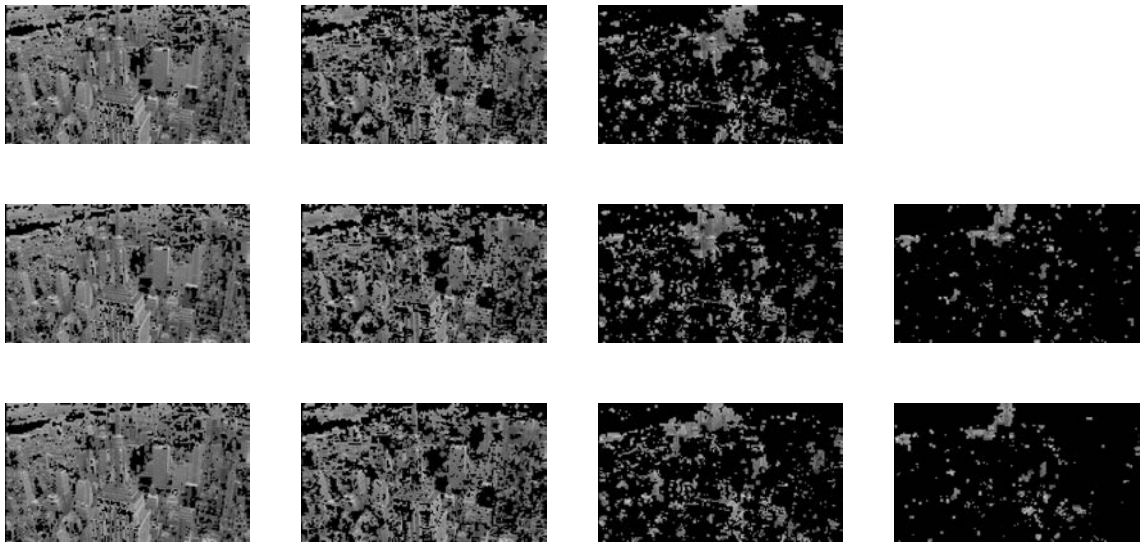


Figure B.5: Epitomes extracted from City. The epitome was obtained using the full search method (top), the list-based method (middle) and the threshold-based clustering (bottom) for the self-similarities search with, from left to right, the thresholds $\varepsilon_M = 3.0, 5.0, 10.0, 15.0$ and the parameter $\alpha_A = 0.5$. (Full search with $\varepsilon_M = 15.0$ could not be produced).

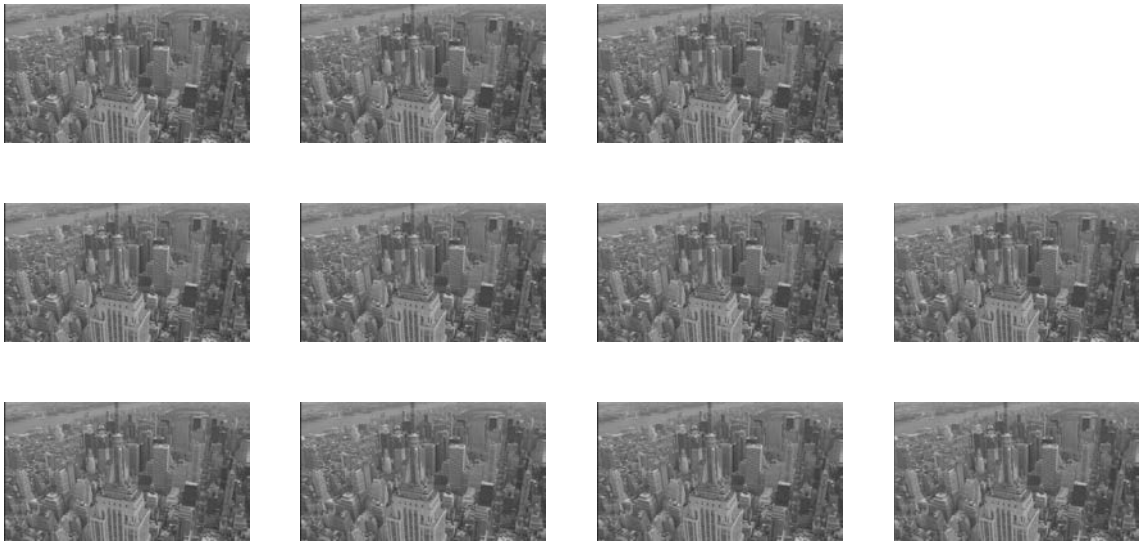


Figure B.6: Reconstructed images corresponding to the epitomes from the previous Figure.

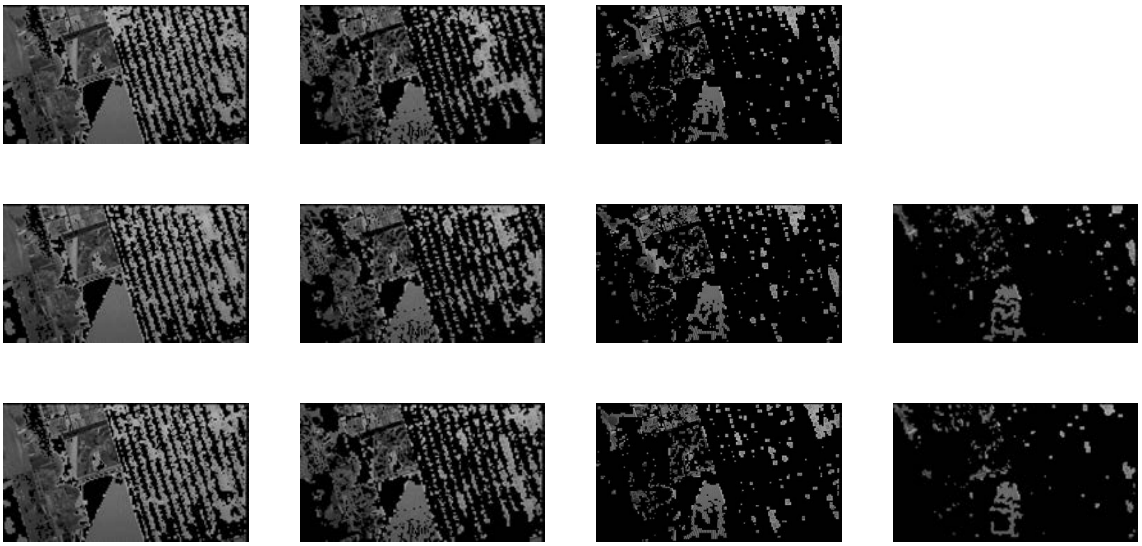


Figure B.7: Epitomes extracted from Calendar. The epitome was obtained using the full search method (top), the list-based method (middle) and the threshold-based clustering (bottom) for the self-similarities search with, from left to right, the thresholds $\varepsilon_M = 3.0, 5.0, 10.0, 15.0$ and the parameter $\alpha_A = 0.5$. (Full search with $\varepsilon_M = 15.0$ could not be produced).

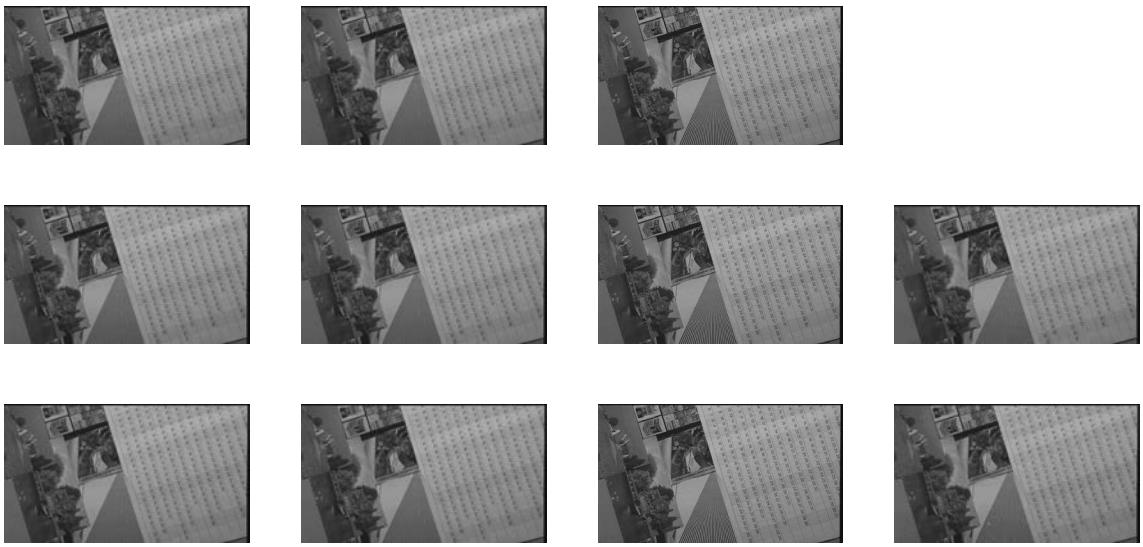


Figure B.8: Reconstructed images corresponding to the epitomes from the previous Figure.

B.2 Epitomes for epitome-based de-noising

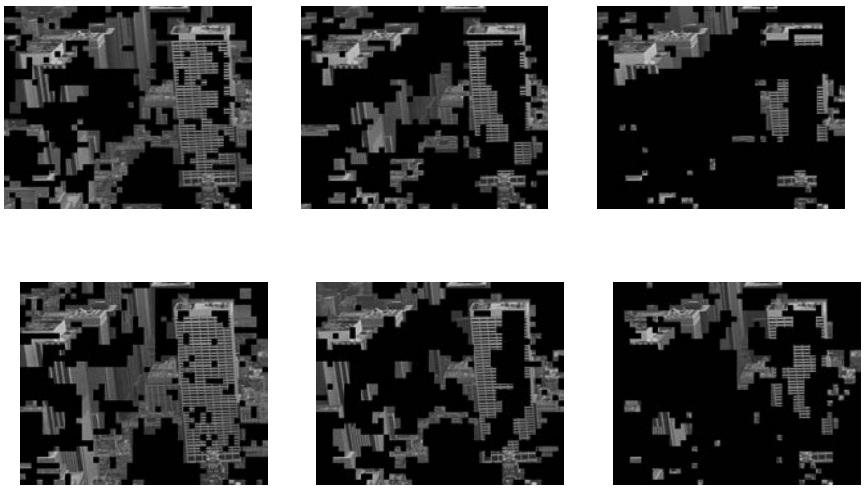


Figure B.9: Epitomes extracted from the first frame (top) of the City first GOP, and first frame (bottom) of the following GOP. The epitomes were obtained, from left to right, with the thresholds $\varepsilon_M = 7.0, 10.0, 15.0$.

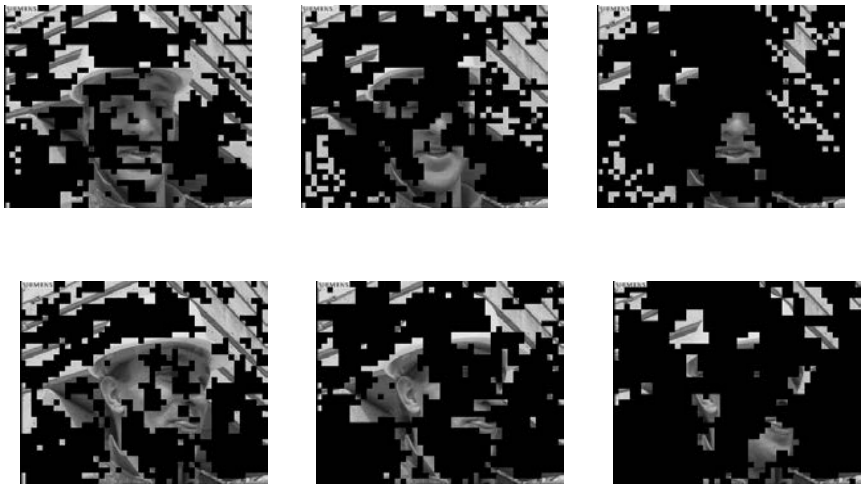


Figure B.10: Epitomes extracted from the first frame (top) of the Foreman first GOP, and first frame (bottom) of the following GOP. The epitomes were obtained, from left to right, with the thresholds $\varepsilon_M = 7.0, 10.0, 15.0$.

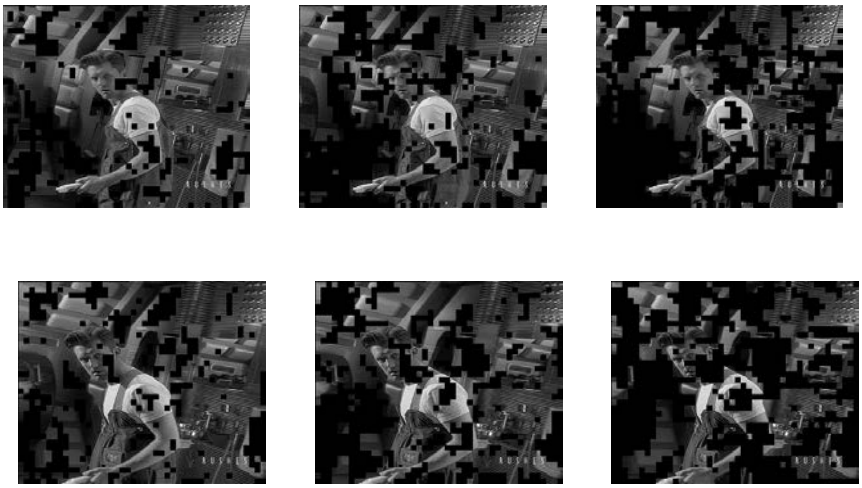


Figure B.11: Epitomes extracted from the first frame (top) of the Macleans first GOP, and first frame (bottom) of the following GOP. The epitomes were obtained, from left to right, with the thresholds $\varepsilon_M = 7.0, 10.0, 15.0$.

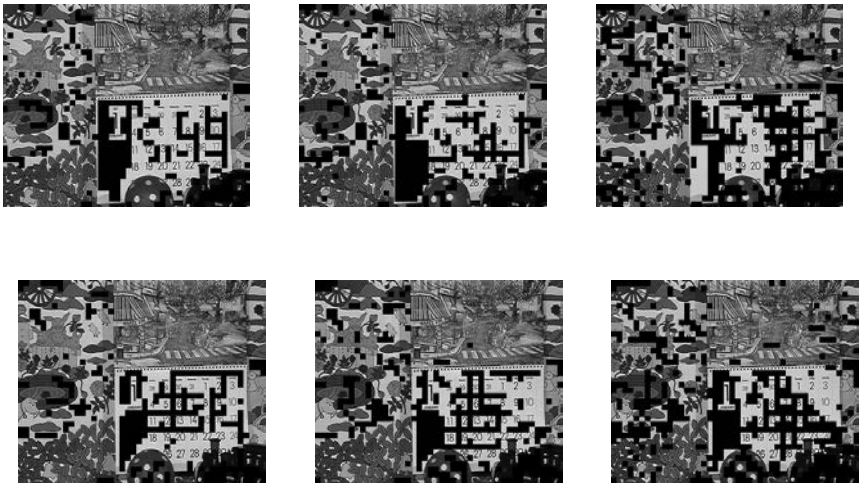


Figure B.12: Epitomes extracted from the first frame (top) of the Mobile first GOP, and first frame (bottom) of the following GOP. The epitomes were obtained, from left to right, with the thresholds $\varepsilon_M = 7.0, 10.0, 15.0$.

Appendix C

Appendix of chapter 4

C.1 De-noising

C.1.1 Results on AWGN and SDN

Table C.1: De-noising performances of K-means ($K = 10$) clustering-based linear mapping learning and BM3D for AWGN

Image	PSNR (dB) Noisy	K-means LM	NLM	BM3D		PSNR (dB) Noisy	K-means LM	NLM	BM3D
	$\sigma = 10$					$\sigma = 20$			
City	28.1328	32.2293	32.4902	34.5002		22.1142	28.3608	29.0843	30.7206
Park Scene	28.1322	33.8874	33.6864	35.6443		22.2235	30.2797	29.7379	32.4186
Tennis	28.1395	35.5153	36.0364	38.7244		22.3378	31.7996	30.8613	35.8702
Kimono	28.1327	36.1916	36.3595	39.3243		22.1961	32.2559	30.7709	36.0030
Cactus	28.1585	32.5164	33.7374	35.6019		22.2537	29.0135	29.9194	32.6742
Terrace	28.1504	31.5588	30.9521	34.3474		22.3295	27.7119	28.5340	30.9289
Basket	28.1327	34.2790	35.7667	37.9416		22.1257	30.3038	30.6961	35.3918
Ducks	28.1313	32.5807	33.1584	34.0325		22.1235	28.9979	29.7445	31.2881
People On Street	28.1428	33.5672	34.5368	36.6175		22.2364	29.6242	30.4280	33.3369
Traffic	28.1397	33.5563	34.8621	37.0335		22.2184	29.6867	30.3450	33.4725
Average	28.1393	33.5882	34.1586	36.3768		22.2159	29.8034	30.0121	33.2105
	$\sigma = 30$					$\sigma = 40$			
City	18.6097	26.2537	26.2798	28.6892		16.1778	24.8783	24.1381	27.2337
Park Scene	18.9712	28.1970	26.9487	30.6751		16.7723	26.6645	24.9175	29.3104
Tennis	19.1523	29.4058	27.7421	33.8289		16.9662	27.7177	25.5186	31.8626
Kimono	18.8934	29.7764	27.5097	33.8796		16.6812	27.9399	25.2814	32.1755
Cactus	18.8952	27.0749	27.0251	30.7290		16.5915	25.6119	24.8450	29.0974
Terrace	19.0742	25.6968	26.2739	28.7810		16.8372	24.2602	24.3995	26.9201
Basket	18.6307	28.2614	27.3983	33.6251		16.1918	26.6241	25.0040	32.2124
Ducks	18.6614	26.8607	26.8084	29.5300		16.2970	25.2968	24.5493	28.1159
People On Street	18.8775	27.3763	27.3785	31.2774		16.5746	25.6033	25.0892	29.4196
Traffic	18.8560	27.4904	27.2278	31.3427		16.5451	25.9054	24.9518	29.6122
Average	18.8622	27.6393	27.0592	31.2358		16.5635	26.0502	24.8694	29.5960

Table C.2: De-noising performances of K-means ($K = 10$) clustering-based linear mapping learning and BM3D for SDN

Image	PSNR (dB) Noisy	K-means LM	NLM	BM3D		PSNR (dB) Noisy	K-means LM	NLM	BM3D
	$\gamma = 0.5, \sigma_u = 1.5, \sigma_w = 10$					$\gamma = 0.5, \sigma_u = 3, \sigma_w = 10$			
City	22.6268	28.6189	24.5089	30.9236		17.6666	25.7233	21.2923	27.9811
Park Scene	24.1703	31.2408	27.5703	33.0157		19.7954	28.6291	24.8378	30.0556
Tennis	24.3593	32.8574	28.3005	35.6816		20.1024	30.0329	25.3632	31.6169
Kimono	24.0531	33.4449	28.0428	36.1713		19.6640	30.3713	25.1521	32.5319
Cactus	22.8072	29.3197	25.0625	32.5979		18.0620	26.5843	21.8379	29.4978
Terrace	22.1833	27.6801	23.6599	30.6325		17.5481	24.6939	20.8275	27.1673
Basket	22.3065	30.5848	24.7466	35.3905		17.2799	27.4412	21.0410	32.7217
Ducks	22.5534	29.2010	24.2460	31.2006		17.6747	26.1324	20.9908	28.4948
People On Street	22.8686	29.9248	25.1835	33.2473		18.0969	26.5102	21.9365	29.7624
Traffic	23.3136	30.3467	26.0561	33.7029		18.5966	27.3527	22.8029	30.3626
Average	23.1242	30.3219	25.7377	33.2564		18.4487	27.3471	22.6082	30.0192
	$\gamma = 0.6, \sigma_u = 1.5, \sigma_w = 5$					$\gamma = 0.7, \sigma_u = 0.5, \sigma_w = 5$			
City	19.7707	26.8971	26.6070	29.1116		24.7772	29.9194	30.8604	32.0709
Park Scene	22.3302	30.0600	29.3638	30.9035		27.3272	32.9692	33.2040	33.9047
Tennis	22.6288	31.6209	30.4743	32.1414		27.5478	34.6473	35.4521	35.5719
Kimono	22.1677	32.1307	30.0747	32.9185		27.1288	35.3970	35.3214	36.0787
Cactus	20.0310	27.7356	27.1652	30.4155		24.7616	30.5537	31.7316	32.9300
Terrace	19.2221	25.8385	25.6659	28.3450		23.5627	28.7434	29.2466	31.2265
Basket	19.2660	28.6660	26.9527	33.6864		24.2184	31.9881	32.4408	36.0480
Ducks	19.6658	27.4106	26.7592	29.3765		24.5538	30.4115	31.4003	31.7627
People On Street	20.1125	28.0206	27.5219	30.8932		21.5625	29.1066	32.3965	33.6299
Traffic	20.8035	28.7525	28.3148	31.3647		25.7511	31.9347	33.2198	34.4228
Average	20.5998	28.7132	27.8899	30.9156		25.1191	31.5671	32.5274	33.7646

Table C.3: De-noising performances of optimal clustering-based linear mapping learning and BM3D for AWGN

Image	PSNR (dB) Noisy	Optimal clustering LM	BM3D		PSNR (dB) Noisy	Optimal clustering LM	BM3D
		$\sigma = 10$				$\sigma = 20$	
City	28.1328	34.4698	34.5002		22.1142	31.1448	30.7206
Park Scene	28.1322	36.0456	35.6443		22.2235	33.2929	32.4186
Tennis	28.1395	38.2799	38.7244		22.3378	35.8716	35.8702
Kimono	28.1327	39.4760	39.3243		22.1961	37.0670	36.0030
Cactus	28.1585	35.4238	35.6019		22.2537	32.5053	32.6742
Terrace	28.1504	33.8943	34.3474		22.3295	30.3751	30.9289
Basket	28.1327	37.8205	37.9416		22.1257	35.3943	35.3918
Ducks	28.1313	34.3445	34.0325		22.1235	31.6280	31.2881
People On Street	28.1428	36.3971	36.6175		22.2364	33.0578	33.3369
Traffic	28.1397	36.6627	37.0335		22.2184	33.4188	33.4725
Average	28.1393	36.2814	36.3768		22.2159	33.3756	33.2105
		$\sigma = 30$				$\sigma = 40$	
City	18.6097	29.4775	28.6892		16.1778	28.4646	27.2337
Park Scene	18.9712	31.9506	30.6751		16.7723	31.2015	29.3104
Tennis	19.1523	34.5894	33.8289		16.9662	33.8045	31.8626
Kimono	18.8934	35.7702	33.8796		16.6812	35.0216	32.1755
Cactus	18.8952	30.9864	30.7290		16.5915	29.9341	29.0974
Terrace	19.0742	28.5569	28.7810		16.8372	27.4634	26.9201
Basket	18.6307	33.9598	33.6251		16.1918	32.9549	32.2124
Ducks	18.6614	30.1503	29.5300		16.2970	29.2063	28.1159
People On Street	18.8775	31.1907	31.2774		16.5746	30.0579	29.4196
Traffic	18.8560	31.7608	31.3427		16.5451	30.6347	29.6122
Average	18.8622	31.8393	31.2358		16.5635	30.8744	29.5960

Table C.4: De-noising performances of optimal clustering-based linear mapping learning and BM3D for SDN

Image	PSNR (dB) Noisy	Optimal clustering LM	BM3D		PSNR (dB) Noisy	Optimal clustering LM	BM3D
	$\gamma = 0.5, \sigma_u = 1.5, \sigma_w = 10$				$\gamma = 0.5, \sigma_u = 3, \sigma_w = 10$		
City	22.6268	31.3827	30.9236		17.6666	29.1100	27.9811
Park Scene	24.1703	34.0137	33.0157		19.7954	32.2189	30.0556
Tennis	24.3593	36.5447	35.6816		20.1024	34.8505	31.6169
Kimono	24.0531	37.8658	36.1713		19.6640	36.1711	32.5319
Cactus	22.8072	32.9804	32.5979		18.0620	30.8396	29.4978
Terrace	22.1833	30.4575	30.6325		17.5481	27.9752	27.1673
Basket	22.3065	35.6551	35.3905		17.2799	33.6653	32.7217
Ducks	22.5534	31.8332	31.2006		17.6747	29.8164	28.4948
People On Street	22.8686	33.4185	33.2473		18.0969	30.9121	29.7624
Traffic	23.3136	34.1739	33.7029		18.5966	31.9198	30.3626
Average	23.1242	33.8326	33.2564		18.4487	31.7479	30.0192
	$\gamma = 0.6, \sigma_u = 1.5, \sigma_w = 5$				$\gamma = 0.7, \sigma_u = 0.5, \sigma_w = 5$		
City	19.7707	30.0145	29.1116		24.7772	32.5389	32.0709
Park Scene	22.3302	33.1917	30.9035		27.3272	35.3929	33.9047
Tennis	22.6288	35.8497	32.1414		27.5478	37.8144	35.5719
Kimono	22.1677	37.1747	32.9185		27.1288	39.1840	36.0787
Cactus	20.0310	31.7938	30.4155		24.7616	34.0870	32.9300
Terrace	19.2221	28.9016	28.3450		23.5627	31.4077	31.2265
Basket	19.2660	34.5423	33.6864		24.2184	36.4812	36.0480
Ducks	19.6658	30.6962	29.3765		24.5538	32.7761	31.7627
People On Street	20.1125	31.9777	30.8932		21.5625	34.5815	33.6299
Traffic	20.8035	33.0412	31.3647		25.7511	35.6138	34.4228
Average	20.5998	32.7183	30.9156		25.1191	34.9877	33.7646

C.1.2 RD curves for optimal clustering

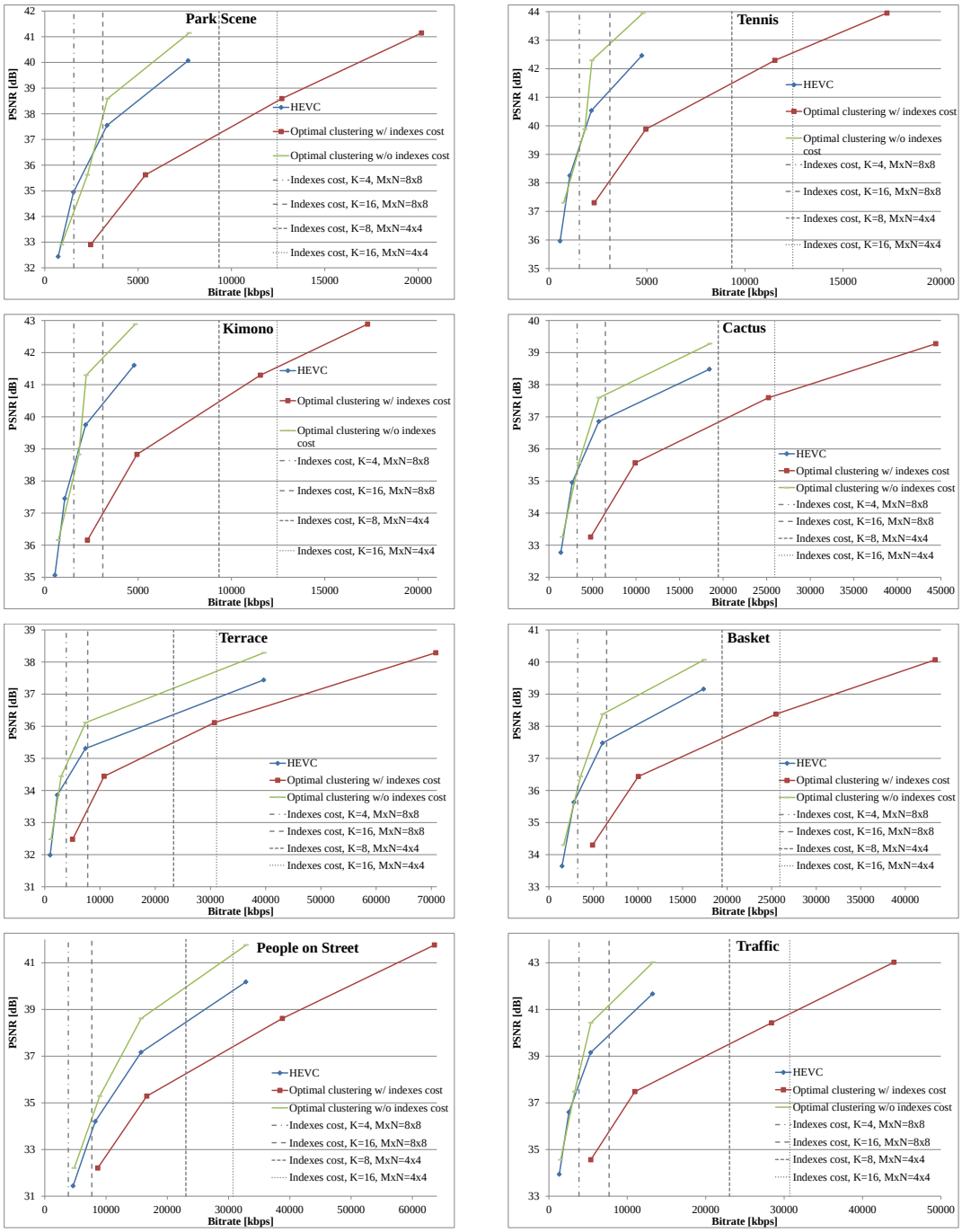


Figure C.1: RD performance of the test sequences using the optimal clustering, with and without the cluster indexes rate-cost

C.2 Super-resolution

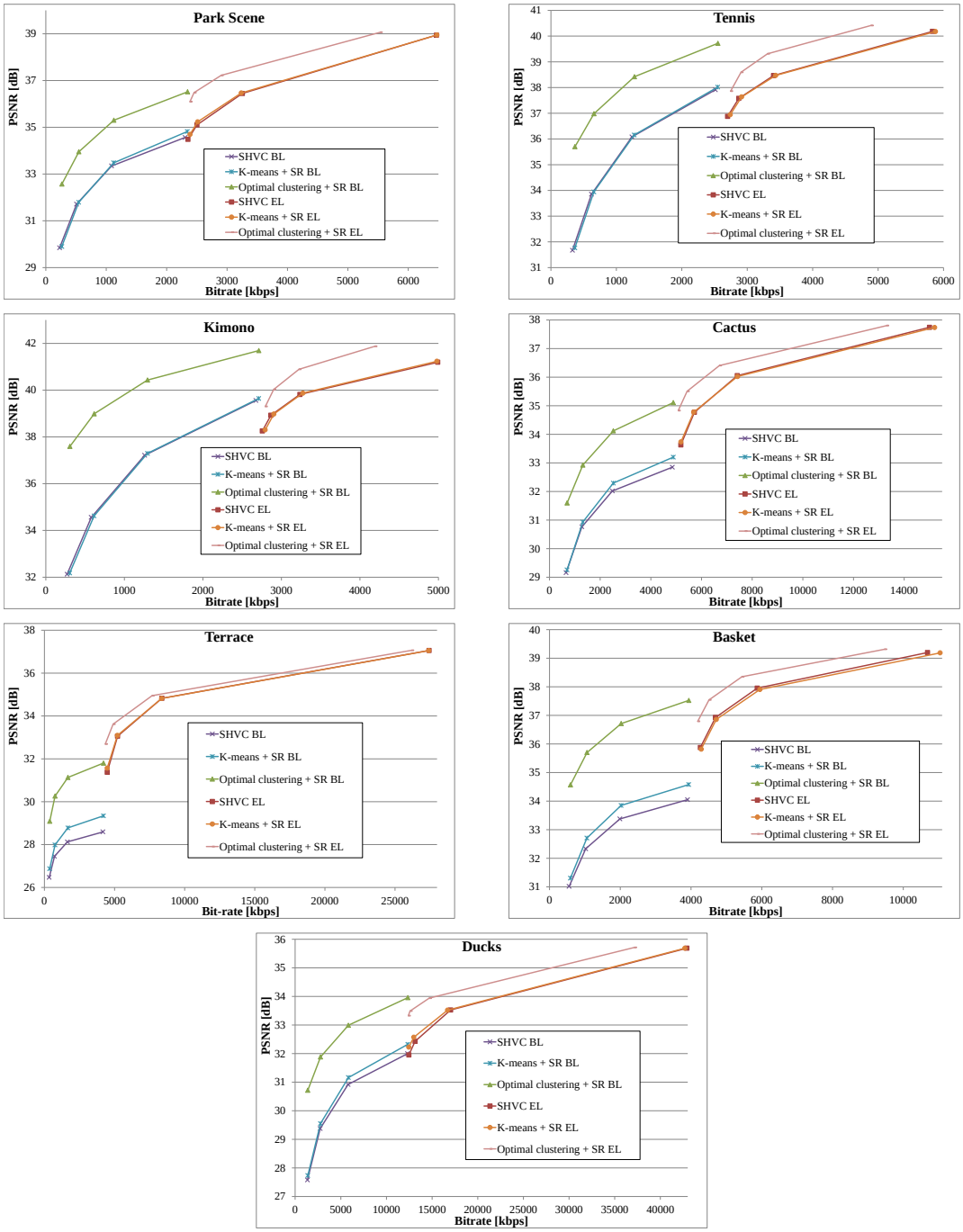


Figure C.2: RD performance of the test sequences using the K-means and optimal clustering for super-resolution

C.3 Classification

C.3.1 Test images

Figure C.3: 200 × 200 test images used to evaluate the classification approach performance. From left to right, top to bottom: City, Basketball, Park Scene, Ducks, Kimono, People on street, Cactus, Traffic, Terrace.

C.3.2 Classification performances

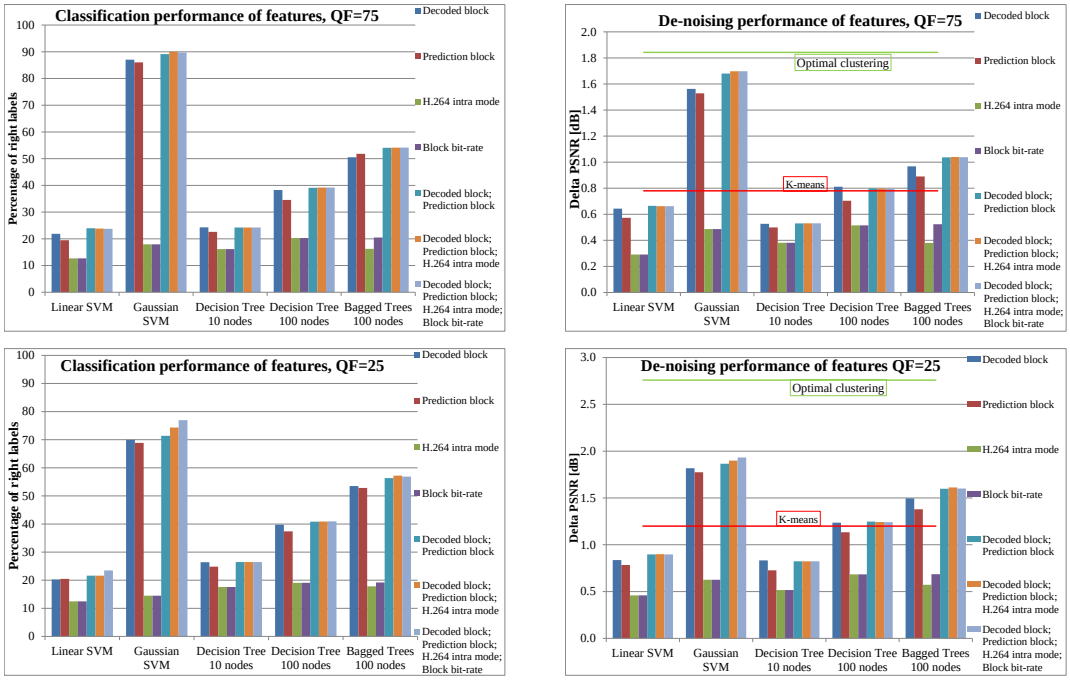


Figure C.4: Average classification (left) and de-noising (right) performances with different features and combination of features, with images encoded at $QF = 75$ (top) and $QF = 25$ (bottom).

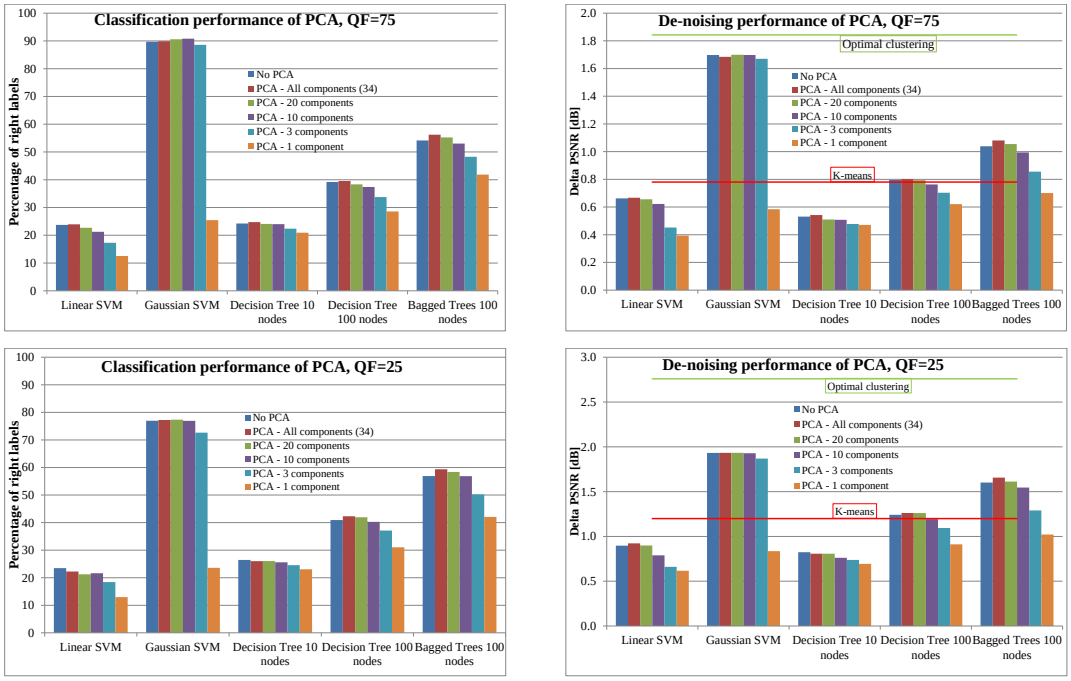


Figure C.5: Average classification (left) and de-noising (right) performances with different dimensions of a PCA performed on the combination of features, with images encoded at $QF = 75$ (top) and $QF = 25$ (bottom).

C.3.3 Classifications performances in a compression context

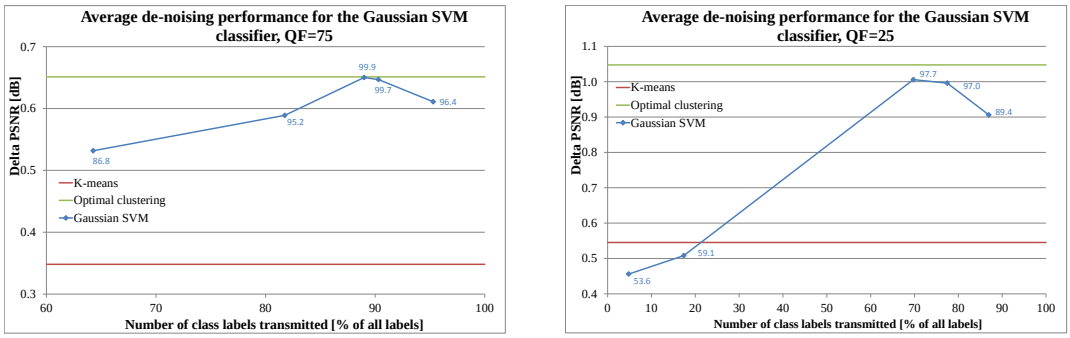


Figure C.6: Average performances of the Gaussian SVM depending on the percentage of support vectors, with $QF = 75$ (left) and $QF = 25$ (right).

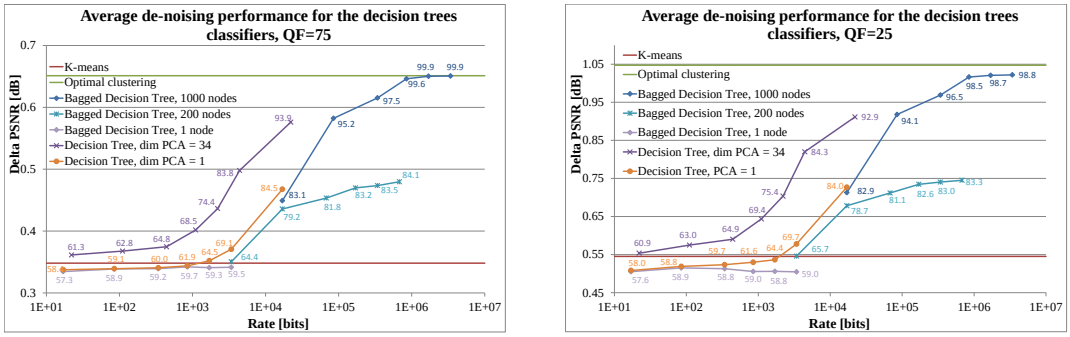


Figure C.7: Average performances of the decision trees, with $QF = 75$ (left) and $QF = 25$ (right).

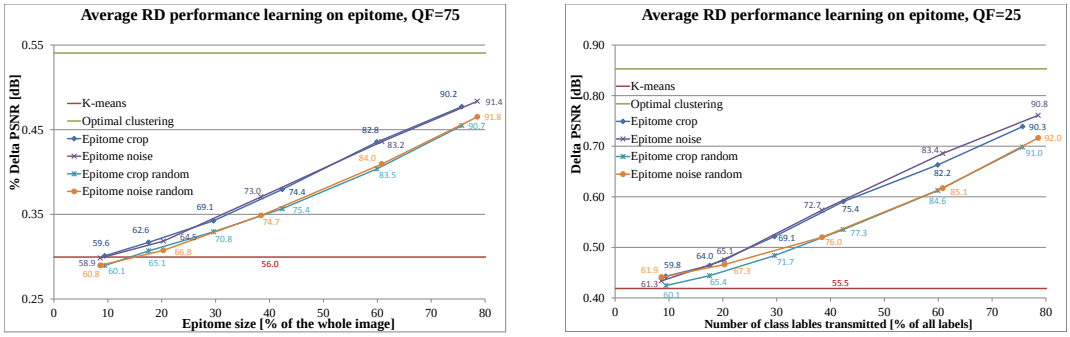


Figure C.8: Average performances of the bagged decision trees depending on the size of the epitome, with $QF = 75$ (left) and $QF = 25$ (right).

C.3.4 RD performances

C.3.4.1 De-noising

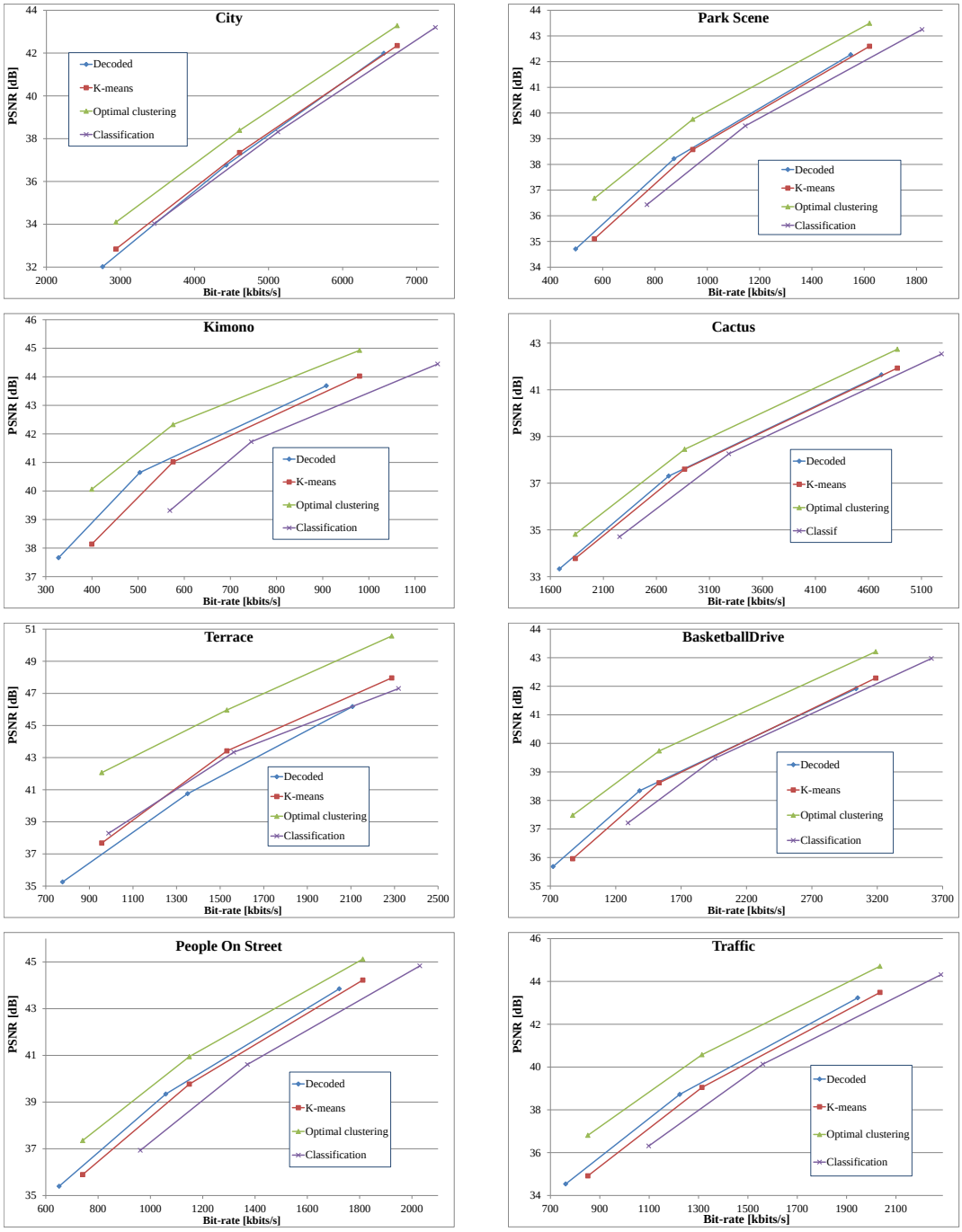


Figure C.9: RD performances with recovering of the optimal class labels using classification.

C.3.4.2 Super-resolution

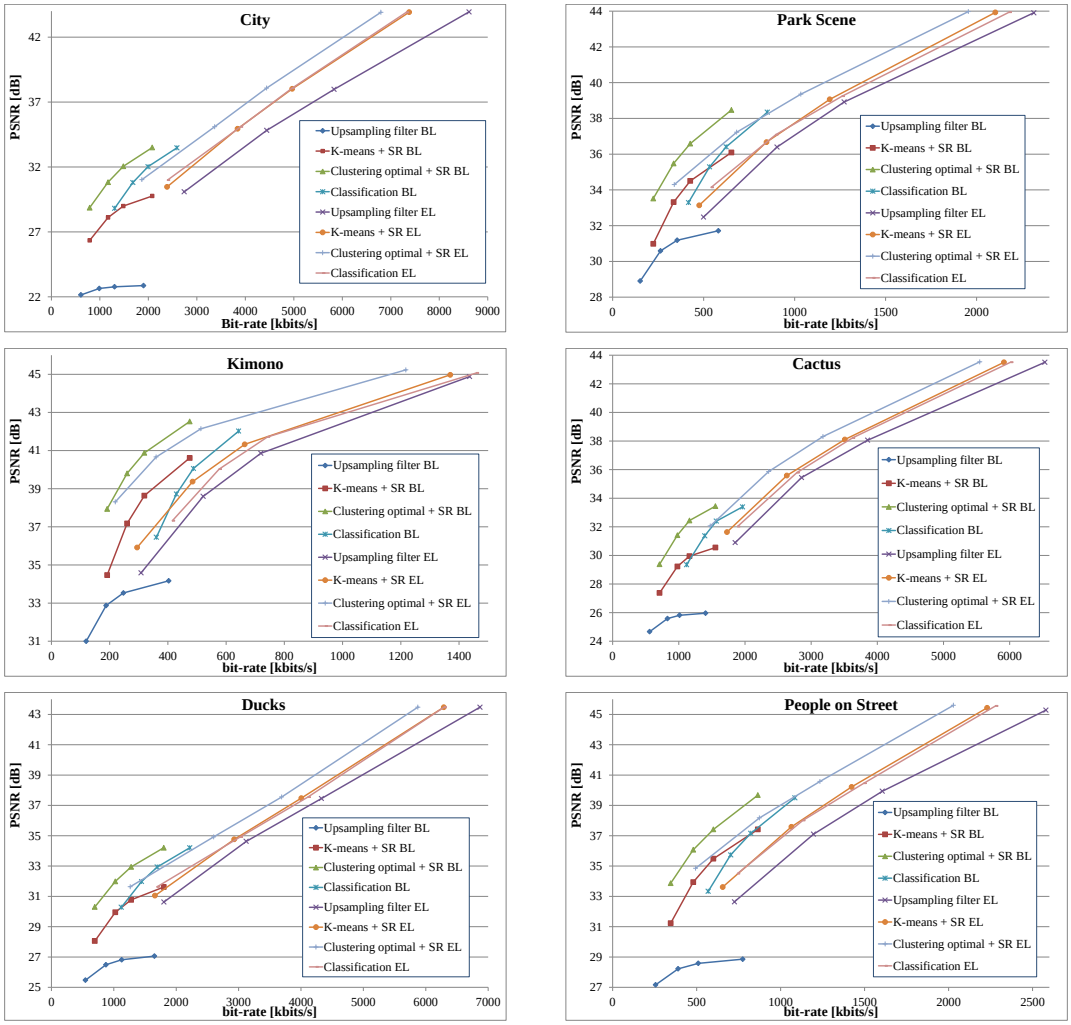


Figure C.10: RD performances with recovering of the optimal class labels using classification.

Glossary

AVC: Advanced Video Coding
(A)WGN: (Additive) White Gaussian Noise
BL: Base Layer
BM: Block Matching
CABAC: Context Adaptive Binary Arithmetic Coding
CU: Coding Unit
CTU: Coding Tree Unit
DCT: Discrete Cosinus Transform
DST: Discrete Sinus Transform
DWT: Discrete Wavelet Transform
EL: Enhancement Layer
GOP: Group Of Pictures
HEVC: High Efficiency Video Coding
HR: High Resolution
i.i.d: Independent and Identically Distributed
ITU: International Telecommunications Union
JCT-VC: Joint Collaborative Team on Video Coding
JPEG: Joint Photographic Experts Group
KF: Key Frame
LDA: Linear Discriminant Analysis
LLE: Locally Linear Embedding
LLM: Local Linear Mappings
LR: Low Resolution
MB: Macroblock
MCP: Motion Compensation Prediction
MPM: Most Probable Mode
MSE: Mean Square Error
MV: Motion Vector
MVP: Motion Vector Predictor
NLM: Non-Local Mean
NMF: Non-negative Matrix Factorization
PCA: Principal Component Analysis
(P)SNR: (Peak) Signal-to-Noise Ratio
PU: Prediction Unit

RD(O):	Rate Distortion (Optimization)
SAD:	Sum of Absolute Distance
SAE:	Sum of Absolute Error
SPIHT:	Set Partitioning In Hierarchical Trees
SSE:	Sum of Square Error
TM(A):	Template Matching (Averaging)
TU:	Transform Unit
VLC:	Variable Length Code

Bibliography

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003. (Cited on pages 8, 18, 22, 23, 25, and 45.)

[2] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012. (Cited on pages 8, 18, 22, 28, and 31.)

[3] G. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, 1992. (Cited on pages 8 and 22.)

[4] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, jun 1996. (Cited on pages 9 and 22.)

[5] M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "An overview of JPEG-2000," in *Data Compression Conference*, pp. 523–541, IEEE Comput. Soc, 2000. (Cited on pages 9 and 22.)

[6] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and Choong Seng Boon, "Inter frame coding with template matching spatio-temporal prediction," in *IEEE International Conference on Image Processing*, vol. 1, pp. 465–468, IEEE, 2004. (Cited on pages 10, 35, and 37.)

[7] T. K. T. T. K. Tan, C. S. B. C. S. Boon, and Y. S. Y. Suzuki, "Intra Prediction by Template Matching," *IEEE International Conference on Image Processing*, pp. 1–4, 2006. (Cited on pages 10, 35, and 37.)

[8] T. K. Tan, C. S. Boon, and Y. Suzuki, "Intra Prediction by Averaged Template Matching Predictors," in *IEEE Consumer Communications and Networking Conference*, pp. 405–409, IEEE, jan 2007. (Cited on pages 10 and 35.)

[9] Y. Suzuki, C. S. B. C. S. Boon, and T. K. T. T. K. Tan, "Inter Frame Coding with Template Matching Averaging," *IEEE International Conference on Image Processing*, vol. 3, pp. 409–412, 2007. (Cited on pages 10, 35, and 37.)

[10] A. Martin, J. J. Fuchs, C. Guillemot, and D. Thoreau, "Sparse representation for image prediction," *European Signal Processing Conference*, pp. 1255–1259, 2007. (Cited on pages 10, 35, and 38.)

[11] M. Türkan and C. Guillemot, "Sparse approximation with adaptive dictionary for image prediction," in *IEEE International Conference on Image Processing*, pp. 25–28, 2009. (Cited on pages 10, 35, and 38.)

[12] M. Türkan and C. Guillemot, "Image prediction based on neighbor embedding methods.," *IEEE Transactions on Image Processing*, vol. 21, no. X, pp. 1885–98, 2012. (Cited on pages 10, 35, 38, and 57.)

[13] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, and P. Perez, "Correspondence Map-Aided Neighbor Embedding for Image Intra Prediction," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 1161–1174, 2013. (Cited on pages 10, 35, 38, 50, and 57.)

[14] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975. (Cited on pages 11, 45, 54, 71, and 157.)

[15] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized PatchMatch correspondence algorithm," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6313 LNCS, pp. 29–43, 2010. (Cited on pages 11, 45, 54, 71, and 157.)

[16] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "2-D Dictionary Based Video Coding for Screen Contents," *Data Compression Conference*, pp. 43–52, 2014. (Cited on pages 11 and 54.)

[17] A. Cherian, S. Sra, V. Morellas, and N. Papanikolopoulos, "Efficient Nearest Neighbors via Robust Sparse Hashing," *IEEE Transactions on Image Processing*, vol. 23, no. c, pp. 3646–3655, 2014. (Cited on pages 11 and 54.)

[18] E. Wige, G. Yammine, P. Amon, A. Hutter, and A. Kaup, "Sample-based weighted prediction with directional template matching for HEVC lossless coding," *Picture Coding Symposium*, pp. 305–308, 2013. (Cited on pages 11, 38, 55, and 157.)

[19] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using GPU," *Conference on Computer Vision and Pattern Recognition Workshops*, no. 2, pp. 1–6, 2008. (Cited on pages 11, 55, and 157.)

[20] N. Jojic, B. Frey, and A. Kannan, "Epitomic analysis of appearance and shape," *IEEE International Conference on Computer Vision*, 2003. (Cited on pages 11, 57, and 58.)

[21] V. Cheung, B. J. Frey, and N. Jojic, "Video epitomes," *International Journal of Computer Vision*, vol. 76, no. c, pp. 141–152, 2008. (Cited on pages 11, 57, and 58.)

[22] H. Wang, Y. Wexler, E. Ofek, and H. Hoppe, "Factoring repeated content within and among images," in *ACM Transactions on Graphics*, vol. 27, p. 1, 2008. (Cited on pages 11 and 57.)

[23] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, and P. Perez, "Epitome-based image compression using translational sub-pel mapping," in *IEEE International Workshop on Multimedia Signal Processing*, 2011. (Cited on pages 11, 12, 57, 62, 63, and 64.)

[24] S. P. Ghael, A. M. Sayeed, and R. G. Baraniuk, "Improved wavelet denoising via empirical Wiener filtering," in *Proceedings of SPIE*, pp. 389–399, oct 1997. (Cited on pages 11 and 66.)

[25] D. L. Donoho, "De-noising by soft thresholding," *Symposium on Wavelet Theory*, vol. 76, p. 1, 1992. (Cited on pages 11 and 66.)

[26] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995. (Cited on pages 11 and 66.)

[27] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 60–65, 2005. (Cited on pages 11 and 66.)

[28] A. Buades, B. Coll, and J. M. Morel, "A Review of Image Denoising Algorithms, with a New One," *Multiscale Modeling & Simulation*, vol. 4, pp. 490–530, jan 2005. (Cited on pages 11 and 66.)

[29] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007. (Cited on pages 12, 67, 85, and 118.)

[30] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," *5th Berkeley Symposium on Mathematical Statistics and Probability 1967*, vol. 1, pp. 281–297, 1967. (Cited on pages 13 and 102.)

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. (Cited on pages 13 and 105.)

[32] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, (New York, New York, USA), pp. 144–152, ACM Press, 1992. (Cited on pages 13 and 107.)

[33] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. 19. 1984. (Cited on pages 13 and 108.)

[34] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996. (Cited on pages 13 and 110.)

[35] 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)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012. (Cited on pages 18, 28, and 55.)

[36] "VP9 Video Codec Summary. www.webmproject.org/vp9." (Cited on page 18.)

[37] "Thor draft. https://tools.ietf.org/html/draft-fuldseth-netvc-thor." (Cited on page 18.)

[38] "Daala project. http://www.xiph.org/daala/," (Cited on page 18.)

[39] M. Schroeder, "Transform coding of image difference signals," July 25 1972. US Patent 3,679,821. (Cited on page 18.)

[40] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, vol. 29, no. 12, 1981. (Cited on page 18.)

[41] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974. (Cited on page 18.)

[42] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, 1952. (Cited on page 18.)

[43] J. J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding," *IBM Journal of Research and Development*, vol. 20, no. 3, pp. 198–203, 1976. (Cited on page 18.)

[44] "PROVISION project. http://www.provision-itn.eu," (Cited on page 18.)

[45] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998. (Cited on page 28.)

[46] A. Ortego and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998. (Cited on page 28.)

[47] Y. Ye and P. Andrivon, "The Scalable Extensions of HEVC for Ultra-High-Definition Video Delivery," *IEEE MultiMedia*, vol. 21, no. 3, pp. 58–64, 2014. (Cited on pages 32, 33, 134, and 135.)

[48] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, pp. 2323–2326, dec 2000. (Cited on page 35.)

[49] S. Kamp, M. Evertz, and M. Wien, "Decoder side motion vector derivation for inter frame video coding," in *IEEE International Conference on Image Processing*, pp. 1120–1123, 2008. (Cited on page 37.)

[50] F. Bossen, V. Drugeon, E. Francois, J. Jung, S. Kanumuri, M. Narroschke, H. Sasai, J. Sole, Y. Suzuki, T. K. Tan, T. Wedi, S. Wittmann, P. Yin, and Y. Zheng, "Video coding using a simplified block structure and advanced coding techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1667–1675, 2010. (Cited on page 37.)

[51] W. H. Peng and C. C. Chen, "An interframe prediction technique combining template matching prediction and block-motion compensation for high-efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 8, pp. 1432–1446, 2013. (Cited on pages 37 and 55.)

[52] Y. Shen, J. Li, and Z. Zhu, "Motion estimation for video coding based on sparse representation," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1394–1398, 2013. (Cited on page 38.)

[53] "KTA Software, Ver. JM 14.2 KTA 1.0. Available: http://iphome.hhi.de/suehring/tml/download/KTA/." (Cited on page 44.)

[54] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *document VCEG-M33, ITU-T VCEG Meeting*, 2001. (Cited on pages 45 and 121.)

[55] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, and P. Perez, "Map-Aided Locally Linear Embedding methods for image prediction," *IEEE International Conference on Image Processing*, pp. 2909–2912, 2012. (Cited on page 57.)

[56] Q. Wang, R. Hu, and Z. Wang, "Improving intra coding in H.264\AVC by image epitome," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5879 LNCS, pp. 190–200, 2009. (Cited on pages 57 and 59.)

[57] S. Cherigui, M. Alain, C. Guillemot, D. Thoreau, and P. Guillotel, "Epitome inpainting with in-loop residue coding for image compression," in *IEEE International Conference on Image Processing*, vol. 2014, pp. 5581–5585, 2014. (Cited on pages 57, 58, and 159.)

[58] Q. Wang, R. Hu, Z. Wang, and B. Hang, "Intra coding and refresh based on video epitomic analysis," in *IEEE International Conference on Multimedia and Expo*, pp. 452–455, IEEE, jul 2010. (Cited on pages 58 and 59.)

[59] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani, "Summarizing visual data using bidirectional similarity," *Conference on Computer Vision and Pattern Recognition*, no. ii, pp. 1–8, 2008. (Cited on page 59.)

[60] Q. Wang, R. Hu, and Z. Wang, "Intracoding and refresh with compression-oriented video epitomic priors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 714–726, 2012. (Cited on page 59.)

[61] M. Aharon and M. Elad, "Sparse and Redundant Modeling of Image Content Using an Image-Signature-Dictionary," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 228–247, 2008. (Cited on page 59.)

[62] H. Wang, Y. Wexler, E. Ofek, and H. Hoppe, "Factoring repeated content within and among images," *ACM Transactions on Graphics*, vol. 27, p. 1, 2008. (Cited on pages 59, 61, and 62.)

[63] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th international joint Conference on Artificial Intelligence*, no. x, pp. 674–679, 1981. (Cited on page 60.)

[64] J. Shi, C. Tomasi, Jianbo Shi, and Tomasi, "Good features to track," in *Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994. (Cited on page 60.)

[65] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive Noise Smoothing Filter for Images with Signal-Dependent Noise," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, pp. 165–177, mar 1985. (Cited on page 65.)

[66] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 299–314, 2008. (Cited on page 65.)

[67] X. Liu, M. Tanaka, and M. Okutomi, "Signal dependent noise removal from a single image," in *IEEE International Conference on Image Processing*, no. 2, pp. 2679–2683, 2014. (Cited on pages 65, 101, and 118.)

[68] M. Elad and M. Aharon, "Image denoising via sparse and redundant representation over learned dictionaries," *IEEE Transations on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006. (Cited on page 65.)

[69] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006. (Cited on page 65.)

[70] J. Sulam, B. Ophir, and M. Elad, "Image denoising through multi-scale learnt dictionaries," in *IEEE International Conference on Image Processing*, pp. 808–812, 2014. (Cited on page 65.)

[71] P. Chatterjee and P. Milanfar, "Clustering-based denoising with locally learned dictionaries," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1438–1451, 2009. (Cited on pages 65 and 101.)

[72] W. Dong, X. Li, L. Zhang, and G. Shi, "Sparsity-based image denoising via dictionary learning and structural clustering," in *Conference on Computer Vision and Pattern Recognition*, pp. 457–464, 2011. (Cited on pages 65 and 101.)

[73] P. Chatterjee and P. Milanfar, "Is denoising dead?," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 895–911, 2010. (Cited on page 65.)

[74] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *International Conference on Computer Vision*, 1998. (Cited on page 66.)

[75] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color Image Denoising via Sparse 3D Collaborative Filtering with Grouping Constraint in Luminance-Chrominance Space," in *IEEE International Conference on Image Processing*, pp. I – 313–I – 316, IEEE, sep 2007. (Cited on page 70.)

[76] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1715–1728, 2012. (Cited on page 70.)

[77] A. Danielyan, R. Foi, V. Katkovnik, and K. Egiazarian, "Image and video super-resolution via spatially adaptive blockmatching filtering," *Proceedings of International Workshop on Local and Non-Local Approximation in Image Processing (LNLA)*, p. 8, 2008. (Cited on page 70.)

[78] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3D transform-domain collaborative filtering," *European Signal Processing Conference*, pp. 145–149, 2007. (Cited on pages 70 and 119.)

[79] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Transactions on Graphics*, vol. 28, p. 1, 2009. (Cited on page 71.)

[80] S. Bhatia, "Adaptive K-Means Clustering.," *FLAIRS Conference*, 2004. (Cited on pages 71 and 73.)

[81] O. G. Guleryuz, "A nonlinear loop filter for quantization noise removal in hybrid video compression," in *IEEE International Conference on Image Processing*, vol. 2, pp. 333–336, 2005. (Cited on page 85.)

[82] "HM Software, Ver. 15.0. Available: https://hevc.hhi.fraunhofer.de/trac/hevc/browser." (Cited on pages 89, 120, and 138.)

[83] "SHM Software, Ver. 9.0. Available: https://hevc.hhi.fraunhofer.de/trac/shvc/browser." (Cited on page 89.)

[84] Yan Lu, Wen Gao, and Feng Wu, "Sprite generation for frame-based video coding," in *IEEE International Conference on Image Processing*, vol. 1, pp. 473–476, IEEE, 2001. (Cited on pages 98 and 157.)

[85] Y. Lu, W. Gao, and F. Wu, "Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 5, pp. 394–405, 2003. (Cited on pages 98 and 157.)

[86] M. Turkan, M. Alain, D. Thoreau, P. Guillotel, and C. Guillemot, "Epitomic image factorization via neighbor-embedding," in *IEEE International Conference on Image Processing*, 2015. (Cited on pages 98, 157, and 159.)

[87] A. A. Dempster, N. N. Laird, and D. D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society Series B Methodological*, vol. 39, no. 1, pp. 1–38, 1977. (Cited on page 103.)

[88] S. Uw, A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in Neural Information Processing Systems 14*, pp. 849–856, 2001. (Cited on page 104.)

[89] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. (Cited on page 108.)

[90] G. J. Sullivan, J. M. Boyce, Y. Chen, J. R. Ohm, C. A. Segall, and A. Vetro, "Standardized extensions of high efficiency video coding (HEVC)," *IEEE Journal on Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1001–1016, 2013. (Cited on pages 117 and 138.)

[91] "HM RExt Software, Ver. HM 15.0 RExt 8.1. Available: https://hevc.hhi.fraunhofer.de/trac/hevc/browser." (Cited on pages 117, 120, and 138.)

[92] N. Pierazzo, M. Lebrun, M. Rais, J. Morel, and G. Facciolo, "Non-local dual image denoising," in *IEEE International Conference on Image Processing*, pp. 813–817, 2014. (Cited on page 118.)

[93] M. Lebrun, A. Buades, and J.-m. Morel, "Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm," *IPOL*, vol. 3, pp. 1–42, 2013. (Cited on page 118.)

[94] C. Knaus and M. Zwicker, "Dual-domain image denoising," in *IEEE International Conference on Image Processing*, no. 4, pp. 440–444, 2013. (Cited on page 118.)

[95] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 228–233, 2001. (Cited on page 125.)

[96] A. Ligeza, *Artificial Intelligence: A Modern Approach*, vol. 9. 1995. (Cited on page 152.)

[97] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, sep 1936. (Cited on page 152.)

[98] T. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, pp. 326–334, 1965. (Cited on page 152.)

[99] G. J. McLachlan, *Discriminant analysis and statistical pattern recognition.* 2004. (Cited on page 152.)

[100] N. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. (Cited on page 152.)

[101] L. Deng and D. Yu, *Deep Learning: Methods and Applications*, vol. 7. 2014. (Cited on page 152.)

[102] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu, "Cloud-Based Image Coding for Mobile Devices-Toward Thousands to One Compression," *IEEE Transactions on Multimedia*, vol. 15, pp. 845–857, jun 2013. (Cited on pages 153 and 158.)

[103] E. Luo, S. H. Chan, and T. Q. Nguyen, "Image denoising by targeted external databases," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2450–2454, 2014. (Cited on page 153.)

[104] H. Yue, X. Sun, J. Yang, and F. Wu, "Image Denoising by Exploring External and Internal Correlations," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1–1, 2015. (Cited on page 153.)

[105] Libin Sun and J. Hays, "Super-resolution from internet-scale scene matching," in *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–12, IEEE, apr 2012. (Cited on page 153.)

[106] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu, "Landmark Image Super-Resolution by Retrieving Web Images," *IEEE Transactions on Image Processing*, vol. 22, pp. 4865–4878, dec 2013. (Cited on page 153.)

[107] K. Zhang, D. Tao, X. Gao, X. Li, and Z. Xiong, "Learning Multiple Linear Mappings for Efficient Single Image Super-Resolution," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 846–861, 2015. (Cited on page 154.)

[108] M. Douze and H. Jégou, "The Yael Library," in *Proceedings of the ACM International Conference on Multimedia*, (New York, New York, USA), pp. 687–690, ACM Press, 2014. (Cited on page 157.)

[109] C. Conti, P. Nunes, and L. D. Soares, "Inter-layer prediction scheme for scalable 3-D holoscopic video coding," *IEEE Signal Processing Letters*, vol. 20, no. 8, pp. 819–822, 2013. (Cited on page 158.)

[110] R. S. Higa, R. F. Larico-Chavez, R. B. Leite, R. Arthur, and Y. Iano, "Plenoptic image compression comparison," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 3, no. 6, pp. 1–6, 2013. (Cited on page 158.)

[111] Y. Li, M. Sjostrom, R. Olsson, and U. Jennehag, "Efficient intra prediction scheme for light field image compression," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, no. 1, pp. 539–543, IEEE, may 2014. (Cited on page 158.)

Abstract

Efficient video compression is nowadays a critical issue, and is expected to be more and more crucial in the future, with the ever increasing video traffic and the production of new digital video formats with high resolution, wide color gamut, high dynamic range, or high frame rate. The MPEG standard HEVC is currently one of the most efficient video compression scheme, however, addressing the future needs calls for novel and disruptive methods.

In fact, the main principles of modern video compression standards rely on concepts designed more than 30 years ago: the reduction of spatial and temporal redundancies, through prediction tools, the use of a transform to further reduce the inner correlations of the signal, followed by quantization to remove non-perceptive information, and entropy coding to remove the remaining statistical redundancies. In this thesis, we explore novel methods which aims at further exploiting the natural redundancies occurring in video signals, notably through the use of multi-patches techniques. First, we introduce LLE-based multi-patches methods in order to improve Inter prediction, which are then combined for both Intra and Inter predictions, and are proven efficient over H.264. We then propose epitome-based de-noising methods to improve the performances of existing codecs in a out-of-the-loop scheme. High quality epitomes are transmitted to the decoder in addition to the coded sequence, and we can then use at the decoder side multi-patches de-noising methods relying on the high quality patches from the epitomes, in order to improve the quality of the decoded sequence. This scheme is shown efficient compared to SHVC. Finally, we proposed another out-of-the-loop scheme relying on a symmetric clustering of the patches performed at both encoder and decoder sides. At the encoder side, linear mappings are learned for each cluster between the coded/decoded patches and the corresponding source patches. The linear mappings are then sent to the decoder and applied to the decoded patches in order to improve the quality of the decoded sequence. The proposed scheme improves the performances of HEVC, and is shown promising for scalable schemes such as SHVC.

Résumé

L'efficacité des services de compression vidéo est de nos jours un enjeu essentiel, et est appelé à le devenir d'autant plus dans le futur, comme l'indique la croissance constante du trafic vidéo et la production de nouveaux formats tels que la vidéo à haute résolution, à gamme de couleur ou dynamique étendues, ou encore à fréquence d'images augmentée. Le standard MPEG HEVC est aujourd'hui un des schémas de compression les plus efficaces, toutefois, il devient nécessaire de proposer de nouvelles méthodes originales pour faire face aux nouveaux besoins de compression.

En effet, les principes de bases des codecs modernes ont été conçus il y a plus de 30 ans : la réduction des redondances spatiales et temporelles du signal en utilisant des outils de prédiction, l'utilisation d'une transformée afin de diminuer d'avantage les corrélations du signal, une quantification afin de réduire l'information non perceptible, et enfin un codage entropique pour prendre en compte les redondances statistiques du signal. Dans cette thèse, nous explorons de nouvelles méthodes ayant pour but d'exploiter d'avantage les redondances du signal vidéo, notamment à travers des techniques multi-patchs. Dans un premier temps, nous présentons des méthodes multi-patchs basées LLE pour améliorer la prédiction Inter, qui sont ensuite combinées pour les prédiction Intra et Inter. Nous montrons leur efficacité comparée à H.264. La seconde contribution de cette thèse est un schéma d'amélioration en dehors de la boucle de codage, basé sur des méthodes de débruitage avec épitome. Des épitomes de bonne qualité sont transmis au décodeur en plus de la vidéo encodée, et nous pouvons alors utiliser coté décodeur des méthodes de débruitage multi-patchs qui s'appuient sur les patchs de bonne qualité contenus dans les épitomes, afin d'améliorer la qualité de la vidéo décodée. Nous montrons que le schéma est efficace en comparaison de SHVC. Enfin, nous proposons un autre schéma d'amélioration en dehors de la boucle de codage, qui s'appuie sur un partitionnement des patchs symétrique à l'encodeur et au décodeur. Coté encodeur, on peut alors apprendre des projections linéaires pour chaque partition entre les patchs codés/décodés et les patchs sources. Les projections linéaires sont alors envoyées au décodeur et appliquées aux patchs décodés afin d'en améliorer la qualité. Le schéma proposé est efficace comparé à HEVC, et prometteur pour des schémas scalables comme SHVC.