Second-order prediction and residue vector quantization for video compression

Bihong Huang

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal et télécommunications

Ecole doctorale MATISSE

présentée par

Bihong Huang

préparée à l'unité de recherche IRISA TEMICS
(TraitemEnt, Modélisation d'Images et CommunicationS)

Second-order prediction and residue vector quantization for video compression

Thèse soutenue à Rennes
le 8 Juillet 2015

devant le jury composé de :

Francois-Xavier Coudoux
Prof. Univ. Valencienne / rapporteur

Frédéric Dufaux
DR. CNRS / rapporteur

Luce Morin
Prof. INSA Rennes / examinateur

Christophe De Vleeschouwer
Chercheur UCL / examinateur

Patrice Brault
Chercheur CNRS / examinateur

Félix Henry
Ingénieur de Recherche Orange / examinateur

Philippe Salembier
Prof. UPC / examinateur

Christine Guillemot
DR. INRIA / directrice de thèse

Résumé

1. Contexte de la thèse

La compression vidéo est une étape cruciale pour une grande partie des applications de télécommunication: diffusion TV, vidéo sur internet fixe et mobile, stockage DVD/Blu-Ray, applications de vidéo-surveillance, vidéo-conférence... Depuis l'avènement de la norme MPEG-2, un nouveau standard de compression vidéo est produit tous les 10 ans environ, avec un gain en compression de 50% par rapport à la précédente. Ainsi, la dernière norme de codage vidéo, HEVC (pour High Efficiency Video Coding), validée par l'ISO et l'ITU en 2013, permet de gagner 50% par rapport à H.264/MPEG-4 AVC, le standard aujourd'hui massivement déployé dans l'infrastructure de télécommunication mondiale.

Le standard HEVC s'appuie sur une partition de chaque image en blocs. Les Coding Units (CU) sont divisés en arbre quaternaires (quadtrees), un CU étant codé en prédiction inter ou intra. Chaque CU est alors divisé en Prediction Units (PU), chaque PU ayant des paramètres de prédiction spécifiques (par exemple, le mode intra). Enfin, chaque CU est à nouveau divisé en un quadtree de Transform Units (TU), support de la transformée, de la quantification scalaire et du codage des coefficients transformés. Nous nous intéressons plus particulièrement au mode de prédiction intra de la norme HEVC. Un PU peut être prédit en intra de 35 façons différentes: 33 modes de prédiction angulaires interpolent les pixels décodés adjacents au PU considéré (les pixels de référence) selon 33 angles uniformément répartis entre 45° et 225°. Deux modes de prédiction spéciaux sont également possibles: un mode DC, où le prédicteur

est fixé à la moyenne des pixels environnants, et un mode planar, où le pixel en bas à droite du PU est estimé, puis le reste du bloc interpolé avec ce pixel. Le résidu de la prédiction est codé par des moyens classiques: transformée (de type Cosinus Discrète), quantification scalaire, et codage des coefficients transformés.

2. Codage de résidu de prédiction intra par prédiction spatiale

Dans ce chapitre, nous proposons une approche consistant à prédire un résidu de prédiction intra courant par un résidu précédemment décodé dans la même image, une approche que nous nommons Intra Residual Prediction (IRP). Dans cette approche, un résidu courant (associé à un TU), est prédit par un bloc issu des résidus décodés précédents dans la même image. Nous procédons par étapes afin de dégager une méthode de codage du prédicteur du résidu courant.

2.1 Localisation des résidus

Dans un premier temps, nous choisissons comme prédicteur un résidu parmi l'ensemble des blocs décodés, par recherche au niveau pixel, par minimisation de la distorsion. En étudiant la localisation spatiale du meilleur prédicteur, nous observons que:

- Certains résidus décodés sont utilisés beaucoup plus souvent que d'autres comme prédicteurs, ceci pouvant s'expliquer par leur caractéristiques propres (énergie, par exemple).

- Par rapport au résidu courant à prédire, le prédicteur a une probabilité plus élevée de se trouver à proximité du résidu courant, sur l'axe horizontal, tendance que nous interprétons par la prédominance de structures verticales et horizontales dans les images naturelles.

- Le prédicteur a tendance à être aligné avec la grille de TUs de taille 4×4 pixels,

ceci pouvant s'expliquer par les discontinués existant aux frontières des TUs, qui pénalisent les prédicteurs se trouvant à cheval sur plusieurs TUs.

Nous en déduisons une première limitation: le prédicteur du résidu courant devra appartenir à une liste de 40 blocs alignés avec la grille de TU 4×4, 20 situé sur une ligne à gauche du bloc courant, et 20 sur une ligne au-dessus, cette zone coïncidant avec la densité maximale d'accumulation du prédicteur optimal au sens de la distorsion.

2.2 Borne supérieure de gain

Afin de valider l'approche à ce stade, nous évaluons une borne supérieure de gain atteignable, en transmettant un élément de syntaxe indiquant si la prédiction de résidu est mise en œuvre ou pas par l'encodeur sur le TU courant, mais sans transmettre l'indice du prédicteur parmi la liste. La borne supérieure expérimentale est un gain de 11% en compression.

2.3 Limitation des candidats prédicteurs

Parmi la liste de 40 candidats, nous éliminons ceux dont l'énergie est faible ainsi que ceux qui sont proches d'un autre candidat de la liste. Enfin, nous classons les candidats restants par un critère de type Template Matching (proximité du template du prédicteur avec celui du résidu courant): 2^N candidat sont conservés, afin d'être codés sur N bits.

2.4 Performances

L'approche proposée est testée sur un ensemble de séquences vidéos extrait de l'ensemble de test utilisé lors de la construction du standard HEVC. Le gain en compression obtenu est faible, de l'ordre de 0.01 à 0.12% suivant les séquences, malgré un taux de sélection élevé de la méthode. Nous en déduisons que l'approche consistant à réduire les corrélations demeurant dans le résidu de prédiction courant en le prédisant par des résidus décodés ne permet pas d'atteindre des gains satisfaisants.

3. Codage de résidu de prédiction intra par quantification vectorielle mode-dépendante

Sur la base des conclusions du chapitre précédent, nous mettons en œuvre une approche de quantification vectorielle mode-dépendante (MDVQ) du résidu ayant les caractéristiques suivantes: dépendance de mode, combinaison avec le moyen de dé-corrélation classique, à base de dictionnaires optimisés débit-distorsion, indépendant du QP. Nous détaillons ici ces différentes caractéristiques.

3.1 QV mode-dépendante

Les résidus issus des 35 modes de prédictions intra ont des statistiques différentes: en général le résidu a une orientation proche de celle du mode de prédiction. Afin d'exploiter cette caractéristique, nous construisons un dictionnaire de quantification spécifique pour chaque mode de prédiction.

3.2 Combinaison avec la dé-corrélation existante

L'approche proposée ne consiste pas à remplacer la méthode actuelle (transformée, quantification scalaire, codage des coefficients) mais à s'insérer une étape préalable à celle-ci. De plus, la quantification vectorielle du résidu est activable au niveau TU, l'activation étant indiquée par un élément de syntaxe transmis au décodeur. Ainsi, un TU donné peut être codé de deux façons:

- par dé-corrélation classique.

- par l'approche proposée: quantification vectorielle du résidu de premier ordre, puis dé-corrélation classique appliquée au résidu de second ordre.

Il est nécessaire de pouvoir activer ou désactiver le codage proposé, car les dictionnaires de quantification utilisés sont de taille fixe. Il y a donc un débit minimal à dépenser lorsqu'ils sont utilisés, ce qui n'est pas adapté pour le codage de résidus de faible amplitude, par exemple. Par ailleurs, il est également nécessaire de conserver

le moyen de dé-corrélation classique. En effet, la quantification vectorielle ne permet pas de couvrir de façon simple tous les débits, en particulier les plus élevés, pour des raisons de complexité et de stockage.

3.3 Dictionnaires optimisés débit-distorsion

Afin de construire les dictionnaires, nous mettons en œuvre un apprentissage à partir de séquences vidéo différentes de celles utilisées pour les tests de performance. Cet apprentissage est basé débit-distorsion et itératif.

Au cours d'une première itération, on applique les étapes suivantes:

- **étape 1** les séquences d'apprentissage sont codées de façon classique, puis,
- **étape 2** les résidus de prédiction intra sont extraits et classés par mode intra, constituant ainsi des résidus d'apprentissage, puis,
- **étape 3** un dictionnaire de quantification est construit pour chaque mode intra à partir des résidus d'apprentissage associés. Le dictionnaire est construit par un algorithme classique (par exemple, k-means ou algorithme de Kohonen).

Chaque itération suivante utilise les résidus d'apprentissage issus de l'itération précédente. Au cours de ces itérations on applique les étapes suivantes:

- **étape 1** les séquences d'apprentissage sont codées avec l'approche MDVQ, ce qui implique que certains TUs sont codés de façon classique, et d'autres avec la quantification vectorielle utilisant les dictionnaires issus de l'itération précédente, le choix étant fait par optimisation lagrangienne classique. Seuls les résidus de prédiction intra codés par quantification vectorielle sont extraits pour constituer des résidus d'apprentissage pour l'itération suivante.
- **étape 2** un dictionnaire de quantification est construit pour chaque mode intra à partir des résidus d'apprentissage associés.

A l'issue de quelques itérations, les dictionnaires sont adaptés aux résidus des TUs qui sont sélectionnés par l'optimisation débit-distorsion pour être codés selon le mode quantification vectorielle. On a alors des dictionnaires bien adaptés à la source que l'on souhaite coder.

3.4 Indépendance de QP

Dans HEVC, le paramètre de quantification (QP) fixe le compromis débit-distorsion. Ainsi, un QP élevé implique une compression à un débit faible, avec une qualité de restitution mauvaise, et inversement. Or la prédiction intra se base sur des pixels reconstruits pour effectuer une prédiction du bloc courant. En conséquence, lorsque le QP est élevé, les pixels reconstruits sont en général de mauvaise qualité, et induisent une prédiction elle-même peu fiable. Le résidu de prédiction possède alors une dynamique plus élevée. On confirme expérimentalement que la dynamique du résidu de prédiction intra est croissante avec le QP. Ceci pourrait nous amener à utiliser des dictionnaires différents pour chaque QP. Or, nous montrons expérimentalement que l'on obtient le même gain en compression en utilisant un seul dictionnaire de quantification pour tous les QP que lorsque les dictionnaires sont construits indépendamment pour chaque QP.

Ceci nous amène à valider de façon théorique le résultat expérimental. Nous montrons que les dictionnaires de quantification vectorielle sont indépendants du QP, sous les hypothèses suivantes:

- la quantification vectorielle doit être activable au niveau TU, et l'activation doit se faire par critère lagrangien.

- les coefficients issus de la transformation et quantification scalaire du résidu secondaire (après prédiction et quantification vectorielle) doivent suivre les hypothèses classiques d'indépendance statistique et de pas de quantification faible devant la dynamique du signal.

3.5 Signalisation MDVQ

Nous disposons maintenant de dictionnaire de quantification apte à dé-corréler les résidus de prédiction intra. Nous fournissons une syntaxe permettant de transmettre au décodeur les informations associées. Concrètement, pour chaque CU, un élément de syntaxe $res\_vq\_cu\_flag$ est transmis, qui indique si au moins l'un de ses TUs utilise la quantification vectorielle du résidu. Cet élément de syntaxe est codé à l'aide d'un contexte entropique unique. Lorsque cet élément de syntaxe vaut 1, un élément de syntaxe $res\_vq\_tu\_flag$ est transmis pour chaque TU contenu dans le CU, lui aussi codé avec un unique contexte entropique. Lorsque cet élément de syntaxe vaut 1, ceci signifie que la quantification vectorielle de résidu est activée lors du codage. Un élément de syntaxe res_vq_idx est alors transmis, qui indique l'index du vecteur de quantification dans le dictionnaire associé au mode de prédiction intra du TU courant, et à sa taille de bloc. Cet élément de syntaxe est codé de façon équiprobable. Ainsi, si le dictionnaire a une taille 2^M, M bits sont transmis.

3.6 Résultats expérimentaux

Nous utilisons les conditions de test classiques recommandées lors de la conception du standard HEVC, et nous nous restreignons à la configuration dans laquelle toutes les images sont codées en intra (All Intra). Les dictionnaires de quantification utilisés sont de taille 256. Nous limitons l'usage des dictionnaires aux blocs de taille 4×4 et 8×8 pixels, afin de conserver un impact réaliste sur la complexité et le stockage. Nous utilisons donc 35 dictionnaires de 256 vecteurs, pour les tailles de TU 4×4 et 8×8.

Par ailleurs, pour construire les dictionnaires, nous comparons deux méthodes: l'approche classique dite k-means d'une part, et l'algorithme de Kohonen d'autre part. L'algorithme k-means, bien connu, consiste, au cours de chaque itération, à remplacer un vecteur du dictionnaire par le centroïde des vecteurs d'apprentissage qu'il a codés. L'algorithme de Kohonen consiste à mettre à jour, pour chaque vecteur d'apprentissage, non seulement le plus proche voisin (au sens de la distorsion) dans le dictionnaire, mais aussi les voisins de celui-ci (au sens de l'index dans le dictionnaire). Plus la distance

dans le dictionnaire est grande, plus la mise à jour est de faible amplitude. On voit que la mise à jour de Kohonen consiste à rapprocher le vecteur de quantification du vecteur courant (qui joue un rôle d'apprentissage), mais aussi, dans une moindre mesure, de ses voisins (au sens de l'index), produisant ainsi un dictionnaire qui possède la caractéristique d'être ordonné: deux vecteurs voisins dans le dictionnaire sont proches au sens de la distorsion.

Dans une première expérience, nous utilisons des séquences d'apprentissage différentes des séquences à coder. Lorsque l'algorithme k-means est utilisé pour construire les dictionnaires, un gain moyen de 1.1% en compression est observé sur l'ensemble des séquences de test. Lorsque l'algorithme de Kohonen est utilisé, nous observons un gain moyen de 0.8%.

Dans une deuxième expérience, nous utilisons comme séquences d'apprentissages les séquences de test. Ceci est une approche non réaliste dont le but est uniquement d'évaluer la particularité des dictionnaires construits. Dans cette approche non réaliste, le gain moyen observé est de l'ordre de 5%.

Enfin, nous observons que l'augmentation du temps de décodage par rapport à un décodage HEVC classique est faible, de l'ordre de 3%. Le temps de codage de notre approche ne nous semble pas pertinent dans la mesure où un codeur par quantification vectorielle n'utiliserait pas une approche de recherche exhaustive dans une application réelle (encodeur professionnel), de nombreuses techniques d'accélération étant disponibles dans la littérature.

3.7 Conclusion

L'approche MDVQ permet un gain substantiel en compression, de l'ordre de 1.1%, dans une approche réaliste: dictionnaires de petite taille limités aux petits TUs, impact très faible sur le temps de décodage. La construction de dictionnaire par k-means donne de meilleurs résultats par rapport à l'approche Kohonen. Par ailleurs, lorsque nous forçons l'adaptation des dictionnaires à la séquence à coder, celle-ci servant de séquence d'apprentissage, on obtient des gains beaucoup plus élevés. Ce scénario non

réaliste semble indiquer que des gains supplémentaires seraient possibles en adaptant les dictionnaires en cours de codage à la séquence traitée. C'est l'objet du chapitre suivant.

4. Codage de résidu de prédiction intra par quantification vectorielle mode-dépendante adaptative

Dans ce chapitre, nous proposons une amélioration de l'approche MDVQ visant à adapter les dictionnaires de quantification au fur et à mesure du codage des TUs.

4.1 Principe de la mise à jour de Kohonen pour l'adaptabilité

Pour ce faire, nous utilisons une étape dite de mise à jour de Kohonen. Elle consiste, lors du codage ou du décodage d'un TU, à mettre à jour un ensemble de vecteurs du dictionnaire. Ainsi, lorsque la quantification vectorielle est activée, par choix débit-distorsion, sur un TU nommé T, celui-ci est décodé de la façon suivante:

$$T = P + V + R \tag{1}$$

où P est le prédicteur intra, V est le vecteur de quantification, et R est le résidu secondaire décodé, issu d'une déquantification scalaire et d'une transformée inverse des coefficients transformés décodés. Nous supposons que le dictionnaire de quantification CB(t) associé au mode intra courant, à l'instant t, comporte les M vecteurs $\{V_0(t), V_1(t), ..., V_{M-1}(t)\}$. La mise à jour de Kohonen définit les vecteurs à l'instant $t+1$ comme:

$$V_i(t+1) = V_i(t) + \alpha \cdot \theta(i) \cdot (V_i(t) - (V + R)) \tag{2}$$

où α est un paramètre fixe, représentatif de l'ampleur de la mise à jour, et θ est un paramètre décroissant de la distance (au sens de l'index) entre $V_i(t)$ et le vecteur V dans le dictionnaire. En un sens, il s'agit de prolonger la méthode de Kohonen, normalement

appliquée comme apprentissage lors de la construction du dictionnaire, et en faire un processus d'adaptation aux vecteurs codés lors du traitement d'une séquence vidéo donnée.

4.2 Dictionnaire initial

Afin de mettre en œuvre la mise à jour de Kohonen en tant que apprentissage, nous voyons qu'il est préférable de partir d'un dictionnaire de quantification ordonné. En effet, la mise à jour de Kohonen a de toute façon tendance à ordonner le dictionnaire, donc partir d'un dictionnaire non ordonné risque d'engendrer une phase de perturbation intermédiaire avant d'arriver à une situation stable. Il est possible de construire le dictionnaire initial avec l'algorithme de Kohonen lui-même, puisque celui-ci produit un dictionnaire ordonné. Toutefois, nous avons vu précédemment que l'algorithme k-means produit expérimentalement un dictionnaire de meilleure qualité que l'algorithme de Kohonen. Nous décidons donc, plutôt que d'utiliser le dictionnaire de Kohonen comme dictionnaire initial, d'utiliser un dictionnaire k-means réordonné. Pour ce faire, nous mettons en œuvre une variante de la résolution du problème du voyageur de commerce (TSP), nous permettant de déterminer un chemin court (au sens de la distorsion) entre les vecteurs de quantification. Bien sûr cette étape est faite une fois pour toutes, et ne vient pas impacter le codage ou le décodage. En somme il ne s'agit que d'indexer les vecteurs du dictionnaire k-means d'une façon particulière.

4.3 Optimisation des paramètres de mise à jour

Nous optimisons les paramètres α et θ de la mise à jour de Kohonen, en mettant en lumière de façon extensive leur lien avec les caractéristiques intrinsèques de la séquence. En particulier, nous montrons que plus la séquence est éloignée des dictionnaires de quantification initiaux, plus la mise à jour doit être importante. Par ailleurs, nous distinguons le cas où une séquence est codée en mode All Intra, où une ré-initialisation des dictionnaires est nécessaire pour chaque image de la vidéo (puisque elles doivent pouvoir être décodées indépendamment), et une configuration de codage de type Random Ac-

cess, où une ré-initialisation des dictionnaires n'est nécessaire que lors de points d'accès aléatoires (image intra) de la vidéo: dans le deuxième cas, le processus d'apprentissage est beaucoup plus long. Pour des raisons de simplicité algorithmique du codeur et du décodeur, nous choisissons de garder ces paramètres fixes. Nous dérivons donc des valeurs optimales sur l'ensemble des séquences vidéo de test.

4.4 Performances

Dans un premier temps, nous appliquons la méthode MDVQ adaptative dans les mêmes conditions de test que la méthode MDVQ, c'est-à-dire en configuration All Intra. Le gain moyen est de 1.5% (alors qu'il est de 1.1% en MDVQ non adaptative). On note que sur la plupart des séquences de type image naturelle, le gain apporté par l'adaptativité est modeste, tandis que pour des contenus plus atypiques, comme la classe F qui comporte des images non naturelles, le gain est beaucoup plus important.

Dans un deuxième temps, nous appliquons la méthode MDVQ adaptative en configuration Random Access. On note que bien qu'il s'agisse d'une configuration qui met en œuvre la prédiction inter, la quantification vectorielle du résidu n'est appliquée que sur les blocs intra. On note également que dans cette configuration, les dictionnaires ne sont ré-initialisés qu'au début de chaque image Intra, ce qui laisse plus de temps pour l'apprentissage. Dans cette configuration, le gain moyen est de 1.4%.

D'un point de vue complexité de décodage, une augmentation de 25% est constatée par rapport au décodeur HEVC classique, ce qui est supérieur aux 3% observés en MDVQ non adaptative, ceci s'expliquant bien sûr par les calculs supplémentaires que provoque la mise à jour du dictionnaire. Toutefois, on reste dans une augmentation de complexité raisonnable par rapport au gain en compression obtenu.

5. Conclusion

Dans cette thèse, nous proposons trois approches pour améliorer la compression vidéo en exploitant les corrélations du résidu de prédiction intra. Une première approche

basée sur l'utilisation de résidus précédemment décodés montre que, si des gains sont théoriquement possibles, le surcoût de la signalisation les réduit pratiquement à néant. Une deuxième approche basée sur la quantification vectorielle mode-dépendent (MDVQ) du résidu préalablement à l'étape classique transformée-quantification scalaire, permet d'obtenir des gains substantiels. Nous montrons que cette approche est réaliste, car les dictionnaires sont indépendants du QP et de petite taille. Enfin, une troisième approche propose de rendre adaptatifs les dictionnaires utilisés en MDVQ. Un gain substantiel est apporté par l'adaptivité, surtout lorsque le contenu vidéo est atypique, tandis que la complexité de décodage reste bien contenue. Au final on obtient un compromis gain-complexité compatible avec une soumission en normalisation.

Nous proposons deux pistes d'amélioration principales. D'une part, afin de réduire les désavantages inhérents à la quantification vectorielle classique qui est complexe côté encodeur, et qui nécessite un stockage de dictionnaires, il serait intéressant de mettre en œuvre une quantification vectorielle sur réseau (Lattice Vector Quantization). Celle-ci, de par sa structure régulière, ne nécessite pas de stockage et est plus rapide. Le défi sera alors le suivi des probabilités des éléments du réseau, pour les différents modes de codage intra. D'autre part, il serait nécessaire d'étendre le concept de quantification vectorielle du résidu au mode inter, de façon à profiter pleinement du gain en performance en mode sur tous les résidus. Il faudra dans ce cas mettre en œuvre une classification des résidus permettant d'identifier les différents types, de façon similaire aux modes intra, afin de mettre en œuvre une quantification spécifique pour chaque type de résidu.

Abstract

Video compression has become a mandatory step in a wide range of digital video applications. Since the development of the block-based hybrid coding approach in the H.261 standard, new coding standard was ratified every ten years and each new standard achieved approximately 50% bit rate reduction compared to its predecessor without sacrificing the picture quality. However, due to the ever-increasing bit rate required for the transmission of HD and Beyond-HD formats within a limited bandwidth, there is always a requirement to develop new video compression technologies which provide higher coding efficiency than the current HEVC video coding standard.

The work done in this thesis aims at improving the intra coding efficiency of the HEVC standard, because even though the intra prediction of the HEVC standard is efficient at reducing the spatial correlation in pixel signals, it has limited performances to exploit complex textures and structures. The methods developed in this thesis intent to compress the residual information after the intra prediction is performed. We first proposed a method of Intra Residual Prediction (IRP) that exploits the remaining redundancy in the residual domain. In this method, the current intra prediction residue is predicted with a reconstructed residual block. Assuming that the cost of signaling the prediction is zero, an upper-bound of 11% of bit rate saving can be achieved with the IRP residual coding approach. Then, a method of Mode Dependent Vector Quantization (MDVQ) is proposed for coding the current intra prediction residue with mode dependent codebooks. With a generic QP-independent codebook, this method achieves a real bit rate reduction of 1.1% on average compared with HEVC standard. Finally, we developed the method of Adaptive Mode Dependent Vector Quantization

(AMDVQ) to code the intra prediction residue adaptively. The adaptivity of vector quantizer further improves the coding efficiency over the second method, the MDVQ residual coding, from 1.1% to 1.5% on average. We conclude that the intra coding efficiency of the HEVC standard can be further improved by exploiting the remaining correlation in the residual domain.

Contents

1 A state-of-the-art in video coding **5**

1.1 High Efficiency Video Coding Standard 5

1.1.1 Block partitioning structure 6

1.1.2 Intra picture prediction 11

1.1.3 Inter picture prediction 12

1.1.4 Transform coefficient coding 13

1.1.5 Entropy coding 14

1.1.6 Encoder control 15

1.2 Other tools of image compression 16

1.3 Conclusions 18

2 Intra residual coding using spatial prediction **19**

2.1 Block based prediction in the residual domain 20

2.2 Residual predictor search region 22

2.3 Upper bound of the RD gain with a limited candidate set 26

2.3.1 On/Off indicator 26

2.3.2 Estimation of an upper bound of the RD gain 27

2.4 Candidate list construction tools 29

2.4.1 Elimination of useless candidates 30

2.4.2 Ordering candidates in terms of similarity 30

2.4.3 Constructing candidates list 34

2.5 Experimental results 36

2.5.1 Syntax element design 36
2.5.2 Simulations 37
2.6 Conclusion 38

3 Intra residual coding using mode dependent vector quantization 41
3.1 Vector coding 42
3.1.1 Vector quantization 42
3.1.2 Vector quantizer design 45
3.2 Intra residual coding using mode dependent vector quantization 50
3.2.1 Mode dependent vector quantization 52
3.2.2 Switched vector quantization combined with optional scalar quantization 55
3.2.3 Rate distortion optimization of codebook construction 56
3.2.4 QP independent codebook construction 57
3.3 MDVQ-based residual coding mode signaling 62
3.4 Experimental results 64
3.4.1 Simulation A: Special codebook constructed by k-means algorithm 66
3.4.2 Simulation B: Generic codebook constructed by k-means algorithm 67
3.4.3 Simulation C: Generic codebook constructed by Kohonen algorithm 68
3.5 Conclusion 69

4 Intra residual coding using Adaptive MDVQ 73
4.1 Adaptive MDVQ residual coding in HEVC 74
4.2 Adaptive codebook construction 76
4.2.1 Adaptive codebook structure 76
4.2.2 Adaptive codebook construction 77
4.3 Codebook update using the Kohonen algorithm 80
4.3.1 Neighboring code vectors 82
4.3.2 Multiplying factors 82
4.3.3 Code vector update 84

4.3.4 Implementation issues 84
4.4 Experimental results 85
4.4.1 Adaptive codebook parameter optimization 85
4.4.2 Performance of Adaptive MDVQ 89
4.4.3 Complexity aspect 90
4.5 Conclusion 94

5 Conclusion and Future works **97**

A **101**

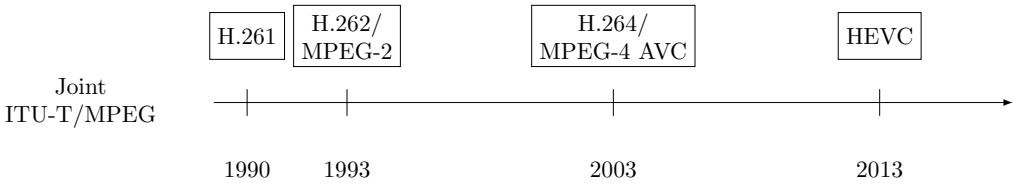
Bibliography **103**

Introduction

Context

Video compression has become a mandatory step in a wide range of digital video applications including broadcast digital TV, video transmission over Internet and mobile network, real-time conversational applications such as video chat, video conferencing, DVD and Blu-Ray discs, video content acquisition and editing systems, and camcorders of security applications [AD12].

Since the development of the block-based hybrid video coding approach in the H.261 standard in 1990, researches in this field has increased significantly in recent years. New compression techniques were developed and included gradually in video coding standards, so that new standard was ratified every ten years. Each new standard achieves approximately 50% bit rate reduction compared to its predecessor without sacrificing the picture quality. The following figure shows the evolution of video coding standards, including MPEG-1 Video [IT93], MPEG-2 Video [IT94], H.264/MPEG-4 Advanced Video Coding (AVC) [oIT03], and High Efficient Video Coding (HEVC).



Evolution of video coding standard.

These video coding standards played an important role in enabling multimedia applications. However, due to the ever-increasing bit rate required for the transmission of HD and Beyond-HD formates (e.g. 4k $\times$ 2k or 8k $\times$ 4k resolution) within a limited bandwidth, there is always a requirement to develop new video compression technologies which provide higher coding efficiency than the current HEVC video coding standard.

The work done in this thesis falls in the HEVC standardization context. The goal is to develop new coding tools on top of the HEVC reference software to further improve the intra coding efficiency and to be potentially integrated into the post-HEVC video coding standard. The work in this thesis was conducted in the Advance Video Coding (CVA) team of Orange Labs, and in the Traitement, Image Modelization and Communications (TEMICS) project of IRISA Laboratory.

Contributions

Several methods aiming at improving the intra coding efficiency of the HEVC standard were developed during this thesis. These methods have in common that they intent to compress the residual information after the intra prediction is performed. The main contributions are

- A method of Intra Residual Prediction (IRP) which exploits the remaining redundancy in the residual domain. This approach uses a predictor, which is selected from a candidate list of reconstructed residual blocks, to predict the current intra prediction residue. Methods of constructing the candidate list have also been explored.

- A method of Mode Dependent Vector Quantization (MDVQ) which codes the intra prediction residue with mode dependent codebooks. These codebooks are constructed by applying a training algorithm (k-means or Kohonen) on residual vectors extracted from training sequences. These codebooks are optimized in a rate distortion sense and are independent of the quantization parameters.

- A method of Adaptive Mode Dependent Vector Quantization (AMDVQ) which allows to code the intra prediction residue adaptively. An approach of Kohonen-based codebook update was developed for modifying the code vectors gradually during the residual coding process.

Thesis structure

This manuscript starts with a state-of-the-art of the coding technologies in the current HEVC standard. Then, it comprises three chapters, each describing one of the previously introduced coding approaches. More precisely, the structure of this manuscript is as follows:

- **Chapter 1** presents a state-of-the-art in 2D video coding. It starts by an overview of the coding tools in the HEVC standard. Then it discusses others coding tools that are widely used in image coding but could be extended into video coding domain.

- **Chapter 2** presents a method of prediction that is applied on intra prediction residue to further reduce the redundancy in the residual domain. In this chapter we study the statistical characteristics of intra prediction residue. We discuss criteria used to restrict the predictor candidate search region, and propose an approach of constructing candidate list using reconstructed residual blocks in the search region. The coding gain of intra residual prediction are reported and interpreted in the section of experimental results.

- **Chapter 3** describes a novel residual coding tool called Mode Dependent Vector Quantization (MDVQ). Codebooks are designed to model more accurately the directional structures of residues in order to exploit the remaining correlations in the residual domain. This chapter also presents how to construct rate distortion optimized codebooks and shows that codebooks can be independent of quantization parameters. Performance is assessed by integrating this method in the

HEVC standard with simulations under HEVC common test configuration.

- **Chapter 4** proposes an approach of Adaptive Mode Dependent Vector Quantization (AMDVQ) to further improve the MDVQ coding tool by modifying gradually the code vectors of the codebooks, to adaptively code the intra prediction residue. We describe in details how to apply the Kohonen learning algorithm for codebook update and how to construct adequate modifiable codebooks. This method is also integrated in the HEVC standard and the significant coding gains of AMDVQ approach are reported and interpreted in the section of experimental results.

- **Chapter 5** ends this manuscript with a summary of the proposed methods and their associated results, as well as some perspectives for future works in this field.

Chapter 1

A state-of-the-art in video coding

The work done in this thesis aims at improving the video coding efficiency of the current state-of-the-art, HEVC standard. Hence, we start with a description of the key features of the coding scheme defined in HEVC standard, and follow with a brief presentation of two coding tools, template matching and vector quantization, which we later apply in the video coding scheme for further improving the video coding performance.

1.1 High Efficiency Video Coding Standard

The classic block-based hybrid video coding approach is the basis of the video coding standards which consists of a hybrid of inter picture prediction for reducing the temporal redundancy, intra picture prediction for reducing the spatial redundancy, and transform coding for reducing the redundancy of the residual information. The current state-of-the-art, HEVC standard follows the same principle, but further extends it to be able to more efficiently compress information by reconsidering several aspects. Indeed, the significant compression efficiency improvement of the HEVC standard over prior standards is the result of assembling a plurality of coding elements each of which provides smaller improvement.

Here, we highlight some key features of HEVC design, including block partitioning structure, intra picture coding, inter picture coding, transform coefficient coding, en-

tropy coding, and encoder control of coding modes. These features are the basis steps of the video coding scheme. Furthermore, these steps are closely related to our contributions. When introducing the proposed residual compression techniques in HEVC, the implementation involves all of these steps.

1.1.1 Block partitioning structure

Of the many new technical aspects of HEVC, the block partitioning structure has been identified as one of the most significant improvements with respect to previous video coding standards. In contrast to the fixed size macro-block structure of H.264/AVC, HEVC adopts a highly flexible and efficient block partitioning structure. Four block concepts having separate functionality are introduced in HEVC [KML+12]: coding tree unit (CTU), coding unit (CU), prediction unit (PU) and transform unit (TU). HEVC also defines the terms coding tree block (CTB), coding block (CB), prediction block (PB) and transform block (TB) for specifying the 2-D sample array of one of the three color components (one Luma and two Chroma) associated with the CTU, CU, PU and TU, respectively. Thus, when the color component sampling format is 4:2:0 in the YCbCr color space, a CTU consists of one luma CTB, two chroma CTBs, and associated syntax elements. This relationship between coding unit and coding block is also valid for CU, PU and TU. We describe in greater detail the functionality of each coding unit and their partitioning structure in the following sections.

A. Coding tree unit

The CTU is the basic processing unit used in the HEVC standard, which is an analogous term to the macro-block in H.264/AVC. Coding a picture (frame) of a video sequence starts with the partitioning of this picture into an integer multiple of CTUs, and processes the CTU coding in a raster scan order.

In the main profile of HEVC [SOHW12], the minimum and the maximum sizes of CTU are specified by the syntax elements in the sequence parameter set (SPS) among sizes of 8×8, 16×16, 32×32 and 64×64. This flexibility of CTU size allows the

coding structure to match the characteristics of not only high definition video content but also small resolution video content distributed over handheld equipments.

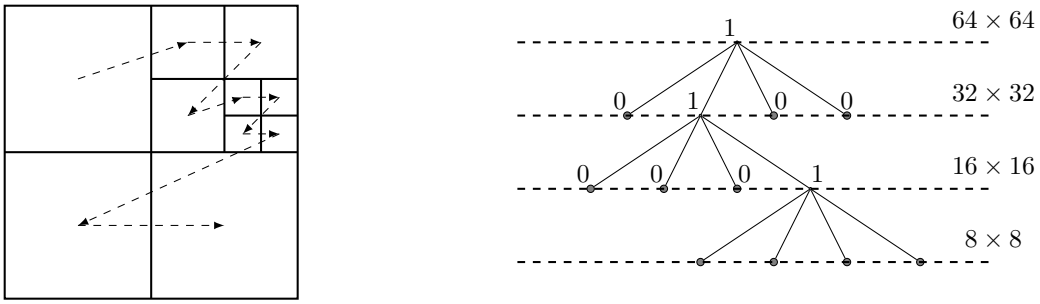
B. Partitioning of the CTU into CUs

HEVC utilizes CU as a unit to determine the basic coding-type among intra picture coding, inter picture coding and skipped coding. According to the coding-type, the CU is referred to as intra coded CU, inter coded CU and skipped CU, respectively. An intra coded CU uses neighboring reconstructed samples in the same frame to predict the current block, while an inter coded CU conducts the prediction using motion compensation scheme. A skipped CU is a special form of inter coded CU which inherits the motion vectors from neighboring CUs and the inter prediction residue is set to zero so that the transform coefficient coding can be skipped.

Partitioning of CTU into multiple CUs of different size has two main advantages. Firstly, it allows coding a CTU with more coding possibilities, and secondly, the encoder can select the best coding solution to adapt to various local characteristics of the region that is covered by the CTU. The partitioning of the CTU is performed recursively using a quadtree structure which is denoted as the coding tree [KML+12]. For example, the CTU of size $2N \times 2N$ can be a single CU or can be split into four smaller units of equal sizes of $N \times N$, which are nodes of the coding tree. If a unit is a leaf node of the coding tree, the unit becomes a CU. Otherwise, the unit is split again into four smaller units when the unit size is equal or larger than the minimum CU size specified in the SPS.

Fig. 1-1 illustrates an example of CTU partitioning and the processing order of CUs when the size of the CTU is equal to 64×64 and the minimum CU size is equal to 8×8. Each squared block in Fig. 1-1(a) represents a CU. In this example, a CTU is split into 10 CUs of different sizes and positions. Fig. 1-1(b) shows the coding tree structure representing the structure of the CTU partitioning in Fig. 1-1(a). Numbers of 1 and 0 on the tree represent whether the CU is further split or not. The CUs are processed in an order of depth-first traversing of the coding tree structure, as shown

by the dotted line in Fig. 1-1(a).



(a) (b)

Figure 1-1: Example of CTU partitioning and the processing order of CUs when the size of the CTU is equal to 64×64 and the minimum CU size is equal to 8×8. (a) CTU partitioning. (b) CTU coding tree structure.

C. Partitioning of the CU into PUs

According to the coding-type, a CU can be split into one, two or four PUs. Each PU works as a basic representative block for specifying all information related to the prediction scheme such as the intra prediction mode for intra coded CU or the index of motion vector for inter coded CU. This information is unique per PU.

Fig. 1-2 illustrates the possible PU partitioning structures when the CU size is equal to $2N \times 2N$. Unlike the CU partitioning structure, the splitting of CU into PUs is allowed to be performed only once. A skipped CU can not be further partitioned. An intra coded CU has two partitioning possibilities, $2N \times 2N$ and $N \times N$. Eight partitioning modes are defined for inter coded CU: two squared modes of $2N \times 2N$ and $N \times N$, two symmetric partitioning modes of $2N \times N$ and $N \times 2N$, and four asymmetric modes of $2N \times nU$, $2N \times nD$, $nL \times 2N$, $nR \times 2N$.

The flexibility of PU partitioning structure improves the prediction accuracy, but increases as well the encoding complexity. To reduce the encoding complexity, HEVC also defines some partitioning restrictions according to the CU size. First, the use of $N \times N$ partition is allowed only when the CU size is equal to the minimum CU

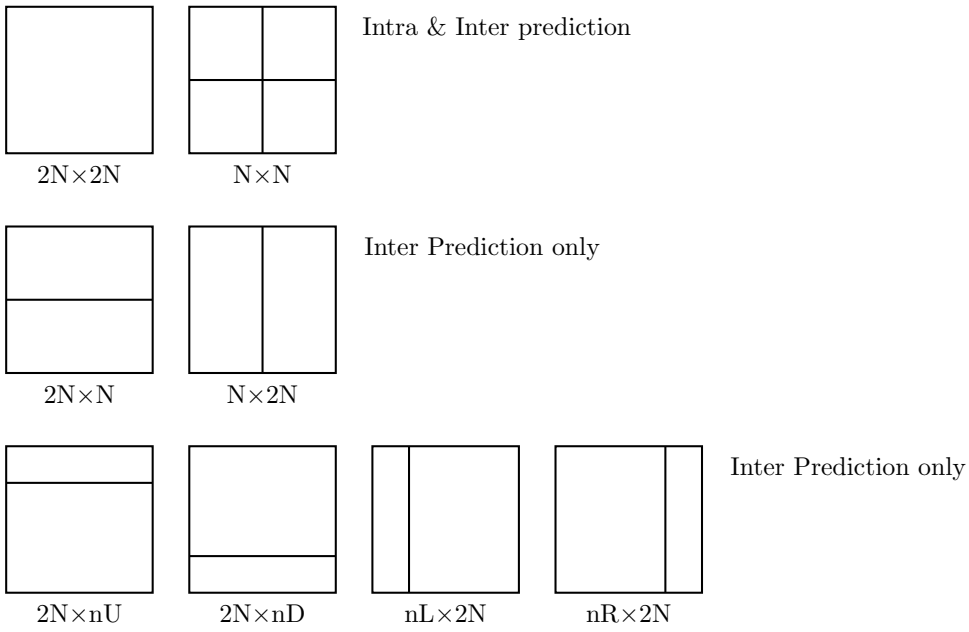


Figure 1-2: Illustration of PU partitioning in HEVC.

size. Because the partitioning of a intra predicted CU into four equal-size PUs is conceptually similar with the case of splitting this CU into four equal-size sub-CUs. When a intra predicted CU is not equal to the minimum CU size, the $N \times N$ PU partitioning mode conducts the same coding result when the CU is further split since the intra prediction is performed at TU level. Second, in case of inter coded CU, the use of the $N \times N$ partitioning mode is disabled when the CU size is equal to 8×8. Moreover, asymmetric shapes for inter coded CU are only allowed when CU size is not equal to the minimum CU size.

D. Partitioning of the CU into TUs

Subsequent to the prediction step, a CU is partitioned into several TUs and each of them is a basic unit of transform coefficient coding. The partitioning of CU into TUs can be performed recursively using a quadtree structure [KML+12]. This tree is called transform tree or residual quadtree (RQT). Fig. 1-3 illustrates an example of transform tree and corresponding TU splitting. Numbers of 0 and 1 on the tree represent whether

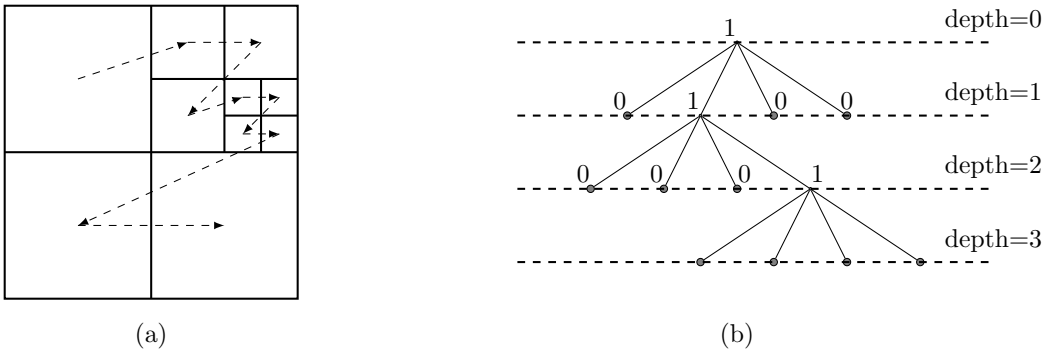


Figure 1-3: Example of partitioning a 32×32 CU into TUs. (a) TU partitioning structure. The dotted lines represent the processing order of TUs during the Transform Coefficient Coding. (b) Transform coding tree structure.

the TU is further split or not. Each leaf node of the transform tree is a TU on which the transform and scalar quantization is applied. The minimum TU size can be predefined by a maximum TU depth parameter. When the TU size is equal to the CU size, the depth of this TU is 0, the TU depth increments at each time the TU is further split.

In HEVC, both the PU and the TU can reach the same size as the corresponding CU. This leads to the fact that the size of TU may be larger than that of the PU or may cover a region of several PUs in the same CU. Consequently, residuals of different PUs inside this TU can be transformed together. However, this case is only allowed for inter coded CU. Because for intra coded CU, the intra prediction mode is established for each PU and is shared for all TUs inside this PU. The intra prediction is actually conducted separately for each TU in a order represented by the doted line shown in Fig. 1-3. The processing order of each TUs inside a CU is actually an order of depth-first traversing of the transform tree structure. One obvious advantage for this processing order of intra prediction is the possibility of always utilizing the nearest neighboring reference samples from the already reconstructed TU.

1.1.2 Intra picture prediction

Intra picture prediction reduces the spatial redundancy by exploiting the spatial correlation of a picture. It based on an extrapolation of previously decoded samples in the neighboring blocks, followed by transform coefficient coding, scalar quantization and entropy coding for further compressing the prediction residuals. For a block of size $N \times N$, the reference samples of intra prediction refer to a total of $4N + 1$ spatially neighboring decoded pixels, as shown in Fig. 1-4. In addition to the reference samples from TUs on the left, above, and above right of the current block used for intra prediction in earlier coding standards, HEVC uses also the samples from lower left blocks for intra prediction when they are available from preceding decoding operations.

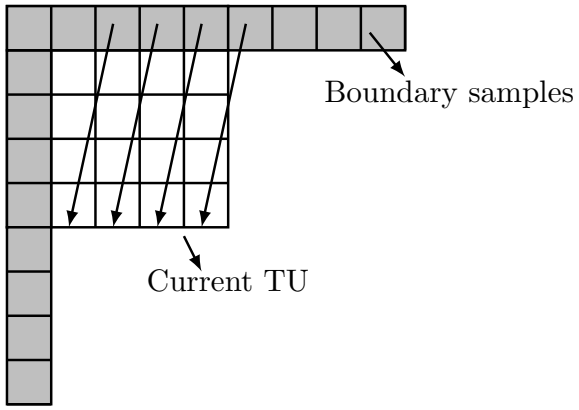


Figure 1-4: Example of intra prediction angular mode.

Compared with previous standards, HEVC introduces more intra prediction modes to model accurately different directional structures as well as smooth regions. In addition to planar and DC prediction, HEVC supports in total 33 angular prediction modes for all the PU sizes. The intra DC prediction predicts the current block with the average value of all reference samples. The intra planar prediction is designed to prevent discontinuities along the block boundaries. It consists of performing firstly two linear predictions of four corner reference samples, and then using their average value for the prediction. The intra angular mode involves an extrapolating of the reference

samples projected according to the intra mode direction. As shown in Fig. 1-5, intra prediction mode 0 refers to the planar intra prediction, mode 1 refers to DC prediction, and mode 2 to 34 refers to angular predictions modes with different directionality.

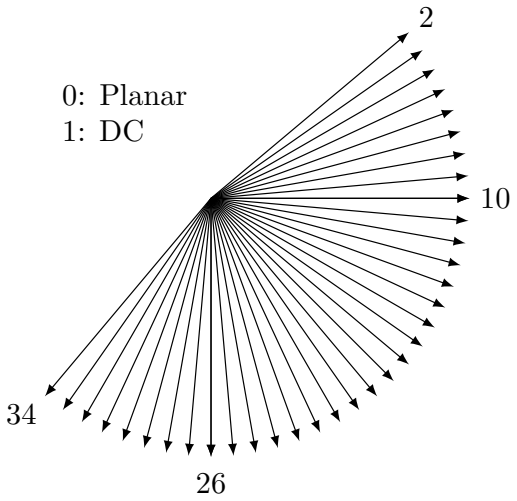


Figure 1-5: Intra prediction mode and directional orientation.

1.1.3 Inter picture prediction

The inter picture prediction reduces the temporal redundancy between neighboring pictures in video sequences. The samples of inter prediction are obtained from the reference block of already decoded reference pictures. The reference block is at a position displaced by a motion vector in the reference picture which is identified by an index in the reference picture list. In the case of Bi-direction inter prediction, an identification of which reference picture list (forward or backward) is associated with each index.

HEVC includes a merge mode to derive the motion information from spatially or temporally neighboring blocks. Motion information typically consists of the horizontal and vertical motion vector displacement values, one or two reference picture indexes. The merge process consists in deriving a list of candidate motion vectors. If the encoder

choses one of these motion vectors, a merge flag is set to be "on" and is transmitted together with the index of the candidate motion vector in the list. Otherwise, the flag is set to be "off" and the motion vector is transmitted explicitly.

In HEVC, the skip mode is treated as a special case of the merge mode when all coded block flags are equal to zero. In this specific case, only a skip flag and the corresponding merge index are transmitted to the decoder.

When an inter picture predicted CU is not coded in the skip nor merge mode, the motion vector is coded differently using a motion vector predictor. Similar to the merge mode, HEVC allows the encoder to choose the motion vector predictor among multiple predictor candidates. The difference between the predictor and the actual motion vector are transmitted to the decoder.

1.1.4 Transform coefficient coding

As in prior standards, HEVC uses transform coefficient coding to further compress the prediction residuals. Despite that, HEVC introduces several new features in transform coding to help improve upon H.264/AVC, such as larger TU sizes, mode dependent coefficient scanning, last significant coefficient coding, multilevel significance maps, improved significance flag context modeling, and sign data hiding [SJN+12].

The mode dependent coefficient scanning [ZCW+11] is a new tool in HEVC that exploits the horizontal or vertical correlation of the residual related to the intra prediction mode. One of the three coefficient scanning orders, diagonal, horizontal, and vertical, is applied on a 4×4 TU or a 8×8 TU depending on its intra prediction mode. A good coefficient scanning order can reduce the number of syntax elements to be coded. For example, if the transform coefficients are clustered in the first few columns, the vertical scan forms a coefficient description consisting of several non zero values followed by a queue of zero values. However, in the case of diagonal scan, the coefficient description is composed of zero and non zero values alternatively. The fact that prediction residuals are intra mode dependent has been also exploited by several coding methods [YK08].

As opposed to H.264/AVC, which interleaves the syntax element for coding the significance of coefficient and the last significant coefficient, HEVC separates the coding of last significant coefficient flag from the coding of the significance coefficient flag, and codes at first the position of the last significant coefficient in a TU block. This method has no performance penalty with respect to the interleaved coding in H.264/AVC, and at the same time, it has several advantages such as reducing the total number of syntax elements for coding the last position, and enabling the usage of efficient context model when coding the bins with CABAC [SJK11].

A method known as sign data hiding [CHJ11] [YWH+12] is used for further compression improvement. The sign bits are coded conditionally based on the number and position of coded coefficients. When sign data hiding is used and there are at least two nonzero coefficients in a 4×4 sub-block and the difference between the scan positions of the first and the last nonzero coefficients is greater than 3, the sign bit of the first nonzero coefficient is inferred from the parity of the sum of the coefficient amplitudes. Otherwise, the sign bit is coded normally.

1.1.5 Entropy coding

HEVC has a single entropy coding mode based on the context adaptive binary arithmetic coding (CABAC) engine. Since the statistical characteristics of syntax elements vary a lot, the CABAC engine defined several probability context models and two operating modes, regular mode and bypass mode, for coding them more efficiently. The regular mode has a context modeling stage, in which a probability context model is selected. Then the coding engine uses the selected context model to code the binary symbol (bin) of the syntax element and updated the context after each bin is coded. The bypass mode has a simple equi-probable model. The applying of bypass mode can increase the throughput and the coding speed of CABAC because it does not require context derivation and adaptation.

1.1.6 Encoder control

The HEVC standard does not specify the encoding process. However, the HEVC software, the HM, implements an encoder which aims at optimizing the objective performance, under the rate-distortion compromise. This encoder has to estimate from all possible modes of coding a CTU the best coding mode in a rate-distortion sense. The set of available coding modes that needs to be investigated is a combination of all following decisions:

- CU partitioning structure.
- Decision of CU coding type: intra coded, inter coded or skip.
- PU partitioning structure and intra prediction mode of a intra coded CU.
- PU partitioning structure and motion information of a inter coded CU.
- TU partitioning structure and transform coefficient coding.

In HEVC reference software, the concept of Lagrangian encoder control is applied for all coding decisions mentioned above. The approach of mode decision by minimizing a Lagrangian cost function was proposed in [WG93], [WLM+96]. Given a set of applicable coding mode C, and a block B containing samples s_i, $i \in B$, the selection of the best coding mode c^* can be expressed as

$$c^* = \arg\min_{c \in C} D(c) + \lambda \cdot R(c) \tag{1.1}$$

where the term $D(c)$ represents the sum of distortion between the original samples s_i and its reconstruction $\hat{s}_i$ using the coding mode c. More precisely, the used distortion measure $D(c)$ can be defined as

$$\sum_{i \in B} |s_i - \hat{s}_i|^\beta \tag{1.2}$$

where $\beta = 1$ refers to the sum of absolute differences (SAD) and $\beta = 2$ refers to the sum of squared differences (SSD). The term $R(c)$ represents the real number or an

estimation of the bits that are required for representing the samples s_i when the block is coded with mode c. It includes the bits required for signaling all associated information of coding mode, such as partitioning structure, intra prediction mode, motion vectors, reference picture indexes, as well as the bits required for transmitting the transform coefficient levels representing the residual signal.

In HEVC, SSD is used as the distortion measure for all coding decisions except for the motion estimation of inter prediction. As a consequence, the encoding is optimized with respect to the mean square error (MSE) or PSNR. The technologies developed in this thesis can be viewed as new coding modes and can use the MSE distortion measure as well. The proposed technologies are involved in the whole process of coding mode decision, and the selection of these coding modes by the encoder means that these new technologies outperform the HEVC coding tools in a rate-distortion sense.

1.2 Other tools of image compression

Besides the inter picture coding and intra picture coding adopted in the video coding standard, several approaches are developed to exploit the temporal and spatial redundancy of a picture such as Template Matching and Vector Quantization.

Template Matching

In image processing, template matching (TM) has been widely applied on texture synthesis problem in computer vision [WL00] [Ash01], and was introduced in [SKS+04] and [TBS06] for inter picture coding and image coding respectively. The template matching process of 4 ×4 block is illustrated in Fig. 1-6.

The template matching algorithm employs the available neighboring pixels of a block (available at both encoder and decoder) as a template, measures the similarity between the template of the target block to be predicted and that of the candidate block in the previously reconstructed regions, and chooses the most similar one as the predictor. The matching operation typically use the Sum of Absolute Differences

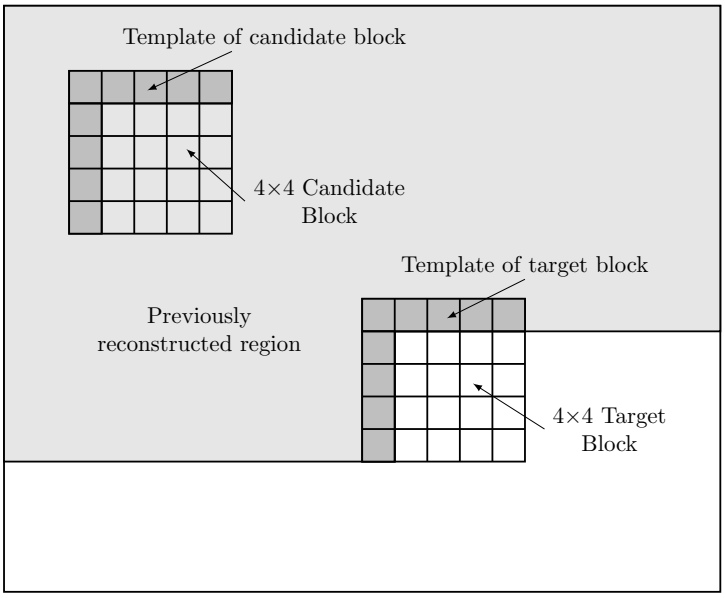


Figure 1-6: An example of template matching for 4×4 block.

(SAD) or Sum of Squared Errors (SSE) between two templates as criteria for retrieving those with higher similarity.

The major advantage of TM is that the same process can be repeated at the decoder without side information such as motion vector for identifying the predictor. Furthermore, since no additional cost is required by TM, the target block can be partitioned into smaller blocks for improving the accuracy of prediction. In addition, an average of multiple prediction candidates identified by TM can be used for getting a better prediction [TBS07].

Vector Quantization

Vector Quantization (VQ) is an efficient and simple approach for image compression [GG92]. In image compression, an image is divided into several blocks and each block contains a set of pixels which can be view as a source vector. The objective of VQ is the representation of the source vectors in an image by a set of reference vectors.

Usually, this set of reference vectors is known as codebook and each reference vector in this codebook is called code vector. The encoding step of VQ refers to the process of searching for the code vector in the codebook which is most similar to the source vector. The encoded results are the index of the code vectors used for representing the blocks. During the decoding, the receiver uses the same codebook to translate the index into their corresponding code vectors for reconstructing the image.

One of the fundamental element of VQ is the codebook generation procedure. A good codebook is one that minimizes the distortion between the original image and the reconstructed image. Several techniques have been proposed for codebook generation. The most commonly used method is the Generalized Lloyd Algorithm (GLA) which is also called Linde-Buzo-Gray (LBG) algorithm [VB93]. LBG is an easy and rapid algorithm but has the local optimal problem [HH93]. An alternative algorithm is Kohonen Self-Organized-Maps [Koh90] which gradually modifies initial code vectors by training vectors.

In this manuscript, we extend the use of VQ form pixel signal compression to residual signal compression. How to apply the LBG and Kohonen algorithm for generating codebooks for residual compression are discussed in Chapter 3 in this manuscript.

1.3 Conclusions

In this chapter, we presented the key features of the current state-of-the-art, HEVC standard and two image coding tools, Template Matching and Vector Quantization. In the subsequent sections, we will describe how to introduce the new image coding tools in HEVC standard for residual compression in order to further improve the intra coding efficiency.

Chapter 2

Intra residual coding using spatial prediction

The HEVC standard contains several elements for achieving equivalent subjective reproduction quality by using 50% less bit rate over the H.264/AVC standard. It supports a quad-tree structure for flexibly splitting an image into Coding Units (CU) with sizes from 64×64 pixels to 8×8 pixels. Then, a CU can be further divided into one or four squared Prediction Units (PU), each of which specifies a region of intra prediction with an individual intra prediction mode. Moreover, 35 intra prediction modes allow to more accurately deal with smooth regions as well as directional structures. Although the intra prediction in HEVC is efficient at reducing the spatial correlation in pixel signals, the accuracy of intra prediction is limited in regions with complex textures and structures. As a result, the intra prediction residue in these regions could have larger magnitudes and account for a major portion of the bit stream of the intra coding.

In this Chapter, we propose an approach of Intra Residual Prediction (IRP) subsequent to the intra prediction in HEVC coding scheme for reducing the remaining redundancy in the residual domain. In the proposed IRP, a previously reconstructed residual block in the causal region of an image is used to predict the residues of the current block. In order to efficiently identify the best predictor for the current residual block, the statistical characteristics of the intra residual predictors are studied, and

several techniques for constructing a predictor candidate list are explored. The upper bound of the bit rate reduction indicates that the intra residual prediction could provide significant coding gains. Experimental results show that the proposed IRP can reduce the remaining correlation in the residual domain and improves slightly the coding efficiency.

2.1 Block based prediction in the residual domain

The conventional intra prediction of the HEVC standard is conducted by using neighboring pixels in the surrounding of the current block to be predicted. Thus, the intra prediction, which exploits the local similarity in the original pixel domain, is efficient at reducing the spatial redundancy. Besides, block-based prediction that exploits the non local similarities in the pixel domain have been proposed in literatures, such as Template Matching [TBS06] [TBS07] [ZYE+08] [GWL08], Epitome-based image compression [CGT+11] and Sparse approximation [TG09] [TG10] [TG12]. The approach of Intra Block Copy [LCW13] [PSG+13], which is also based on block prediction, has been integrated in the Screen Contents Coding (SCC) extension of HEVC.

The main focus here is the remaining redundancy in the residual domain. To improve the intra coding efficiency, the concept of prediction has been extended to the intra prediction residue. An approach of Intra Residual Prediction (IRP) is proposed to exploit the non-local similarities in the residual domain. In IRP, after the intra prediction, the residues of the current block will be predicted by a previously reconstructed residual block. Let us consider the conventional intra prediction of the HEVC standard as a First Order Prediction (FOP), thus, the proposed IRP can be viewed as a Second Order Prediction (SOP) subsequent to the FOP. The scheme of intra coding with IRP is shown in Fig. 2-1.

Let $\mathbf{s}$ denote the original pixel signals in the current block to be predicted, and let $\mathbf{p}_1$ denote the prediction generated during the intra prediction, the intra prediction

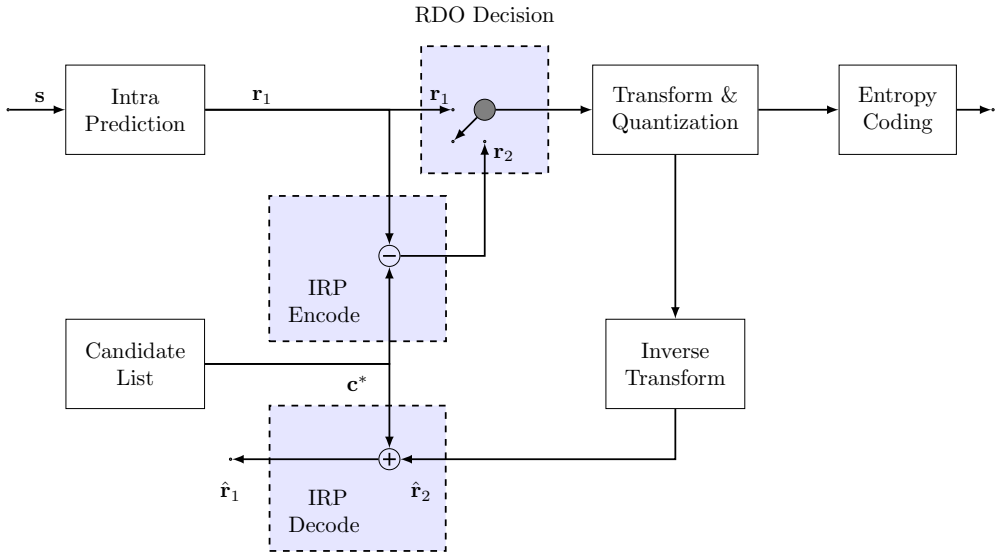


Figure 2-1: The scheme of residue coding using Second Order Prediction (Intra Residual Prediction). The original intra prediction residue is denoted by $\mathbf{r}_1$, the predictor is denoted by $\mathbf{c}^*$, and the second order residue is denoted by $\mathbf{r}_2$. The reconstructed first order residue and the second order residue are denoted by $\hat{\mathbf{r}}_1$ and $\hat{\mathbf{r}}_2$ respectively.

residue which can be referred as to the first order residue $\mathbf{r}_1$ is computed by

$$\mathbf{r}_1 = \mathbf{s} - \mathbf{p}_1 \tag{2.1}$$

In the subsequent step of IRP residual coding, let $\mathbf{p}_2$ denote the prediction of the IRP which refers to the actual values of a reconstructed residual block in the causal region of the current block. Thus, the residue of the intra residual prediction, which can also be referred as to the second order residue $\mathbf{r}_2$ is computed by

$$\mathbf{r}_2 = \mathbf{r}_1 - \mathbf{p}_2 \tag{2.2}$$

The second order residue $\mathbf{r}_2$ is then sent to the conventional residual coding step including DCT/DST-based transform [SJN+12], scalar quantization of transform coefficient levels and entropy coding of scalar quantized symbols.

2.2 Residual predictor search region

The predictor of the IRP is a block of reconstructed residues selected from the region consisting of previously decoded residues, which is also referred to as the causal area of the current block. During the intra coding of an image, the full predictor search region that covers the whole causal area of the current residual block expands gradually, as well as the number of the predictor candidates in the search region. For an image of large dimension, searching for the best predictor in the whole causal area is very expensive due to the considerable computational complexity. In addition, this region may contain many reconstructed residual blocks that are not suitable to predict the current block and this may cause a waste of bits to specify the best predictor in the search region. As a result, the IRP predictor search region should be restricted to a relatively small area containing only suitable candidates.

How to restrict the predictor search region can be converted into a problem of determining the location where the predictors accumulate mostly. To solve this, we studied the statistical characteristics of the reconstructed residual blocks in the causal area which can be used to predict efficiently the current block in the IRP residual coding. As our objective is to find the suitable candidates of residual prediction, the signalization cost such as communicating the predictor position in the search region to the receiver is out of consideration at this stage. Consequently, the predictor is actually the closest matching block of the current TU in terms of distortion, which is measured by the Sum of Squared Errors (SSE).

The following definitions of the location are used in our study. The location of a residual block using the image as reference is determined by the absolute coordinate of the residue sample in the block's upper-left corner. For instance, $(\mathbf{x}_p, \mathbf{y}_p)$ denotes the absolute coordinate of the predictor in the causal area and $(\mathbf{x}_c, \mathbf{y}_c)$ denotes the absolute coordinate of the current TU block to be predicted. The location of the residual block with respect to the current TU block, which is actually the relative coordinate of the predictor, can be expressed as $(\mathbf{x}_p - \mathbf{x}_c, \mathbf{y}_p - \mathbf{y}_c)$.

Consequently, the study of the predictor's location is divided into two categories.

The first category deals with the location of predictor using the image as reference. The objective of the study is to identify the region where the predictors accumulate. The second one addresses the location of predictor with respect to the location of the current TU which aims at verifying that the locations of the predictor follow a specific behavior, such as for example, being usually close to the current TU block.

A. Location of predictor using the image as reference

The study is performed on the frames of the *BasketballPass* sequence [Bos08] with resolution of 416×240 pixels per frame. The sequence is encoded in All-Intra mode with HEVC coding scheme. This is representative of the predictor distribution that is encountered for any sequences of the HEVC test set.

During the encoding, the location of the closest matching block for each 4×4 TU in the full causal area is recorded. Fig. 2-2 represents the statistical results of these locations using the image as reference. As the search region covers the whole causal

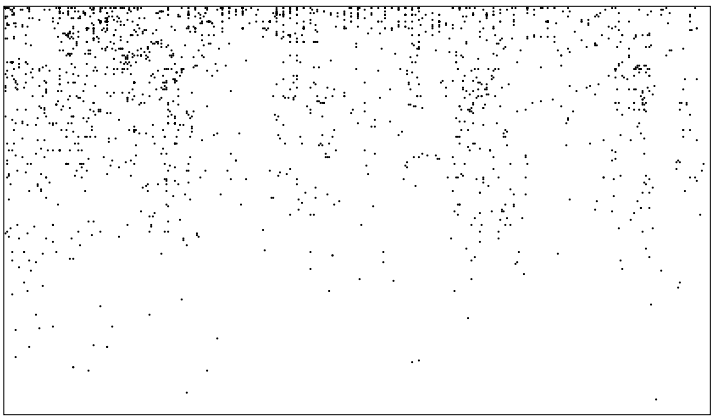


Figure 2-2: Special distribution of the localization of the closest matching block using the image as reference. The simulation is performed with encoding in All-Intra mode the first frame of the *BasketballPass* sequence in the resolution of 416×240 pixel per frame.

area of the current TU, the frequency of using the previously reconstructed residual blocks as a candidate of predictor is higher than that of using recently reconstructed

residual blocks. However, we observed that the positions of the predictors have a relatively regular distribution in the whole image. This means that it is unnecessary to find the predictor in the whole causal area of the current TU, thus the size of the search region can be reduced. In addition, some decoded residual blocks seem to have specific characteristics that make them good predictors and some reconstructed residual blocks have never been used as predictor. So how to distinguish good predictors from the search region is an important issue which need to be investigated.

B. Location of the predictor using the position of the current TU as reference

From the positions of the current block and its closest matching block recorded during the encoding of the sequence, we can deduce the location of the predictor with respect to the current TU. Fig. 2-3 shows the distribution of the relative location of the closest matching block of each 4×4 TU blocks.

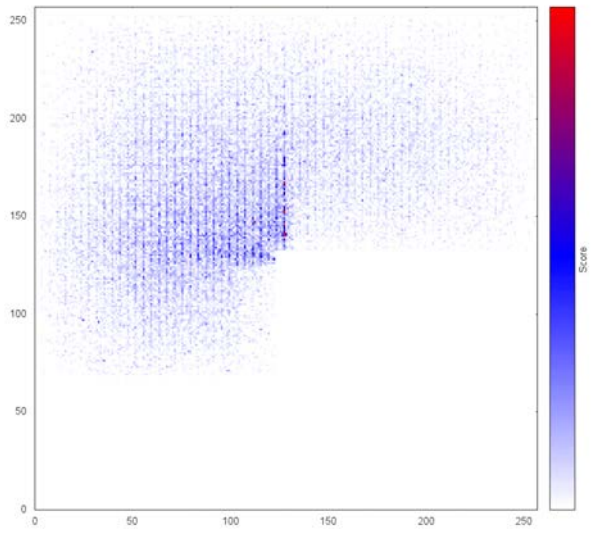


Figure 2-3: Statistical result of the localization of the closest matching block using the position of the current 4×4 TU as reference. The simulation is performed with encoding in All-Intra mode the first frame of the *BasketballPass* sequence in the resolution of 416×240 pixel per frame.

The center point in the figure is the position of the current TU block, and the points around represent the relative positions of the predictors. The darker areas in the figure illustrate the places where predictors tend to accumulate.

From the statistical results, we have the following observations. First, the predictors' location are distributed irregularly in the causal area. This means that in the whole causal area of the current TU block, some of the reconstructed residual blocks are useful as predictor, others are not. So, criteria are needed for identifying the appropriate predictor candidates. Second, predictors of 4×4 TU blocks tend to aligned with the 4×4 grid. This is because in the HEVC standard, the intra prediction mode is determined at PU block, but the intra predictor is actually generated at each TU block. As discussed in Section 1.1.1, the advantage of this approach is the possibility of always utilizing the nearest neighboring reference samples from the already reconstructed TU. Thus, a discontinuity of residuals can be observed at the border of two neighboring TU blocks. As a result, the candidates containing residues overlapping the grid may not be suitable for predicting the residues of the current TU block. Furthermore, predictors are not limited to the adjacent blocks of the current TU. In fact, they seem to accumulate in a L-shape region with respect to the current TU block. This can be attributed to the tendency of energies to exhibit horizontal and vertical structures, since other sequences produce the same behavior. The intra prediction residues of the blocks in these area also share the same structure, so it is easier to find from this area the closest matching block for the current TU block.

The conclusions of the statistical studies about the predictor location can be summarized as follows:

- The IRP search region can be restricted to a smaller causal area adjacent to the current TU.

- The predictors of 4×4 TU blocks in the search region are reconstructed residual blocks aligned with the 4×4 grid.

- The search region can be further restricted to an area of L-shape with respect to

the current TU.

These three features are the guidelines for designing the search region of IRP.

2.3 Upper bound of the RD gain with a limited candidate set

2.3.1 On/Off indicator

Systematically applying the IRP on every TU block in an image is not realistic because of certain constraints. First, when applying the IRP on a TU block containing residues with complex structures, it is difficult to find a reconstructed residual block in the restricted search region which having similar structure as the current block. Thus the second order residue of this TU block may contain even more energy than the first order residue of the intra prediction. Second, applying the IRP on the residual blocks having small magnitude samples would probably be worthless due to the transmission of the bits required for identifying the predictor's location. As a result, the IRP should not be systematically applied in the residual domain for each TU, and an indicator is needed for notifying whether the residual block is encoded with the IRP or with the conventional approach.

The purpose of IRP is to have less information remaining in the second order residue so that less bits are required to transmit it to the receiver. The IRP is activated for the current TU if and only if the second order residue has less energy remaining than the intra prediction residue, otherwise, the IRP is not activated. The energy of the first order residue $\mathbf{r}_1$ is computed by

$$E(\mathbf{r}_1) = ||\mathbf{r}_1||_2^2 \tag{2.3}$$

Let C denote the set of candidates in the search region, the second order predictor $\mathbf{p}_2^*$ of the current TU is the one that minimizes the energy of the second order residue. It

can be expressed as

$$\mathbf{p}_2^* = \arg\min_{\mathbf{p}_2 \in C} ||\mathbf{r}_1 - \mathbf{p}_2||_2^2 \tag{2.4}$$

Thus, the second order residue $\mathbf{r}_2$ are computed by

$$\mathbf{r}_2 = \mathbf{r}_1 - \mathbf{p}_2^* \tag{2.5}$$

The energy of the second order residue is computed by

$$E(\mathbf{r}_2) = ||\mathbf{r}_2||_2^2 = ||\mathbf{r}_1 - \mathbf{p}_2^*||_2^2 \tag{2.6}$$

For a given TU block, the IRP is activated only if the following condition is satisfied.

$$||\mathbf{r}_1 - \mathbf{p}_2^*||_2^2 < ||\mathbf{r}_1||_2^2 \tag{2.7}$$

2.3.2 Estimation of an upper bound of the RD gain

Before investigating the criteria for selecting the predictor in the search region, it is useful to estimate the potential Rate Distortion (RD) gain that the IRP could achieve. In order to estimate an upper bound of the RD gain, we limit the L-shape search region to contain 40 candidates composed of 20 candidates in the horizontal line on the left side of the current TU and 20 candidates in the vertical line on the top of the current TU. Each candidate is aligned with the 4×4 TU grid of the frame as discussed in Section 2.2. Fig. 2-4 shows an example of the search region for a 4×4 TU block.

As the IRP is not systematically applied for each TU, a TU level flag irp_tu_flag is used to indicate whether the TU block is encoded with the IRP. If the condition in Equation (2.7) is satisfied, the TU block is encoded with the IRP and the TU flag is set to 1, otherwise, the IRP is not activated and the TU flag is set to 0. The maximum gain is achieved when the predictor can be selected automatically with the predefined criteria so that no additional bits are required for specifying the predictor. The simulation is performed with only encoding the TU flag in the CABAC code with

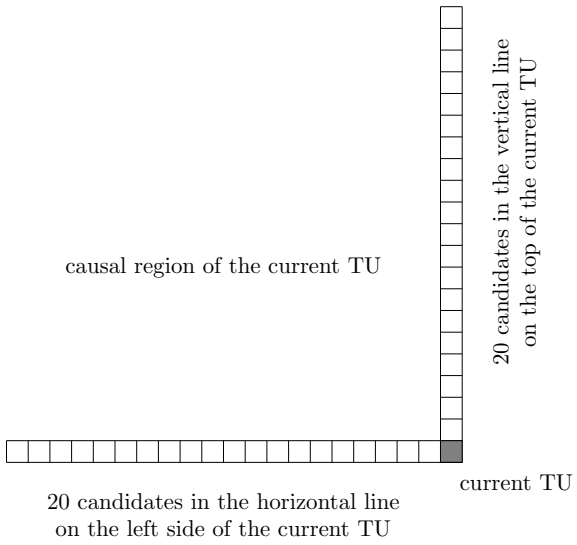


Figure 2-4: L-shape search region. It contains 20 candidates in the horizontal line on the left side and 20 candidates in the vertical line on the top of the current TU.

1 bit. The position of the predictor is not encoded but they are stored aside for the decoding. The experimental results in Table 2.1 show that the IRP could provide a potential gain of around 11% on bit rate reduction compared with HEVC encoding scheme.

Quantization Parameter	**bit-rate saving (%)**	**PSNR (dB)**
22	−5.69	−0.00
27	−8.94	−0.00
32	−13.06	+0.07
37	−17.21	+0.23
average	−11.22	+0.07

Table 2.1: Bit rate savings of video coding under All-Intra coding mode. Simulation is performed with encoding the first 8 frames of the sequence of *BasketballPass*. The resolution of the sequence is 416×240 pixels per frames. The IRP is performed on 4×4 TU blocks and the search region is the L-shape region with respect to the current TU block. The search region contains 40 candidates. This is an estimation of the upper bound of the performance because the signalization cost of residual predictor is not taken in account.

It should be noted that this is an upper bound of the performance with only 40 candidates in the search region. However, having some efficient criteria for automatically

identifying the best predictor is ambitious. Usually, criteria are used for eliminating useless candidates in the search region or for selecting appropriate candidates. In addition, side informations about the predictor are also required for specifying the predictor in these appropriate candidates. How to encode efficiently the actual predictor needs to be investigated. In the next section, we present the techniques used for eliminating useless candidates and for identifying appropriate candidates in the search region. We also discuss how to construct a candidate list, so that the side information to be transmitted is the index of the predictor in the candidate list. These elements are emphasized because the more efficiently the candidates are selected from the search region, the smaller will be the candidate list size and the better performance could be achieved by the IRP.

2.4 Candidate list construction tools

The candidate list construction is the fundamental element of IRP. The size of the candidate list determines the number of bits required for communicating the predictor to the receiver. A candidate list of size N means that the encoder will use $\log_2 N$ bits to code the index of the predictor. How to efficiently identify adequate candidates to construct a relatively small candidate list is the main focus of this section. To achieve this, our approach has three steps of separate functionality: elimination of useless candidates, ordering candidates in terms of similarity and construction of the candidates list.

Firstly, a step of pretreatment is designed for the candidates in the L-shape search region with the purpose of eliminating useless candidates based on a threshold of candidate's energy. The second step consist of ordering candidates in terms of their similarity to the current block. Two alternatives, Template Matching and Side Matching, are proposed to sort the candidates after the elimination. In the last step, we pick up the candidate one by one from the sorted list from the most similar to the least similar to the current block and put it into the candidate list with the condition of avoiding the

duplication of similar candidates. At the same time, we try to make use of previous predictors outside of the search region in a way of putting them into the candidate list as well. We describe in greater details these steps in the following sections.

2.4.1 Elimination of useless candidates

As discussed in Section 2.3, the IRP is rarely used on the residual blocks which have smaller magnitude samples. So the candidates with small energy can be eliminated. Thus, the first criterion identifying the candidates from the search region is the energy of the candidate. Let δ denote the threshold of the candidate energy, the candidate having energy $E(\mathbf{c}) < \delta$ is eliminated. A larger threshold can decrease significantly the number of the candidates so that the bits required for specifying the predictor can be reduced. But a larger threshold may also eliminate some suitable candidates as well. To determine the adequate value of the threshold δ, we performed the same simulations represented in Section 2.3.2 with a range of δ value applied on the 40 candidates. The experimental results are shown in Fig. 2-5. Note that increasing the threshold of the candidate energy can indeed reduce the number of candidates. The bit rate reduction stays unchanged before the threshold of the candidate energy reaches a certain value, and then it decreases with the threshold. We conclude that eliminating some candidates of small energy in the search region will not affect the IRP performance. But there are still too many candidates in the search region and some other criteria are needed for identifying the really useful candidates.

2.4.2 Ordering candidates in terms of similarity

A. Ordering candidates with Template Matching

The method of intra coding with Template Matching, as proposed in [TBS06], assumes that the similarity between the predictor candidate and the current block can be represented by the similarity of their templates. A template refers to a region adjacent to the block, which is typically a L-shaped region to the left and above the block. An

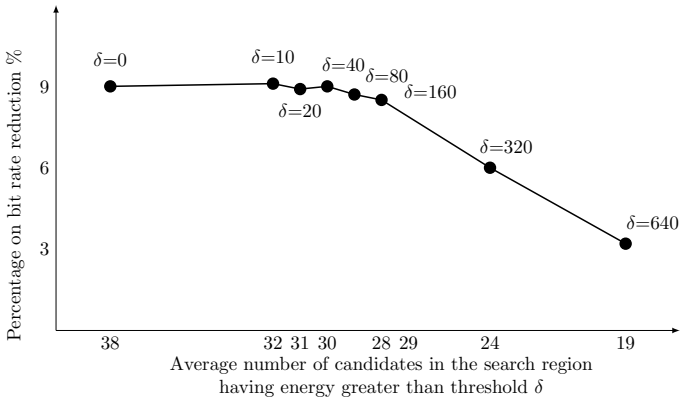


Figure 2-5: The impact of eliminating candidates with energy smaller that a threshold δ. When applying the IRP, the bit rate reduction remains unchanged when the threshold is relatively small, it then decreases when the threshold of the candidate energy increases.

example of the template of a 4×4 block is shown in Fig. 1-6. The 9 pixel samples in gray form the template of the current block. Matching means that the predictor is identified by minimizing the distortion of the reconstructed pixels of the template of the current block and that of the candidate.

In our approach, Template Matching is applied in the residual domain. Here, the templates of the current block and of the candidates contain reconstructed residual samples. The similarity of the current residual block and the candidates can be then evaluated by the similarity of their templates. Let $\mathbf{r}_1$ and $T(\mathbf{r}_1)$ denote the residuals of the current residual block and of its template respectively. Let $\mathbf{p}_2$ denote the residuals of the predictor candidate, and $T(\mathbf{p}_2)$ denotes the residuals of its template. The Sum of Squared Error (SSE) measures the similarity between the templates of the current block and the candidate. Thus,

$$SSE(T(\mathbf{r}_1), T(\mathbf{p}_2)) = ||T(\mathbf{r}_1) - T(\mathbf{p}_2)||_2^2 \tag{2.8}$$

After eliminating worthless candidates by the threshold of candidate energy, the remaining candidates in the search region are sorted by the result of the template match-

ing, the SSE value between the templates of the current block and the candidates, in an ascending order. The candidates having smaller SSE have higher priority to be selected to add into the candidate list.

B. Ordering candidates with Side Matching

In image coding, it is usually assumed that pixel values vary smoothly and neighboring pixels are highly correlated. Thus, the similarity between the reconstructed pixels of the current block and those of the neighborhood can be used as a criterion indicating whether the current block is well predicted. This is the principle of the Side Matching approach for identifying the appropriate candidates. According to Side Matching, the appropriate candidates are those that can produce a reconstructed block which has less discontinuities between the pixel signals in the upper and left boundaries of the current block. In addition, if a candidate can model accurately the first order residue, the second order residue produced by this candidate would be negligible. Thus, the reconstructed block using the predictor candidate $\mathbf{p}_2$ can be expressed as

$$\begin{aligned} \hat{\mathbf{s}} &= \mathbf{p}_1 + \mathbf{p}_2 + \hat{\mathbf{r}}_2 \\ &\simeq \mathbf{p}_1 + \mathbf{p}_2 \end{aligned} \tag{2.9}$$

An example of the surrounding reconstructed pixels and border reconstructed pixels of the current block to be compared is depicted in Fig. 2-6.

Let $\mathbf{s}_N$ denote the reconstructed pixels in the surrounding neighborhood of the current block, and $\mathbf{s}_B$ denotes the reconstructed pixels on the border of the current block. The number of reconstructed pixels in $\mathbf{s}_N$ and in $\mathbf{s}_B$ are different. When computing the SSE between them, the difference is always computed between a pixel on the border of the current block and the nearest pixels among the neighboring pixels outside the current block. So we define the vector of neighboring pixels and the vector of border

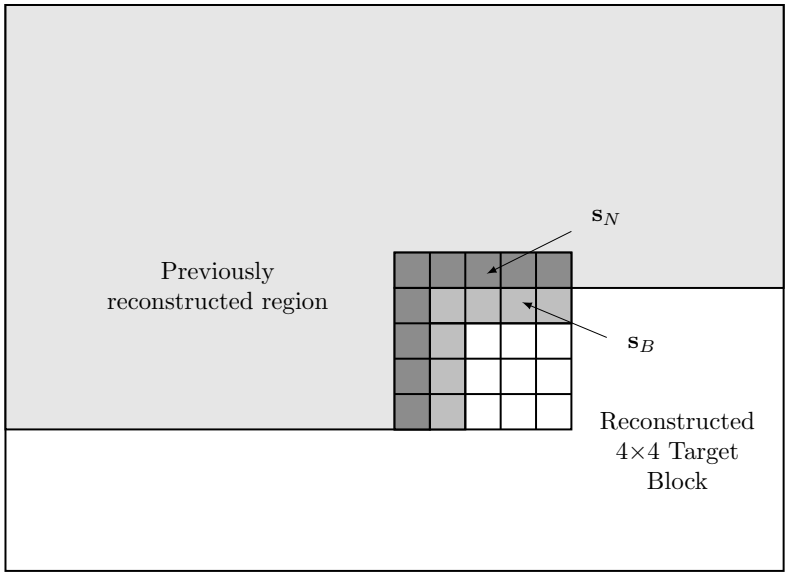


Figure 2-6: Example of side matching for a 4×4 TU block. $\mathbf{s}_B$ denote the 9 pixels in the border of the current 4×4 TU block and $\mathbf{s}_N$ denote the 9 pixels in the surrounding neighborhood of the current block.

pixels as,

$$\begin{aligned} \mathbf{s}_N &= \{x_{-4}, x_{-3}, x_{-2}, x_{-1}, x_0, x_1, x_2, x_3, x_4\} \\ \mathbf{s}_B &= \{y_{-3}, y_{-2}, y_{-1},\ y_0,\ y_0,\ y_0,\ y_1, y_2, y_3\} \end{aligned} \tag{2.10}$$

where the pixel at the upper-left corner of the current block is repeated three times. The SSE is expressed as,

$$SSE(\mathbf{s}_N, \mathbf{s}_B) = ||\mathbf{s}_N - \mathbf{s}_B||_2^2 \tag{2.11}$$

Then, all candidates in the search region are sorted by their SSE in an ascending order and candidates of lower SSE have the priority to be put into the candidate list.

Note that the methods of Template Matching and Size Matching are mutually exclusive. They are further investigated in Section 2.5 to identify the more appropriate solution.

2.4.3 Constructing candidates list

A. Avoiding the duplication of similar candidates

As discussed in 2.3.1, the best candidate to predict the residues of the current TU is the one that minimizes the energy of the second order residues. According to Equation (2.4), the best candidate is the one that most similar to the current block in terms of SSE. In reality, it is possible that two candidates that are very close to each other may provide the same prediction result. In this case, keeping both of them in the candidate list can cause a waste of bit when specify the predictor. In addition, removing the duplication of similar candidates can reduce the size of the candidate list and less bits will be required to transmit the index of predictor. On the other hand, if one of the similar candidates is not a good residual predictor, neither of the others. As a result, the duplication of similar candidates in the candidate list should be avoided during the candidate list construction.

The process of candidate list construction can be summarized as:

- **Step 1** Pick up $\mathbf{c}_i$, which has the minimum value of sorting in the list of ordered candidates.
- **Step 2** For each candidate $\mathbf{c}_j$ in the candidate list, compute $SSE(\mathbf{c}_i, \mathbf{c}_j)$.
- **Step 3** The candidate $\mathbf{c}_i$ will be considered as a duplication of someone in the candidate list if,

  $$SSE(\mathbf{c}_i, \mathbf{c}_j) < \delta \tag{2.12}$$

  where δ_s is a predefined threshold. In this case the candidate $\mathbf{c}_i$ is excluded from the candidate list. Otherwise, the candidate $\mathbf{c}_i$ will be put into the candidate list.
- **Step 4** Remove the candidate $\mathbf{c}_i$ from the ordered list and return to **Step 1**.

Simulations are conduced to determine an adequate threshold δ during the candidate list construction. Because small values can not efficiently avoid the duplication of

similar candidates but large values have risks of including useless candidates into the candidate list.

B. Reusing previously used predictor

When restricting the search region to the L-shape area of the current block, those residual blocks outside of this area are eliminated directly. However, it is possible to find a better predictor of the current block from those residual blocks outside of the L-shape search region. As a result, adding predictors of previously encoded block into the candidate list can be useful because it enlarges the candidate searching scope in including the potential candidates outside the L-form search region.

To make use of these predictors outside the search region, the candidate list is designed as consisting of two sub-parts. The first part of the candidate list is filled with candidates of the L-shape search region of the current block. The second part of the candidate list is filled with the actual predictors of previously encoded residual blocks.

To include the candidates that are outside the L-shape region but are suitable to predict the current block, we provide a method of dynamic dictionary. During the intra residual prediction, the second part of the candidate list is updated by replacing the old predictors with new predictors of recently encoded residual block. Meanwhile, avoiding the duplication of similar predictors in the candidate list is always considered. The update process of this sub-part of the candidate list is presented as follows.

- **Step 1**: If the IRP is performed on the current block, the residual predictor $\mathbf{p}_2$ will be used for updating this sub-part of the candidate list.
- **Step 2**: For each candidate $\mathbf{c}_j$ in the sub candidate list, compute $SSE(\mathbf{p}_2, \mathbf{c}_j)$.
- **Step 3**: The update decision is made between the predictor $\mathbf{p}$ and the candidate $\mathbf{c}_j^*$ in the sub candidate list, with

$$\mathbf{c}_j^* = \arg\min SSE(\mathbf{p}_2, \mathbf{c}_j) \tag{2.13}$$

- **Step 4**: If $SSE(\mathbf{p}_2, \mathbf{c}_j^*)$ is smaller than the threshold δ, the predictor is considered as a duplication of $\mathbf{c}_j^*$ and it is rejected to be put into the candidate list. Otherwise, it replaces $\mathbf{c}_j^*$ and the candidate list update is performed.

During the encoding process, when we try to code the current residual block with the proposed IRP, we first construct its candidate list, and then find the best candidate to predict the current block in order to minimizing the energy of the second order residue. In this section we have presented several tools for constructing the candidate list of intra residual prediction. The experimental results of IRP using these tools are shown in next section.

2.5 Experimental results

2.5.1 Syntax element design

The IRP approach reduces the remaining redundancies by using a reconstructed residual blocks in the causal area to predict the current residual block. Indeed, it is easier to find a block with similar structure and texture in the causal area for a current block of smaller size such as 4×4 than one of a larger size such as 16×16. So, in our experiments, the IRP is applied only for 4×4 TU block, and we would extend it to larger TU blocks if it can significantly improve the intra coding efficiency.

The proposed IRP is integrated in the HEVC intra coding as an option of residual coding. This means that the encoder can choose from the two options for coding a TU residual block. It can either use the intra residual prediction and then code the second order residue with the conventional residual coding scheme containing DCT/DST-based transform, scalar quantization and entropy coding, or the encoder decides to code the first order residue directly with the conventional residual coding scheme if the intra residual prediction does not outperform in the sense of rate distortion. Thus, a TU level flag irp_tu_flag is used for specifying the coding option of a 4×4 TU block. The flag is set to 1 when the intra residual prediction is activated, otherwise it is set to 0. This syntax is coded with CABAC code using one bit.

The size of the candidate list is determined considering the trade-off between the computation complexity and the coding performance. In our experiments, the candidate list contains 16 candidates in total, with half of them selected from the L-shape search region which contains 40 candidates, and the others come from the set of previously used predictors. The syntax element *irp_tu_idx* is used to specify the index of the predictor in the candidate list. As the index of the predictors have a relatively regular distribution, a fixed length code is used for coding this syntax element. Furthermore, our simulations show that template matching lightly outperforms than side matching for identifying useful candidates. So in our experiments, the process of candidate list construction consists of: elimination of useless candidate with a threshold of candidate's energy, sorting of remaining candidates with template matching in a ascending order in terms of dissimilarity from the current block and constructing the candidate list without duplicating similar candidates.

2.5.2 Simulations

The proposed intra residual prediction is implemented in the reference software HM8.0 of the HEVC test model. In our experiments, all test sequences are encoded with All-Intra mode because our algorithm aims at improving the intra coding efficiency. The encoder is configured following the JCT-VC Common Test conditions [Bos08] with All-Intra profile. The performance is evaluated by comparing our algorithm with the conventional video coding scheme in HM8.0. The experiments are performed with quantization parameter values of 22, 27, 32, and 37. Table 2.2 shows the performance of eight video sequences of different resolutions where the negative values mean a rate saving.

The experimental results show that the proposed intra residual prediction provides a bit rate saving of 0.05% compared with the video coding in HEVC. However, about 6.5% of the 4×4 TU blocks on average take the IRP coding mode for residual coding. It should be noted that, without the cost of signalization the side information such as TU level flag and predictor index of intra residual prediction, a bit rate reduction of

sequence class	**bit-rate saving (%)**
ParkScene_1920x1080p	−0.02
Cactus_1920x1080p	−0.04
BasketballDrill_832x480p	−0.12
BQMall_832x480p	−0.06
BasketballPass_416x240p	−0.04
BlowingBubbles_416x240p	−0.04
BQSquare_416x240p	−0.04
RaceHorses_416x240p	−0.01
average	−0.05

Table 2.2: Experimental results of coding sequences with IRP approach. All video sequences are coded under All-Intra coding mode with QPs of 22, 27, 32 and 37. The IRP provides on average 0.05% bit rate reduction compared with HEVC coding scheme. The percentage of 4×4 TU blocks which use IRP residual coding mode is on average about .

11% could be achieved. This is a theoretical upper limit that gives us an indication about the possible improvements. We conclude that the approaches investigated in this chapter for constructing predictor candidate list are not efficient for identifying the best candidate of intra residual prediction.

2.6 Conclusion

In this chapter, we proposed an approach of intra residual prediction which aims at reducing the remaining redundancy in residuals of intra prediction by exploiting non-local similarity in the residual domain within a picture. By studying the statistical characteristics of residual predictors, we found that a current residual block can be predicted by a previously reconstructed residual block having a similar structure. Several techniques for identifying predictor candidates in the causal region of the current block has been investigated. These techniques aim at identifying those residual blocks that are probably suitable to predict the current residual block.

The proposed IRP approach does reduce the redundancy in the residual domain, and provides a small bit rate reduction compared with the HEVC coding scheme. This improvement of intra coding efficiency is not significant due to the fact that the

signalization cost, such as communicating the residual coding mode and the index of the predictor, is too expensive to negate the benefit of residual prediction. We conclude that improvement of intra coding is possible by exploiting the redundancy in residual domain but others techniques need to be investigated.

Chapter 3

Intra residual coding using mode dependent vector quantization

The approach of intra residual prediction presented in Chapter 2 aims at reducing the remaining redundancy in the residual domain by exploiting the spatial correlation in adjacent residual blocks. One of the key elements in this approach is to construct a candidate list of reconstructed residual blocks and to select from the list a predictor for the current residual block. We have investigated several techniques and criteria for filling the candidate list with more adequate reconstructed residuals, so as to improving the accuracy of intra residual prediction with this candidate list. However, a side information used to specify the selected predictor needs to be transmitted to the receiver. As a result, most techniques yield an unfavorable rate distortion compromise: the gain in prediction obtained by previously decoded residuals is negated by the bit rate required to transmit the side information.

Interestingly, from the viewpoint of vector quantization, a candidate list of reconstructed residual blocks can be seen as a codebook of code vectors. Thus, the concept of intra residual prediction with a candidate list is very similar to the concept of vector quantization using the candidate list as a codebook. This inspired us to design a real vector quantization scheme for exploiting the correlation in intra prediction residues. Although vector quantization is a well-known technique for exploiting correlation in

a vector of samples, it is unclear whether it could be used for quantizing the intra prediction residue. Thus, how to utilize vector quantization in coding residual signals and what would be the related potential benefits are the main purposes of this chapter. The structure of this chapter is as follows: Principles of vector coding and vector quantization are described at first. Then we introduce the approach of residual coding using Mode Dependent Vector Quantization (MDVQ). Finally, the section of experimental results shows that the proposed MDVQ approach provides a significant increase in coding efficiency of the HEVC-based coding scheme.

3.1 Vector coding

3.1.1 Vector quantization

It has been pointed out in [GG92] that no other coding technique exists that can do better than vector quantization (VQ) for quantizing a signal vector. Compared to scalar quantization, VQ is a more efficient data compression technique which exploits correlation in a vector directly. A vector quantizer Q can be defined as a mapping of K-dimensional Euclidean space R_K into a finite subset C of R_K. Let $C = \{\mathbf{c}_i\}, i = 1, ..., N$ be a subset of N reproduction vectors and $\mathbf{c}_i \in R_K$, the quantizer Q can be expressed as:

$$Q : R_K \rightarrow C \tag{3.1}$$

Usually, the subset C is called codebook and it has N vectors, and each vector is called codeword or code vector. Given a source vector $\mathbf{x} \in R_K$ which is mapped to a code vector $\mathbf{c}_i$, the quantized source vector is denoted by:

$$Q(\mathbf{x}) = \mathbf{c}_i \tag{3.2}$$

The difference between the source vector and its associated code vector is referred to as the quantization error. The distortion of the vector quantization can be calculated

by:

$$\begin{aligned} D &= ||\mathbf{x} - Q(\mathbf{x})||_2^2 \\ &= ||\mathbf{x} - \mathbf{c}_i||_2^2 \end{aligned} \tag{3.3}$$

Minimizing the distortion of quantization for all source vectors is the main focus of vector quantizer design. Thus, the best codebook C for vector quantizing a set of source vectors has to satisfy two criteria: the Nearest Neighbor Condition and the Centroid Condition.

- **Nearest Neighbor Condition**: Let S_n denote the subset of R_K consisting of all source vectors associated with the code vector $\mathbf{c}_n$, this subset should contain all vectors that are closer to $\mathbf{c}_n$ than any of the other code vectors.

$$S_n = \left\{ \mathbf{x} \in R_K |\ \forall \mathbf{c}_m \in C,\ ||\mathbf{x} - \mathbf{c}_n||_2^2 \leq ||\mathbf{x} - \mathbf{c}_m||_2^2 \right\} \tag{3.4}$$

- **Centroid Condition**: In order to minimize the within-subset distortion, the code vector $\mathbf{c}_n$ should be the average of all vectors that are in the subset S_n.

$$\mathbf{c}_n = \frac{\sum_{\mathbf{x}_m \in S_n} \mathbf{x}_m}{\sum_{\mathbf{x}_m \in S_n} 1} \quad n = 1, 2, ..., N \tag{3.5}$$

In signal coding applications, VQ can be viewed as the cascade of an encoder and a decoder, and both of them use a codebook stored in the form of a look-up table. Given a K-dimensional source vector $\mathbf{x}$ as input, the encoder looks for the best matching vector $\mathbf{c}^*$ in the codebook as a reproduction of the source vector, and produces the index of the selected best matching code vector as an output. The decoder receives an index, and uses this index to retrieve the corresponding code vector in the codebook as the reconstructed source vector. With the condition of minimizing the distortion of vector quantization, the best matching code vector is the one most similar to the source vector $\mathbf{x}$. Usually, this similarity is measured by the Sum of Squared Error (SSE), thus

the best matching code vector can be determined by,

$$c* = \arg\min_{\mathbf{c}_i \in C} ||\mathbf{x} - \mathbf{c}_i||_2^2 \tag{3.6}$$

Supposing that a source vector consisting of K samples can be represented in b bits per sample, and a quantizer using a codebook of size N, the compression ratio of vector quantization R_{VQ} can be calculated as:

$$R_{VQ} = \frac{K \cdot b}{\log_2 \cdot N} \tag{3.7}$$

We can deduce from this equation that given a fixed size codebook, larger is the dimension of source vector and code vector, higher is the compression ratio of vector quantization. Actually, not only the compression ratio increases with the source vector dimension, but also the SSE computation complexity as well as the codebook storage requirement. However, the complexity of searching a code vector in a large codebook can be mitigated by the approaches using a tree-structured codebook [BFOS84] at the expense of a small reduction of compression efficiency. Thus, the storage requirement is often the limitation factor of vector quantization. An elegant solution to the codebook storage problem is *Lattice* Vector Quantization (LVQ) [CS82] where the codebook does not need to be stored at the encoder and the decoder. This is because the codebook for Lattice VQ has a regular structure which is known by both encoder and decoder, and nearest neighbors can be identified by algorithmic steps.

Given a set of source vectors of a fixed dimension, the number of the code vector determines the distortion of vector quantization. Let us consider the case of scalar quantization which can be viewed as a one-dimension vector quantizer. The distortion of scalar quantization can be reduced by decreasing the scalar quantization step. In one-dimension value space, the quantization step represents an interval. Reducing the quantization step actually increases the number of interval in the value space which is actually an increase in the number of one-dimension code vectors. As a result, in scalar quantization or one-dimension vector quantization, reducing quantization distor-

tion can be achieved just by increasing the number of code vectors. Furthermore, the additional code vectors can be obtained by just mathematical computation. Similarly, for a two-dimension vector quantizer, a code vector can be viewed as a representation of a Voronoi cell in Euclidean space. Increasing the number of code vector actually increases the number of Voronoi cells, and reduces the size of each of them. Consequently, the distortion of two-dimension vector quantization can also be reduced by increasing the number of code vectors. However, it is difficult to generate the additional two-dimension code vectors by sample mathematical computation. Usually, the whole codebook need to be reconstructed when the number of the code vector is changed. We can obviously construct a range of codebooks in different size for different quantization distortion requirements, but the storage requirements is still a burden, so that a super-size codebook is out of consideration in our quantizer design.

3.1.2 Vector quantizer design

Among the algorithms of vector quantizer design proposed in the literature, we are interested in two representative solutions. The first one is the k-means algorithm which optimizes code vectors iteratively in using the conditions represented in Equation (3.4) and Equation (3.5). The second one is the Kohonen Self-Organized Maps approach which gradually modifies code vectors based on acquired information in training vectors. These two algorithms have been investigated for quantizer design, and the performance of MDVQ residual coding using the k-means based quantizer and Kohonen based quantizer has also been assessed respectively.

A. Vector quantizer design using k-means algorithm

The generalized Lloyd algorithm, also referred to as k-means algorithm [GG92], is widely used for VQ codebook design. The objective of k-means algorithm is to partition a set of observations into N subsets in minimizing the average within-cluster distortion. Given a training sequence of M observations $\mathcal{T} = \{\mathbf{x}_m\}, m = 1, ..., M$ and an initial codebook $C = \{\mathbf{c}_n\}, n = 1, ..., N$ where $\mathbf{c}_n$ represents a subset, the k-means algorithm

proceeds by alternating between two steps:

- **Assignment step**: Assign each observation to the subset according to Equation (3.4). This step is actually the reproduction of $\mathbf{x}_m$ by the code vector of the subset which $\mathbf{x}_m$ belongs to:

$$Q(\mathbf{x}_m) = \mathbf{c}_n \tag{3.8}$$

- **Update step**: For each subset, calculate the mean value of all observations belonging to this subset by using the centroid condition represented in Equation (3.5) and use this mean value as a new code vector to replace the current code vector.

The k-means algorithm is a simple and efficient method for VQ design, but it suffers from some disadvantages. For instance, the quantization result depends on the initial codebook, especially for a set of observations having an irregular distribution. In this case, if the codebook is initialized with unsuitable values, the convergence will lead to a local minimum and it may be quite different from the global minimization. An example is illustrated in Fig. 3-1. The rectangles and triangles represent the

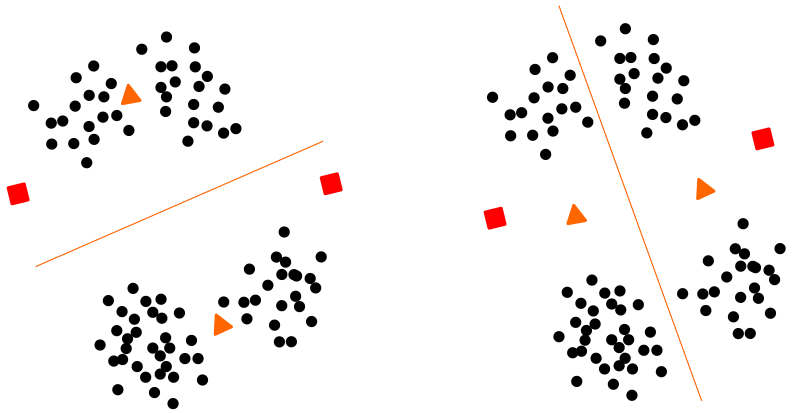


Figure 3-1: A typical example of the k-means algorithm convergence to a local minimum. The rectangles and triangles represent the initial centroids and the final centroids respectively.

initial centroids and the final centroids respectively. In the left side of the picture, the final centroids correspond to the desired centroids and the within cluster distortion is minimum, whereas in the right side, triangles represent the "bad" local minimum centroids of clusters and the within cluster distortion is larger.

A common solution to this problem is to perform the k-means algorithm repeatedly with different initial codebooks. In fact, to avoid the local minimum situation in quantizer design, we have studied the statistical characteristics of intra prediction residues. In order to give an intuitive representation of the distribution of the residual sample values, we need to map the 16-dimensional residual vectors into 2-dimensional residual vectors. Supposing that the intra prediction residue of a 4×4 TU block have been raster scanned into a residual vector, the first and the second residual samples in the residual vector are more correlated, whereas the first residual sample is less correlated to the last residual simple in this vector. Fig. 3-2 shows an example of taking these residual samples to represent the statistical characteristics of intra prediction residues.

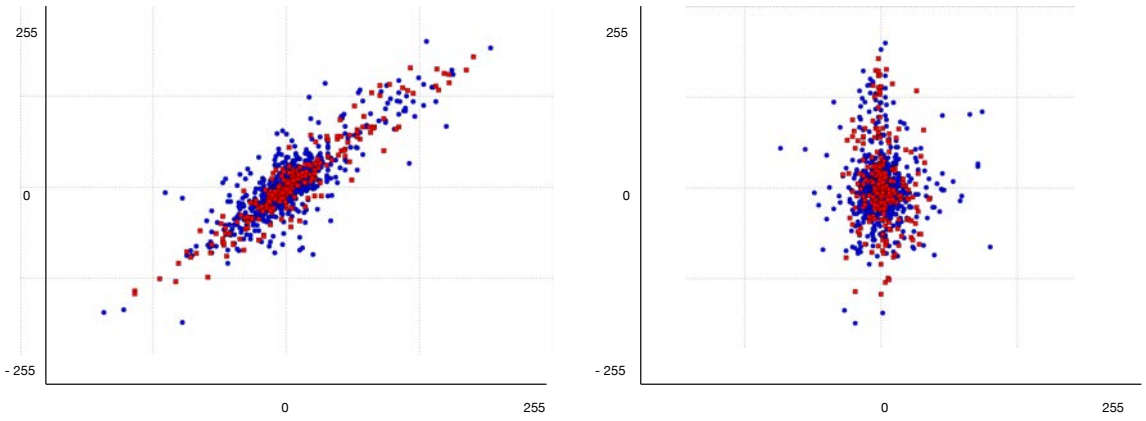


Figure 3-2: An example of statistical characteristics of the sample values of intra prediction residual vectors. The axis X represents the value of the 1^{st} residual sample in a residual vector. In the left side of the figure, the axis Y represents the value of the $2^n d$ residual sample in a residual vector. In the right side, the axis Y represents the value of the last residual sample in the residual vector.

In the left side of Fig. 3-2, the horizontal axis and the vertical axis represent the value of the first and the second residual samples respectively. In the right side of Fig. 3-2, the horizontal axis and the vertical axis represent the value of the first and the last residual sample respectively. We can observe that the residuals have a relatively regular distribution regardless of the correlation between two residual samples. This means that the drawback of the k-means algorithm is negligible for residual samples quantizer design.

In our experimentations, different initial codebooks have been used for performing the k-means algorithm but no significant variation of coding performance have been observed. As a result, we decided to apply the k-means algorithm for mode dependent codebooks design, and the performance of vector quantization using k-means codebooks is used as a basis for comparing codebooks constructed by other algorithms.

B. Vector quantizer design using Kohonen self-organization map

The Kohonen Self-Organizing Map (SOM) is an approach of vector quantizer design which is very different from the k-means algorithm. During the codebook construction, Kohonen algorithm gradually modifies the sample values of code vectors to adapt to input source vectors. Compared with the k-means algorithm, the Kohonen algorithm has two main features: First, in Kohonen algorithm, the code vectors are modified every time a source vector is quantized by this vector, whereas in the k-means algorithm, the code vectors are updated after the mapping of all source vectors. This feature ensures that the Kohonen algorithm adapts code vector to the local changes of the statistical characteristics of the source vectors. Second, an input source vector is used to modify not only the selected code vector, but also its neighboring code vectors in terms of the index of the codebook. At the end of code book construction, these neighboring code vectors in terms of index are actually also neighboring code vectors in the sense of distortion.

Let $\{\mathbf{x}_t\}, t = 1, ..., M$ be a training set of M source vectors, and each source vector $\mathbf{x}_t \in R_K$ is a K-dimension vector. Let C denote a codebook containing N code vectors,

and $\mathbf{c}_i(t) \in C$ with $i = 0, 1, ...N-1$ denote the code vectors at the moment of quantizing the source vector $\mathbf{x}_t$. The Kohonen algorithm can be summarized as follows:

- **Step 1** Before vector quantization, code vectors are initialized with random values.

- **Step 2** Given a source vector, compute the Sum of Squared Error (SSE) between the source vector and each code vector. Select the one minimizing the SSE as the best matching code vector $\mathbf{c}^*(t)$ for quantizing this source vector.

$$\mathbf{c}^*(t) = \arg\min_{\mathbf{c}_i \in C} ||\mathbf{x}_t - \mathbf{c}_i(t)||_2^2 \tag{3.9}$$

- **Step 3** Determine the neighboring code vectors which need to be updated with the selected code vector by a neighbor function,

$$N(\mathbf{c}^*(t), \mathbf{c}_i(t)) = \begin{cases} 1 & \text{if } \mathrm{D}(\mathbf{c}^*(\mathrm{t}), \mathbf{c}_\mathrm{i}(\mathrm{t})) < \mathrm{D(t)} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

  where $D(\mathbf{c}^*(t), \mathbf{c}_i(t))$ measures the Euclidean distance from a code vector to the selected vector, and $D(t)$ controls the maximum distance which decreases with the number of quantized vectors.

- **Step 4** For all neighboring code vectors of the selected code vector obtained in Step 3, as well as the selected code vector itself, adaptively modify them by

$$\mathbf{c}_i(t+1) = \mathbf{c}_i(t) + \alpha(t) \cdot \theta(\mathbf{c}_i(t)) \cdot (\mathbf{x}(t) - \mathbf{c}_i(t)) \tag{3.11}$$

  The multiplying factor $\alpha(t)$ decreases monotonically with the number of update times t. It is defined by,

$$\alpha(t) = A_1 \cdot (1 - \frac{t}{M}),\ 0 < A_1 < 1 \tag{3.12}$$

where A_1 is a constant. The multiplying factor $\theta(\mathbf{c}_i(t))$ is defined by,

$$\theta(\mathbf{c}_i(t)) = A_2 \cdot (1 - \frac{D(\mathbf{c}^*(t), \mathbf{c}_i(t))}{D(t)}) \tag{3.13}$$

where A_2 is a constant. It controls also the amount of change which decreases with the distance of a neighboring code vector to the select code vector.

- **Step 5** Go to Step 2.

As opposed to the k-means algorithm, the Kohonen algorithm is not iterative and does not segment the input training sequence into partitions for calculating the mean of source vectors. The Kohonen algorithm adaptively changes the values of the code vectors by using the adaptation expression in Step 3. It is in fact adaptively partitioning the input vector space into smaller subspaces. A major advantage of the Kohonen algorithm as a method for designing VQ codebooks is that, it is suitable for constructing codebooks with a large sequence of training data without requiring the storage of the training set for repeated iterations as in the k-means algorithm.

3.2 Intra residual coding using mode dependent vector quantization

As discussed in Chapter 2, the accuracy of intra prediction of HEVC is limited in regions with complex textures and structures so that residuals in these regions usually have a larger magnitude. In order to reduce the redundancy in the residual domain, several strategies exploiting the inter block correlation in residual domain have been studied. In these methods, a candidate list consisting of decoded residual blocks is constructed, and a predictor is selected from this candidate list to predict the current residual block. Compared to vector quantization, this candidate list can be viewed as a codebook and the prediction of a residual block is very similar to quantizing a residual vector. However, the experimental results of intra residual prediction lead us to conclude that the code vectors in the codebook filled with decoded residuals are not

adequate neither for intra residual prediction nor for vector quantization. The possible reasons are, for example, that less correlation remains between adjacent residual blocks, more noise signals exist in the intra prediction residues, and the signaling cost for specifying the predictor negates the prediction gain.

Consequently, it is anticipated that improved residual vector quantization can be achieved by using vector quantization with more conventional codebooks. In this section, we present a novel approach of residual coding using mode dependent vector quantization in order to exploit the correlation in residual domain. The proposed MDVQ based residual coding has four characteristics which we want to highlight.

- **Mode dependent vector quantization**: 35 codebooks are designed for quantizing residuals of the 35 HEVC intra prediction modes.

- **Switched vector quantization combined with optional scalar quantization**: The vector quantization step is activated only when the rate distortion optimization identifies a gain. If it is activated, the quantization error of the MDVQ is sent to the conventional transform and scalar quantization step and finally entropy encoded .

- **Rate distortion optimized codebook**: Codebooks are optimized in a rate distortion sense during the design.

- **QP independent codebook**: Codebooks can be applied to residuals produced under different quantization parameters while maintaining rate distortion efficiency.

Vector quantization is an efficient data compression technique. However, the coding efficiency increases with the vector dimension, so does the size of the required codebooks and the complexity of searching for the best matching codeword. Since the codebook size is enormous for quantizing vectors of a very high dimension, vector quantization is generally considered to be too complex to be implemented for practical use. In addition to the above features of MDVQ, an appropriate codebook size is also considered

during the quantizer design. With our approach, good quantization performance can be achieved for high dimensional vectors by using relatively small codebooks.

In the next section, we first describe the scheme of residual coding using MDVQ in a HEVC based encoder. Then the main aspect of MDVQ, the mode dependent codebook construction is presented including the rate distortion optimization and QP independent construction. Simulations in different coding environments are discussed in the end.

3.2.1 Mode dependent vector quantization

HEVC supports planar, DC and 33 directional prediction modes. Similar residual patterns can be observed in regions of same intra prediction modes. As such, in a block of samples predicted with planar mode or DC mode, the residual signals have a relatively homogeneous structure, whereas those derived from angular prediction modes tend to have directional structures. Fig. 3-3 depicts the difference between residuals derived from a planar mode and those derived from a directional mode.

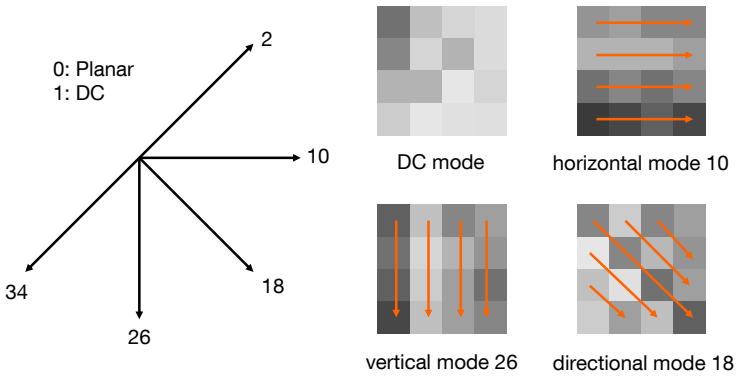


Figure 3-3: The direction of intra prediction modes is shown on the left side. The right side shows the examples of intra prediction residue of 4×4 TU blocks extracted from *BQSquare* sequence in resolution of 416×240.

Actually, this characteristic has already been investigated to improve coding efficiency. In [YK08], a directional transform and an adaptive coefficient scanning are used for coding the intra prediction residue. Indeed, one of the central aspects of our proposed approach is the codebook construction with the purpose of modeling the directional structure in residuals of intra prediction. In other words, for each intra prediction mode, a specific codebook is designed only for residuals of this mode. The process of intra prediction residual coding using mode dependent vector quantization is depicted in Fig. 3-4.

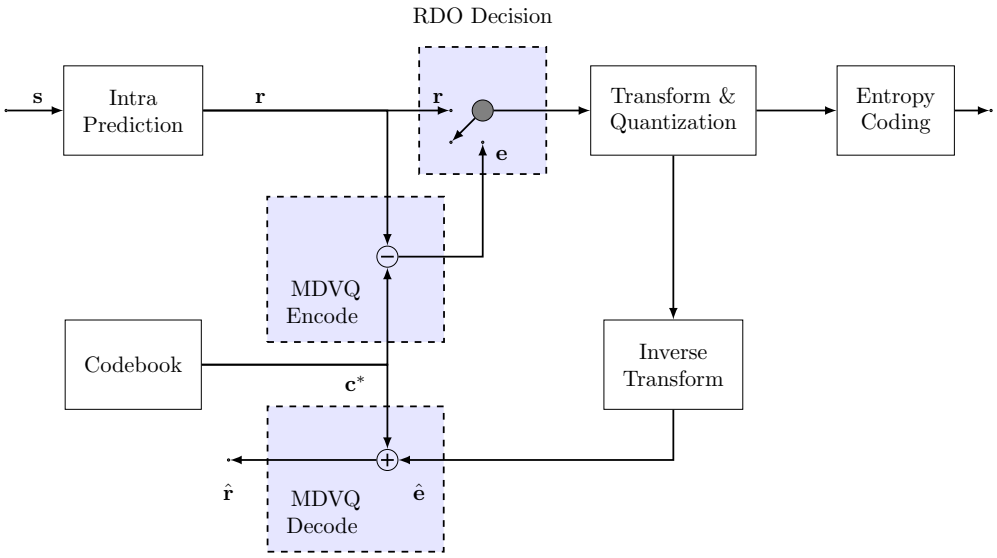


Figure 3-4: The scheme of residue coding using mode dependent vector quantization. The original intra prediction residue is denoted by $\mathbf{r}$, the best matching code vector is denoted by $\mathbf{c}^*$, and the final vector quantization error is denoted by $\mathbf{e}$.

Firstly, a $N \times N$ image block is intra predicted by mode $i \in M$ where M represents the set of available intra prediction modes. The corresponding original intra predicted residuals $\mathbf{r}$ of the same dimension are computed. The MDVQ-based residual coding is performed on this vector of residual samples using a pre-generated VQ codebook C specific for the intra prediction mode i. Formally, let us denote by $\mathbf{c} \in C$ a codeword

in this codebook. Let $\mathbf{e_{r,c}}$ be the quantization error of MDVQ, which represents the difference between the vector of residual $\mathbf{r}$ and the codeword $\mathbf{c}$. The final quantization error will be processed by a series of conventional operations: DCT-based transform, scalar quantization and entropy coding.

The major feature of the proposed MDVQ residual coding scheme we want to emphasize is that, the vector quantization does not replace completely the scalar quantization. Indeed, the final quantization error of vector quantization will be encoded with transform, scalar quantization and entropy coding. As discussed in Section 3.1.1, it is difficult to adapt to the different quantization distortion requirements of high dimension vectors, because changing the number of code vectors in the codebook means that the whole codebook needs to be reconstructed. As a result, preserving the scalar quantization for the vector quantization error is necessary in the proposed MDVQ to satisfy the distortion requirements.

Another important feature of the MDVQ scheme is the mode dependent codebooks. Compared with VQ using a single larger codebook for all residuals from all intra modes, the mode dependent VQ has two major advantages. Firstly, residuals of same intra prediction mode are more correlated than residuals of two different modes. Consequently, a codebook learned only with residual vectors of a specific intra mode can more accurately quantize a source vector of the same mode. In addition, compared with a larger codebook containing code vectors learned for all intra prediction modes, organizing these code vectors into several smaller mode dependent codebooks can reduce the complexity of searching for the best matching code vector. Secondly, using the codebook specific for an intra prediction to quantize the residual vector of the same intra prediction mode can be viewed as a vector quantizer with switched codebooks. This vector quantizer can adaptively select a more adequate codebook for quantizing an input intra prediction residual vector. In addition, this specification does not need to be communicated to the receiver because the intra prediction mode has been already decoded at the decoder side before quantizing or reconstructing a residual vector.

3.2.2 Switched vector quantization combined with optional scalar quantization

We have pointed out in Section 3.1 that vector quantization can not really be used for lossless coding as scalar quantization by simply increasing the number of code vectors. In fact, for the purpose of simplifying the VQ design, a codebook of predefined size is used for each intra mode. For this reason, if we want to be able to cover a wide range of rate-distortion trade-off, the operations of transform, scalar quantization and entropy coding have to be kept for coding the vector quantization error. On the other hand, vector quantization has to transmit the information specifying the selected code vector in the codebook to the receiver. From a viewpoint of rate distortion optimization, residuals of small amplitude are probably not worth to be quantized due to a fixed signaling cost for transmitting the code vector index. As a consequence, the vector quantization for coding intra prediction residue is activated only on those residual vectors where it actually brings a rate distortion benefit.

When coding an original residual block in a frame, vector quantization on this residual vector consists of two steps: find the best matching code vector in the codebook to quantize this residual vector and apply the vector quantization on the residual vector if it is better than the conventional coding approach in a rate distortion sense.

- **criteria of best matching vector** Let us denote $\hat{\mathbf{e}}_{\mathbf{r},\mathbf{c}}$ the reconstructed quantization error at the encoder. The distortion measured on the intra predicted residual coded with MDVQ can be then expressed as:

$$\begin{aligned} \mathcal{D}(\mathbf{r}_i, \mathbf{c}) &= ||\mathbf{r} - \mathbf{c} - \hat{\mathbf{e}}_{\mathbf{r},\mathbf{c}}||_2^2 \\ &= ||\mathbf{e}_{\mathbf{r},\mathbf{c}} - \hat{\mathbf{e}}_{\mathbf{r},\mathbf{c}}||_2^2 \end{aligned} \tag{3.14}$$

  Let $\mathcal{R}(\mathbf{r}, \mathbf{c})$ represent the number of bits required for signaling the vector $\mathbf{c}$ in the codebook, plus the bits needed to code the quantization error. The best matching codeword for an intra predicted residual block is the one minimizing the Lagrangian cost function [WG93] [WLM+96]:

$$\mathbf{c}^* = \arg\min_{\mathbf{c}\in\mathcal{C}} \mathcal{D}(\mathbf{r},\mathbf{c}) + \lambda \cdot \mathcal{R}(\mathbf{r},\mathbf{c}) \tag{3.15}$$

where λ denotes the so-called Lagrange multiplier.

- **Criteria to apply MDVQ based residual coding** MDVQ-based residue coding will be used for a given residue block if the rate-distortion trade-off given by this mode satisfies:

$$\mathcal{D}(\mathbf{r},\mathbf{c}^*) + \lambda \cdot \mathcal{R}(\mathbf{r},\mathbf{c}^*) < \mathcal{D}(\mathbf{r}) + \lambda \cdot \mathcal{R}(\mathbf{r}) \tag{3.16}$$

In other words, the MDVQ-based residual coding is performed only if the rate-distortion cost of coding an image block is reduced compared with the conventional HEVC coding.

3.2.3 Rate distortion optimization of codebook construction

The fact that the MDVQ step is skipped when coding some intra prediction residual blocks indicates that codebooks have to be optimized to code these target residual vectors. As a consequence, the training sequences for codebook construction should contain only adequate training vectors. We propose a method of codebook construction based on an iterative learning process. At each iteration, the training vectors are obtained by extracting from the training video sequences the actual set of residual vectors that satisfy Equation (3.16). As a consequence, the training set is not "polluted" by the residual vectors that would not be coded by MDVQ. In more details, the iterative approach used in our codebook construction procedure can be described as follows.

- **Iteration** 0**:**

  - Step a: The training video sequences are encoded with the conventional HEVC scheme and the residual coding by MDVQ is not applied.

- Step *b*: Vectors of original intra predicted residual are gathered to form the training set for codebook construction.

- Step *c*: The codebooks are constructed with k-means algorithms discussed in Chapter 3.1.

- **Iteration** 1**:**

- Step *a*: The training video sequences are encoded with MDVQ-based residual coding, using codebooks generated at the previous iteration.

- Step *b*: Only the vectors of original intra predicted residual coded by MDVQ are extracted to build the training set of residual vectors for the next iteration. It should be noted that the MDVQ will only be used for coding residuals for which the condition described by Equation (3.16) is fulfilled.

- Step *c*: The codebooks are constructed with the same clustering algorithm used in previous iterations.

Then the steps in Iteration 1 can be repeated several times. The process stops after a predefined number of iterations. In our simulations, we found that the performance of MDVQ based residual coding is improving quickly over the first iteration, then the performance is stable after a certain number of steps often nearing 4. So in our final experiments, all codebooks are learned with 4 iterations.

3.2.4 QP independent codebook construction

In HEVC, the standard scheme of coding the original intra prediction residue of a TU block consists of DCT-based transform, scalar quantization of transform coefficient levels and entropy coding of scalar quantized coefficient levels. However, the Quantization Parameter (QP), which determines the scalar quantization step, has an influence on the magnitude of intra prediction residue of TU blocks. A larger quantization step yields a higher quantization distortion and a lower quality TU block reconstruction. When the reconstructed pixels in a TU block are used as reference pixels to generate the intra or

inter prediction for another TU block, obviously, a prediction of lower quality can be observed. This will lead to an increase in the difference between the intra prediction of the TU block and the original pixels of the next block. As a consequence, the value of the quantization parameter has a direct effect on the magnitude of the original intra prediction residue. Fig. 3-5 shows the differences of original intra prediction residue extracted from the video sequence coded with different QP. It is clear that at higher QP, the residuals have more complex structures whereas at lower QP, residuals are relatively homogeneous.

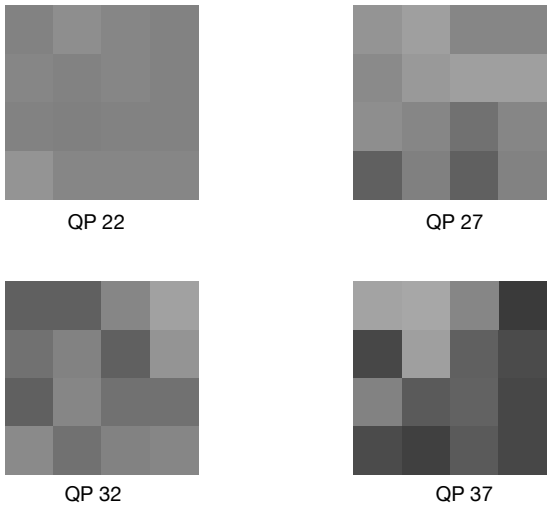


Figure 3-5: Representative examples of intra prediction residue of 4×4 TU blocks extracted from the sequence being encoded with four different QP values: 22, 27, 32, and 37.

As the amplitude of intra prediction residue is related to the quantization parameter, it is natural to construct a specific codebook for a certain QP value. For example, we can encode a sequence with QP values of 22, 27, 32 and 37 to produce the corresponding training vectors, and then use these training residuals for constructing the codebook of the corresponding QP value. A disadvantage of using these QP-specific codebooks is that the storage requirement is much higher if one wants to have codebooks available for arbitrarily QP values. In order to avoid the storage of a larger set

of codebooks, we have first envisioned a coding scheme where a simple codebook would be normalized to fit the magnitude of the residuals produced at a given QP value.

In order to achieved this, we have first studied the code vectors in the four groups of codebooks corresponding to the four QP values by a regression algorithm: taking arbitrarily the codebook at QP value 27 as the reference codebook, to determine the multiplying factor for generating codebooks of the other three QP values. The objective of this study was to find for each QP value the multiplying factor which minimizes the distortion between the QP-specific codebook and the scaled codebook. We observed that the values obtained for the multiplying factor for different QPs are all around 1. This means that the code vectors in codebooks constructed under different QPs are quite similar in terms of magnitude.

To better understand this behavior, let us consider the case where an intra prediction residual vector is encoded with MDVQ instead of the conventional HEVC residual coding approach. Given an intra prediction residual vector $\mathbf{r}$, let D and $R(\mathbf{r})$ denote respectively the distortion and the number of bits required for coding this residual vector with the conventional HEVC residual coding approach. When this intra prediction residual vector is encoded with MDVQ residual coding approach, we use $\mathbf{r}_{vq}$ to represent the final residual after vector quantization. Let D_{vq} denote the distortion, $R(\mathbf{r}_{vq})$ denote the number of bits required for coding the residuals and R_{vq} denote the number of bits required for coding the index of the code vector in MDVQ approach. According to the approach of encoding transform coefficient levels based on the minimization of a Lagrangian function as proposed in [SG88], this TU residual block is encoded with MDVQ activated if and only if:

$$D_{vq} + \lambda \cdot R(\mathbf{r}_{vq}) + \lambda \cdot R_{vq} < D(\mathbf{r}) + \lambda \cdot R(\mathbf{r}) \tag{3.17}$$

where the parameter λ is directly derived from the QP value.

As the distortion of these two residual coding approaches both depends on the scalar quantization step, which is the same during the encoding phase, we can safely

assume that,

$$D_{vq} \simeq D \tag{3.18}$$

By applying this into the Equation (3.17), we obtained the new criterion:

$$\lambda \cdot R(\mathbf{r}_{vq}) + \lambda \cdot R_{vq} < \lambda \cdot R(\mathbf{r}) \tag{3.19}$$

In the conventional residual coding approach, the intra prediction residual vector $\mathbf{r}$ is sent directly to the following steps: DCT-based transform, scalar quantization and entropy coding. In the MDVQ approach, it is the final residual of vector quantization $\mathbf{r}_{vq}$ which is sent to the same steps. To determine the number of bits required by the two residual coding approaches is actually to compute the number of bits for coding the scalar quantized symbols of the transformed coefficient levels.

Let $y[i]$ denote the samples of transform coefficient levels. The probability density function of $y[i]$ is denoted by $p_y(x)$. Supposing that the scalar quantization consists of M quantization levels, and let $l_j, j = 1, ..., M$ denote an output symbol of the quantization level associated with the interval $[t_{j-1}, t_j]$. The probability of an input $y[i]$ being quantized to l_j, in other words, the probability that $y[i]$ is in the interval $[t_{j-1}, t_j]$ can be expressed as,

$$\begin{aligned} p_j &= P\{\hat{y}[i] = l_j\} = P\{t_{j-1} < y[i] \leq t_j\} \\ &= \int_{t_{j-1}}^{t_j} p_y(x)\mathrm{d}x, \ j = 1, ..., M \end{aligned} \tag{3.20}$$

Thus, the entropy of the quantized samples $\hat{y}[i]$ is determined by,

$$\begin{aligned} H(\hat{y}[i]) &= \sum_{i=1}^{M} p_j \log_2(p_j) \\ &= \sum_{i=1}^{M} \int_{t_{j-1}}^{t_j} p_y(x)\mathrm{d}x \log_2(\int_{t_{j-1}}^{t_j} p_y(x)\mathrm{d}x) \end{aligned} \tag{3.21}$$

The samples of scalar quantizer input value $y[i]$ is assumed to be statistically inde-

pendent, to have a zero mean, a variance σ_y^2, and a smooth probability density function. Let us denote by Δ a uniform step size for the scalar quantizer, thus,

$$\Delta \ll \sigma_y \tag{3.22}$$

The number of quantization levels M is large, which means that,

$$M\Delta \gg \sigma_y \tag{3.23}$$

This justifies the approximation $M = \infty$, which is used in what follows. According to the source coding theory presented in [VB93], the minimum bit rate needed to transmit a quantized signal is determined by the entropy $H(\hat{y}[i])$ of the quantizer-output symbols. If the probability density function $p_y(x)$ is sufficiently smooth, then,

$$P\{\hat{y}[i] = l_j\} = \int_{t_{j-1}}^{t_j} p_y(x)\mathrm{d}x \simeq \Delta \cdot p_y(l_j) \tag{3.24}$$

We obtain an approximation of the bit rate for coding the transformed coefficient levels l_{vq}:

$$\begin{aligned} H(\hat{l}_{vq}) &= -\sum_{j=1}^{M} p_j \log_2(p_j) \\ &\simeq -\sum_{j=1}^{M} \Delta \cdot p_y(l_j) \log_2(\Delta \cdot p_y(l_j)) \\ &= -\sum_{j=1}^{M} \Delta \cdot p_y(l_j) \log_2(p_y(l_j)) - \log_2(\Delta) \sum_{j=1}^{M} \Delta \cdot p_y(l_j) \\ &\simeq -\int_{-\infty}^{+\infty} p_y(u) \log_2(p_y(u))\mathrm{d}u - \log_2(\Delta) \end{aligned} \tag{3.25}$$

The samples of the transform coefficient levels of original intra prediction residue have the same statistical characteristics with the probability density function denoted by $q_y(x)$. As the same scalar quantizer has been used for quantizing l, the condition

for applying MDVQ residual coding in Equation (3.17) can be transformed to:

$$\begin{aligned} \lambda \cdot (-\int_{-\infty}^{+\infty} p_y(u) \log_2(p_y(u)) \mathrm{d}u - \log_2(\Delta) + R_{vq}) \\ < \lambda \cdot (-\int_{-\infty}^{+\infty} q_y(u) \log_2(q_y(u)) \mathrm{d}u - \log_2(\Delta)) \end{aligned} \tag{3.26}$$

$$\int_{-\infty}^{+\infty} p_y(u) \log_2(p_y(u)) \mathrm{d}u + R_{vq} < \int_{-\infty}^{+\infty} q_y(u) \log_2(q_y(u)) \mathrm{d}u \tag{3.27}$$

with R_{vq} is a constant specifying the bits of coding MDVQ side information.

We can conclude that the decision of apply MDVQ based residual coding does not depend on the quantization parameter. This means that, whatever the QP value used to generate the residuals of the four codebook training sequences, the codebook construction procedure discussed in Section 3.2.4 will select independently the actual training vectors using Equation (3.17)for the QP value. This result suggests us to construct a group of mode dependent codebooks which is independent of the QP value. As a consequence, a training sequence consisting of residual vectors obtained with different QP values is used for constructing the QP-independent MDVQ codebooks. The experimental results have confirmed that the same coding performance can be achieved using the generic codebook compared with the groups of codebooks learned separately for the different QP values. This property is very useful since it helps reducing storage requirements, and simplifies the design of the codec: it is not necessary to scale the code vectors to match a certain rate-distortion compromise.

3.3 MDVQ-based residual coding mode signaling

In the HEVC standard, Transform Unit (TU) blocks are organized in a quad-tree structure which allows the splitting of a Coding Unit (CU) block into TU blocks with sizes from 4×4 to 32×32. Each intra predicted TU block contains a vector of residual samples which can be further compressed with MDVQ based residual coding. In our experiments, the proposed approach is applied for coding TU blocks of size 4×4 and

8×8. Obviously, larger TU blocks for example of size 16×16 and 32×32 can make use of the VQ residual coding as well. However, this option is eliminated in current version of implementation due to the increase in storage requirements and encoding complexity.

As described in Section 3.2.1, a TU block will be encoded either with MDVQ based residual coding activated or using the regular HEVC residual coding without MDVQ. If the vector quantization performs better in a rate distortion sense than the ordinary residual coding of the HEVC standard, the MDVQ based residual coding will be used for coding this TU block. In this case, side information indicating the use of vector quantization of this block and specifying the index of the best matching code vector needs to be transmitted to the receiver. Thus, these two informations are subject to the coding mode signaling design with an objective of minimizing the bit rate consumption. Firstly, for the purpose of coding the vector quantization flag efficiently, the distribution of TU block of an image has been taken into account. Secondly, since it is impossible to have the information about the probability of code vector utilization, we assumed that it is relatively uniform. In that case, a fixed length code is used for coding the index of code vector.

In reality, 4×4 TU blocks are much more frequently represented in 8×8 CUs than in 16×16 CU blocks, so we limit the residual coding to 4×4 TU blocks which reside inside a 8×8 CU block. For an 8×8 CU block which is split into four 4×4 TU blocks, we use three syntax elements and a tree structure for the residual coding mode signalization. The syntax element $res\_vq\_cu\_flag$ indicates whether at least one of the four TU blocks inside the 8×8 CU block uses MDVQ. This flag is coded using one bit and an associated context-adaptive binary arithmetic coding (CABAC) context. When $res\_vq\_cu\_flag$ is set to 1 , for each 4×4 TU block a syntax element $res\_vq\_tu\_flag$ indicates whether MDVQ is used. This syntax element is also coded using one bit and an associated CABAC context. When MDVQ is used on a TU, a fixed-length syntax element res_vq_idx is transmitted, representing the index of the codeword in the codebook associated with the intra prediction mode and TU size.

As the distribution of 8×8 TU blocks in different sizes of CU block is relatively uniform, only two syntax elements $res\_vq\_tu\_flag$ and res_vq_idx are signalling the use of MDVQ on 8×8 TU blocks. Except for the CU flag parsing step for 4×4 TU blocks, the decoding steps of syntax elements are shared between TU 4×4 blocks and TU 8×8 blocks.

3.4 Experimental results

The scheme of MDVQ-based residual coding has been implemented in the reference software HM8.0 of HEVC standard. In order to investigate the performance of various test cases, the following test conditions were used. In addition to the configurations defined by MDVQ, all tests were performed by setting different configurations in HM 8.0 reference software with the JCT-VC common test conditions [Bos08]. Test conditions are summarized as follows.

- HEVC main profile was used.
- Four quantization parameters were used: 22, 27, 32 and 37.
- All intra configurations were used.
- Bjøntegaard delta bit rate is computed using piece-wise cubic interpolation [Bjø08].
- 35 QP-independent codebooks corresponding to 35 intra prediction modes.
- A total of 20 video sequences in six classes were evaluated. These sequences cover a wide range of resolutions shown in Table 3.1 and image patterns. Class E and class F contain video conferencing sequences and screen content sequences respectively.
- When splitting a CU block into TU blocks for coding intra prediction residue, the allowed size of the TU blocks range from 4×4 to 32×32 with two constraints: First, the size of the TU blocks should not be larger than the CU block that the

TU blocks reside inside. Second, when splitting a TU block into smaller sub-TU blocks, the maximum of the splitting depth is 3. Table 3.2 shows the possible block partition for a CU block.

Class Type	**Image Resolution**
class A	2560×1600
class B	1920×1080
class C	832×480
class D	416×240
class E	1280×720
class F	1024×768

Table 3.1: Resolutions of test sequences.

CU size	**TU size**
64×64	32×32, 16×16, 8×8
32×32	32×32, 16×16, 8×8
16×16	16×16, 8×8
8×8	8×8, 4×4

Table 3.2: Possibility of partitioning a CU block into TU blocks.

As discussed in Section 3.3, each intra predicted TU block contains a vector of residual samples which can be further compressed with MDVQ based residual coding. However, due to the notable computation complexity and the considerable codebook storage requirements for quantizing larger dimension residual vectors, the proposed approach of residual coding is activated only for 4×4 TU blocks and 8×8 TU blocks. For each of them, 35 QP-independent codebooks corresponding to 35 intra prediction modes have been derived.

In our experiments, the influence of size of the 35 codebooks on the performance of residual coding has been investigated. Codebooks of size from 16 to 512 are constructed with the k-means algorithm by using the same training vector sequences. The experimental results show that codebooks of larger size provide a better bit rate saving than codebooks of smaller size. However, the bit rate saving is quite stable when the size of the codebooks is larger than 256. Besides, the vector quantization complexity

increases with the size of the codebooks. For this reason, for all simulations presented in this chapter, each codebook is limited to containing 256 codewords. The simulation results shown in Section 3.4.2 and Section 3.4.3 demonstrate that these generic QP independent codebooks have a relatively small dimension, but can provide a good compromise between storage, complexity and performance.

3.4.1 Simulation A: Special codebook constructed by k-means algorithm

In order to evaluate the potential of the learning method in the ideal case where the codebook is well suited to the input sequence, we have first constructed the codebook using training vectors from the same sequence. Residual vectors are extracted from the same sequence as the measure for compression performance evaluation. In other words, residuals extracted from a test sequence are collected into a training sequence of residual vectors, and then this training sequence is used for constructing codebooks that are used for coding the test sequence. This is of course unrealistic, but the purpose of this approach is to evaluate an upper bound on the performance that could be achieved when the codebook is fully adapted to the sequence to be encoded. Table 3.3 shows the performance of the proposed MDVQ based residual coding using this approach on class B sequences. In this ideal but unrealistic case, one can observe that MDVQ can provide average bit rate saving of 4.9% on class B sequences.

sequence	**bit-rate saving (%)**
Kimono1_1920x1080p	−0.25
ParkScene_1920x1080p	−3.19
Cactus_1920x1080p	−13.66
BQTerrace_1920x1080p	−1.99
BasketballDrive_1920x1080p	−5.16
Average of class B	−4.95

Table 3.3: Bit rate savings of MDVQ based residual coding in using sequence-dependent codebooks under All-Intra coding mode.

3.4.2 Simulation B: Generic codebook constructed by k-means algorithm

In reality, a generic codebook is always needed because it is impossible to transmit a specific codebook for each sequence. So we studied the possibility of using general quantization codebooks for all types of video sequences. A way of constructing generic codebooks is to use training vectors which have no relation to the test sequences. We selected a set of sequences containing natural images from a variety of sources:

sequence	resolution	number of frame
traffic	4096×2048	6
bonnefranquette	1920×1080	25
manege	1920×1080	25
samsaraCoaligned	1920×1080	25
sedofCropped	1920×1080	25
samoaDeinterlaced	1280×720	50

Table 3.4: External sequences for constructing generic codebooks.

These sequences are encoded with the HEVC normal encoding scheme. Those intra prediction residual vectors, which satisfy the condition of rate distortion optimization discussed in Section 3.2.2, are selected as training vectors. Then the training vectors of all sequences are mixed together to form a training sequence for codebook construction with k-means algorithm. Table 3.5 shows the simulation results of coding test sequences with the MDVQ approach. An average bitrate saving of 1.1% can be observed in this realistic use case. Appendix A gives some examples of parts of decoded residual images using the MDVQ coding scheme and the conventional HEVC coding scheme respectively. By using MDVQ, less energy is remained in the residuals, which means that MDVQ exploits the correlation in the residual domain efficiently.

Interestingly, a larger gain is observed for low-resolution sequences which are usually more difficult to compress, this being attributed to the larger proportion of 4×4 TU blocks and 8×8 TU blocks. The method also performs well on video conference (class E) and screen content (class F) sequences. Comparing with the higher performance shown in Table 3.3 where the simulation uses codebooks that are particularly adapted

to each test sequence, we conclude that improved gains may be obtained with a realistic adaptive codebook approach.

3.4.3 Simulation C: Generic codebook constructed by Kohonen algorithm

In order to investigate the impact of codebook construction algorithm on the vector quantizer design, we compared the coding performances using the codebooks constructed with the k-means algorithm and that using the codebooks constructed with the Kohonen algorithm. The same training sequences are used for building generic codebooks with the Kohonen algorithm. Let us suppose that the j^{th} code vector in the list $\mathbf{c}_j$ is selected for quantizing the source vector, thus the neighboring code vectors for the selected code vector is determined by,

$$N(\mathbf{c}_j(t), \mathbf{c}_i(t)) = \begin{cases} 1 & \text{if } |\mathrm{j} - \mathrm{i}| < \mathrm{D} \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

where D controls the maximum number of neighboring code vectors that are modified when updating the selected code vector. The multiplier factor $\theta(\mathbf{c}_i(t))$ is also simplified into,

$$\theta(\mathbf{c}_i(t)) = A_2 \cdot (1 - |j - i| \cdot D^{-1}) \tag{3.29}$$

where the constant A_2 controls the maximum modification of a code vector. The constants D and A_2 as well as the constant A_1 of multiplier factor $\alpha(t)$ in Equation (3.12) are determined experimentally in order to minimize the distortion of quantizing the training sequence.

Then we used the Kohonen codebooks for MDVQ-based residual coding. From the experimental results shown in Table 3.6, one can observe an average bit rate saving of 0.8%. Even though the Kohonen codebooks is less efficient compared with the k-means codebooks in MDVQ residual coding, it has an advantage that the Kohonen codebooks can be continuously updated by new additional training vectors. This means

that the Kohonen codebooks can even be updated by new training vectors during the encoding of a test sequence. This feature is very important as it shows the possibility of construction some kind of adaptive codebooks.

3.5 Conclusion

This chapter represents a new approach for coding the intra prediction residue in an HEVC based encoder. The MDVQ based residual coding can also be seen as a second-order prediction which aims at further removing the remaining correlation in residuals of first-order intra prediction. The method is based on vector quantization with a codebook tailored for each intra prediction mode. Each codebook is hence learned considering a set of residual signals obtained when using a specific intra prediction mode. This training set of residual signals is further limited to the blocks for which the VQ will bring improved RD performance when compared to the reference HEVC encoding. These mode dependent codebooks do not only improve the accuracy of the residual prediction by directly exploiting the correlation in residual domain, but also reduce the computation complexity of the best matching code vector search in a super-size codebook. Another advantage of the proposed quantizer design is that these codebooks are QP independent. The importance of the QP independent codebook has to be particular emphasized because it makes the MDVQ based residual coding being a practical VQ solution which can encode a sequence with an arbitrary QP value.

In comparing the experimental results of the three simulations, we conclude that a significant bit rate saving can be achieved when the codebook is adapted to the sequences to be coded. Moreover, codebook designed with the k-means algorithm provides better coding performances than those designed with the Kohonen algorithm. However, the principle of Kohonen algorithm can be used for designing an adaptive codebook of MDVQ based residual coding. In the next chapter, we present an approach of residual coding using adaptive mode dependent vector quantization where the adaptive codebook are designed with the Kohonen algorithm.

sequence class	**bit-rate saving (%)**
NebutaFestival	−0.09
PeopleOnStreet	−1.62
SteamLocomotiveTrain	0.44
Traffic	−1.51
Average of class A	**-0.7**
BasketballDrive	0.12
BQTerrace	−1.33
Cactus	−1.03
Kimono	0.03
ParkScene	−0.14
Average of class B	**-0.5**
BasketballDrill	−2.27
BQMall	−1.34
ParkScene	−0.95
RaceHorse	−0.52
Average of class C	**-1.3**
BasketballPass	−1.26
BlowingBubbles	−1.24
BQSquare	−0.98
RaceHorse	−1.68
Average of class D	**-1.3**
FourPeople	−1.61
Johnny	−1.67
KristenAndSara	−1.62
Average of class E	**-1.6**
BasketballDrillText	−2.53
ChinaSpeed	−1.29
SlideEditing	−0.91
SlideShow	−1.80
Average of class F	**-1.6**

Table 3.5: Bitrate savings of MDVQ-based residual coding using sequence-independent k-means codebooks under All-Intra coding mode.

sequence class	**bit-rate saving (%)**
NebutaFestival	−0.23
PeopleOnStreet	−1.81
SteamLocomotiveTrain	0.38
Traffic	−1.48
average of class A	**-0.8**
BasketballDrive	0.02
BQTerrace	−0.84
Cactus	−0.79
Kimono	0.14
ParkScene	−0.07
Average of class B	**-0.3**
BasketballDrill	−1.38
BQMall	−0.65
ParkScene	−0.57
RaceHorse	−0.51
Average of class C	**-0.8**
BasketballPass	−0.77
BlowingBubbles	−0.98
BQSquare	−0.38
RaceHorse	−1.22
Average of class D	**-0.8**
FourPeople	−1.32
Johnny	−0.88
KristenAndSara	−0.9
Average of class E	**-1.0**
BasketballDrillText	−1.73
ChinaSpeed	−0.64
SlideEditing	−0.22
SlideShow	−1.54
Average of class F	**-1.0**

Table 3.6: Bitrate savings of MDVQ-based residual coding using sequence-independent Kohonen codebooks under All-Intra coding mode.

Chapter 4

Intra residual coding using Adaptive MDVQ

The MDVQ residual coding approach proposed in the previous chapter has shown its ability of coding intra prediction residue in order to further reduce the redundancy in the residual domain. In MDVQ, the objective of the mode dependent codebooks constructing is not only to model the directional structures in residuals but also to optimize the code vectors in a rate distortion sense. The MDVQ approach achieves an important bit rate saving of coding the test sequences with the generic mode dependent codebooks. However, it is of interest to further investigate the cases where the codebooks are constructed with training vectors extracted from the test sequences themselves. In this unrealistic case, a much higher bit rate saving was obtained. This observation suggests that a better coding efficiency can be achieved if the generic codebooks can somehow be adapted to the test sequence during the encoding stage.

To achieve this, we propose in this chapter an approach of intra residual coding using Adaptive MDVQ (AMDVQ). This approach includes a codebook update algorithm based on the Kohonen Self-Organized Maps. The structure of this chapter is organized as follows. Section 4.1 gives an overview of the AMDVQ residual coding approach. Section 4.2 deals with the key elements for constructing the adaptive codebooks. In Section 4.3, we present the algorithm of code vector update as well as the parameter

optimization of this algorithm. Finally, the experimental results are shown in Section 4.4.

4.1 Adaptive MDVQ residual coding in HEVC

Let us consider the simulation of the MDVQ residual coding in Section 3.4.1. In this unrealistic case, test sequences are encoded with specific codebooks that are constructed with training vectors extracted from the test sequences themselves. Obviously, we can analyze the test sequences and adapt the codebooks to the test sequences before encoding them, but this requires the communication of the codebooks update to the receiver, which is usually very expensive. Thus, in practice, it is impossible to have specific codebooks at the encoder and the decoder for coding and decoding a specific test sequence. However, it is possible to have adaptive codebooks by updating the generic codebooks based on the information collected when encoding the test sequence.

In the MDVQ residual coding scheme, residual vectors are sequentially extracted from an image and are individually quantized by code vectors of the fixed generic codebooks. Usually, the statistical characteristics of successive residual vectors are not independent. Compared with the independent distribution of the residual vectors, the conditional distribution of one residual vector given the observation of some neighboring residual vectors provides much more information. In this respect, it is preferable to construct a vector quantizer which continuously adapts the code vectors based on information about the decoded residual vectors. Often the changes in the statistical characteristics of the residual vectors to be encoded cannot be simply described as a time variation of some easily identifiable features of the vectors such as mean or norm. Therefore, special attention is given to the Kohonen algorithm which uses training vectors to continuously update the code vectors during the codebook construction. By considering the features of residual vectors mentioned above, the approach of Adaptive MDVQ residual coding uses previously reconstructed residual vectors to update the code vectors during the encoding and decoding of a test sequence. The structure of

the adaptive codebooks have been also investigated in order to be able to update code vectors with the Kohonen algorithm.

The Adaptive MDVQ residual coding approach consists of two steps: mode dependent vector quantization and code vector update. The scheme of the Adaptive MDVQ residual coding is depicted in Fig. 4-1.

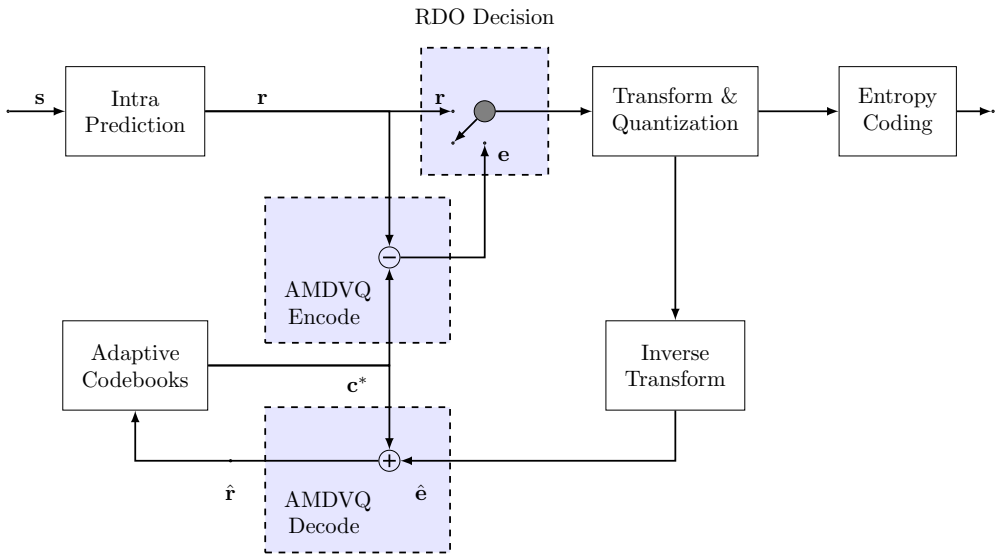


Figure 4-1: The scheme of residue coding using Adaptive Mode Dependent Vector Quantization. The original intra prediction residue is denoted by $\mathbf{r}$, the best matching code vector is denoted by $\mathbf{c}^*$, and the final vector quantization error is denoted by $\mathbf{e}$. The reconstructed first order residual is denoted by $\hat{\mathbf{r}}$, and it will be used for codebook update.

In adaptive MDVQ residual coding, the step of vector quantization is the same as in MDVQ: find the best matching code vector in the mode related codebook and compute the vector quantization error. Then this quantization error is transformed, and the transformed coefficients are then quantized with a scalar quantizer and entropy coded. The MDVQ approach stops here, but the Adaptive MDVQ approach continues with the step of code vector update. Supposing that the residual vector $\mathbf{r}$ is encoded with vector quantization based residual coding mode, in the step of code vector up-

date, the reconstructed residual vector $\hat{\mathbf{r}}$ is used to update certain code vectors of the codebook before encoding the next residual vector. In Chapter 3, we have discussed in detail the criteria of the best code vector and the condition of switching between the ordinary mode and the MDVQ mode for encoding a residual vector. Here, we focus on the subject of adaptive codebook construction, and we will present in the subsequent sections the structure of the adaptive codebooks and how to adapt the code vectors with the reconstructed residual vectors.

4.2 Adaptive codebook construction

4.2.1 Adaptive codebook structure

In Chapter 3, we have presented how to use the Kohonen algorithm to construct mode dependent codebooks. Here, instead of using the Kohonen algorithm only in the codebook construction stage, we use the principle of code vector update of the Kohonen algorithm to adapt the code vectors to the test sequence. In fact, using the Kohonen algorithm for codebook update during residual coding is quite different from using it for codebook construction and special considerations are required for different use cases. In the case of codebook construction, the training vectors are original residual vectors extracted from the training sequences and the codebook construction process can be repeated several times in order to produce optimal codebooks. However, in the case of codebook updates, the residual vectors used for updating the code vectors are reconstructed residual vectors and the number of these residual vectors is limited in a picture of sequence. Thus, the code vectors in the codebooks are updated in finite times when encoding a picture. Consequently, when using the Kohonen algorithm for codebook updates, not only the structure of the codebooks, but also the parameters in the Kohonen algorithm such as the number of code vectors to be updated as well as the amount of the modification of the code vectors need to be reconsidered.

To explain how we use the Kohonen algorithm as a method of codebook update of adaptive vector quantization, we first recall the basic definitions of the Kohonen

algorithm when it is used as a method of codebook construction. At the beginning of the codebook construction with the Kohonen algorithm, code vectors are organized into an unordered list and the samples of each code vector are initialized to random values. Given an input training vector, the first step is to select the best matching code vector for this training vector. Then, neighboring code vectors of the selected code vector are determined by a neighbor function. Finally, the input training vector is used to update not only the best matching code vector, but also those neighboring code vectors. This mechanism ensures that the whole codebook is gradually modified to adapt to the training sequences and that no code vector is omitted during the codebook construction. Indeed, as the training residual vector modifies not only the selected code vector but also its neighboring code vectors, at the end of the codebook construction, a smaller difference in term of Sum of Square Error (SSE) can be observed between two successive code vectors in the list than that between two nonsuccessive code vectors. Thus, at the end of the Kohonen codebook construction process, these code vectors have the following characteristics:

- Code vectors contain adequate sample values to quantize the residual vectors of the test sequence.

- Code vectors are organized into an ordered list so that smaller SSE can be observed between two successive code vectors than that between two nonsuccessive code vectors.

- Code vectors can be continuously updated with new training vectors if the codebook construction process is iterated.

4.2.2 Adaptive codebook construction

In MDVQ-based residual coding approach, either the Kohonen codebooks or the k-means codebooks contain adequate code vectors for quantizing the residual vectors when encoding the test sequence. However, the k-means codebooks do not satisfy

the characteristics of the Kohonen codebooks mentioned above so that the fact of updating the selected code vector as well as the neighboring code vectors may damage the k-means codebooks. As a consequence, if the Kohonen algorithm is used for updating the k-means codebooks, these codebooks need to be reorganized to satisfy the characteristics of the Kohonen codebooks.

We have considered to use the Kohonen codebooks shown in Chapter 3 as the initial codebooks in AMDVQ approach. However, the experimental results indicate that the k-means codebooks provide a better coding performance than the Kohonen codebooks. This means that, by using the same training sequences, the code vectors derived with the k-means algorithm are more suitable for quantizing the residual vectors than the code vectors derived with the Kohonen algorithm. We could definitely improve the coding efficiency when using the Kohonen codebooks by updating the code vectors with the reconstructed residual vectors during the encoding process, but the number of reconstructed residual vectors in a picture is limited. So it is difficult to have suitable codebooks within a limit times of update in a picture. With this constraint, the k-means codebooks are preferred as the initial codebooks but they need to be reorganized so that we can use the Kohonen algorithm for the codebook update.

As in the Kohonen codebooks, the code vectors in the initial k-means codebooks are organized into a list with a constraint on the SSE between two successive code vectors. Fig. 4-2 illustrates a representative example of the structure of the reordered k-means codebook. Noted that the SSE between two code vectors also represents the their Euclidean distance, the reordering of the code vectors in the k-means codebooks can be transformed into finding a shortest path to visit each code vector only one time without returning back to the beginning code vector. This is actually a Traveling Salesman Problem (TSP) without returning back to the departure city. The traditional TSP algorithm aims at searching the shortest cycle of visiting all cities.

To find a solution for the shortest path to visit all code vectors, we can add a virtual code vector in the codebook and use the TSP algorithm to find the shortest cycle of connecting these code vectors including the virtual code vector. Then, we cut the cycle

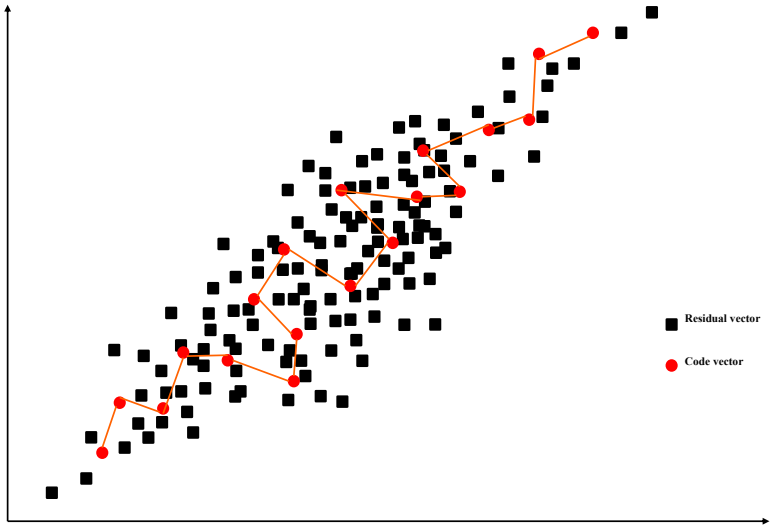


Figure 4-2: The structure of reordered k-means codebook of adaptive MDVQ residual coding.

at this virtual code vector and thus obtain a solution of the shortest path. The two code vectors connected to the virtual code vector are the departure code vector and the terminal code vector respectively. So, to determine the position of adding the virtual code vector in the codebook, we need to identify firstly the departure code vector and the terminal code vector. According to the Kohonen algorithm, code vectors are organized into a list so that two successive code vectors have smaller distance than two non successive code vectors. As a consequence, we conclude that the largest variation can be found between the code vector at the beginning of the list and the one at the end of the list. So we select two code vectors in the codebook with the largest sum of squared error between themselves as the departure code vector and the terminal code vector respectively and connect them to the virtual code vector. The detailed algorithm for constructing the reordered k-means codebooks is as follows:

- **Step 1**: Construct the k-means codebooks with the k-means algorithm.
- **Step 2**: Select two code vectors with the largest sum of squared error between themselves as the departure code vector and the terminal code vector respectively.

- **Step 3**: Add a virtual code vector between the departure code vector and the terminal code vector. Set the distance from this virtual code vector to all other code vectors equal to zero.

- **Step 4**: Find the shortest cycle of visiting all code vectors including the virtual code vector with a TSP algorithm.

- **Step 5**: Cut the cycle at the position of the virtual code vector and eliminate the virtual code vector from the codebook.

- **Step 6**: Order the code vectors in the initial codebook by using the shortest path obtained with the TSP algorithm.

The TSP is a NP-complete problem and several algorithms providing approximate solutions have been discussed in the literature. We used a slightly modified version of the Lin-Kernighan heuristic (LKH) algorithm [LK73] for the part of TSP solution.

4.3 Codebook update using the Kohonen algorithm

In the previous section, we described how to reorganize the k-means codebooks into the adaptive codebooks in order to be able to update the code vectors with the Kohonen algorithm. Here, we present how to use the reconstructed residual vector to adaptively modify the selected code vector and its neighboring code vectors. Before describing the codebook update step in details, we highlight the following definitions of the Kohonen algorithm:

- **The reordered codebooks** refer to the reordered k-means codebook, where code vectors are ordered into a list with respect to the shortest TSP path.

- **The selected code vector** is the best matching code vector for encoding an input residual vector.

- **The neighboring code vectors** are code vectors that will be updated while updating the selected code vector. The neighboring code vectors of the selected code vector are determined by the neighbor function.

As described in Chapter 3, when using the Kohonen algorithm for codebook construction, it can be divided into a learning phase and an adjustment phase. At the beginning of learning phase, the sample of the code vectors are initialized with random values. In order to learn the characteristics of the training vectors quickly, a large number of the neighboring code vectors is involved in the codebook learning phase, and the multiplying factors $\alpha(t)$ and $\theta(\mathbf{c}_i(t))$ in the Equation (3.11) are set to relatively large values in order to quickly modify the code vectors. Then, during the adjustment phase where the code vectors have been self-organized, significant modification of code vectors should be avoided. As a consequence, the number of neighboring code vectors has to be reduced as well as the value of the multiplying factors in this phase.

Here, when using the Kohonen algorithm for codebooks update during the encoding of the test sequence, the conditions for applying the Kohonen algorithm have been changed. First, at the beginning of the encoding of a picture, codebooks of AMDVQ are already in the final state of the codebook construction process so that the codebook update can be viewed as the adjustment phase which continues during the encoding of the picture. As a result, the number of neighboring code vectors and the multiplying factors have to be set to relatively small values to modify the code vectors smoothly. Second, opposite to the codebook construction process which uses a large quantity of training vectors and iterates the learning process several times, the codebook update process has a limited number of reconstructed residual vectors in a picture for modifying code vectors so that the code vector update should be efficient. As the residual characteristics also vary in time, the codebooks have to be continuously modified during the encoding of image and all reconstructed residual vectors should have the same impact on the code vectors. This means that it is preferable to have the values of the parameters such as the number of neighboring code vectors as well as the multiplying factors unchanged.

4.3.1 Neighboring code vectors

Let $\mathbf{c}_j(t)$ denote the selected code vector in the index j of the codebook for quantizing the t^{th} residual block of a picture, and $\mathbf{c}_i(t)$ denote a code vector in the index i of the codebook, the neighbor function $N(\mathbf{c}_j(t), \mathbf{c}_i(t))$ that determines whether this code vector is a neighboring code vector of the selected code vector at present is expressed as,

$$N(\mathbf{c}_j(t), \mathbf{c}_i(t)) = \begin{cases} 1 & \text{if } |\mathrm{j} - \mathrm{i}| < \mathrm{D(t)} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $D(t)$ controls the maximum difference between the index of the code vector $\mathbf{c}_i(t)$ and the index of the selected code vector $\mathbf{c}_j(t)$ in the codebook. If the value of the neighbor function is 1, this code vector is a neighboring code vector of the selected code vector and it will be updated with the reconstructed residual block as well. If not, this code vector is not a neighboring code vector and it will not be concerned by the codebook update step.

When the Kohonen algorithm is used for codebook construction, the number of neighboring code vectors reduces during the codebook training process. Here, we use the Kohonen algorithm for codebook update, the number of neighboring code vectors to be updated should not be changed, and each reconstructed residual vector has the same impact on the whole codebook. As a result, a constant D is used for specifying the maximum difference between the index of the neighboring code vector and the selected code vector. Thus, the neighbor function defined in Equation (4.1) is refined into:

$$N(\mathbf{c}_j(t), \mathbf{c}_i(t)) = \begin{cases} 1 & \text{if } |\mathrm{j} - \mathrm{i}| < \mathrm{D},\ D > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

4.3.2 Multiplying factors

Two multiplying factors defined in Equation (3.11) control the amount of modification of the code vectors. The value of the multiplying factor $\alpha(t)$ decreases when applying the Kohonen algorithm to the codebook construction process. But in codebook update

process, it can be expressed as,

$$\alpha(t) = A_1,\ 0 < A_1 < 1 \tag{4.3}$$

Another multiplying factor $\theta(\mathbf{c}_i(t))$ used to control the amount of modification of each neighboring code vector is defined as,

$$\theta(\mathbf{c}_i(t)) = 1 - \frac{|j-i|}{D} \tag{4.4}$$

The reconstructed residual vector has a larger impact on the selected code vector than on the neighboring code vectors, and the amount of the modification decreases with the difference of index between the selected code vector and the neighboring code vector. Fig. 4-3 shows a representative example of this multiplying factor. As shown in this example, the closer the neighboring code vector to the selected code vector, the larger it will be modified with the reconstructed residual vector.

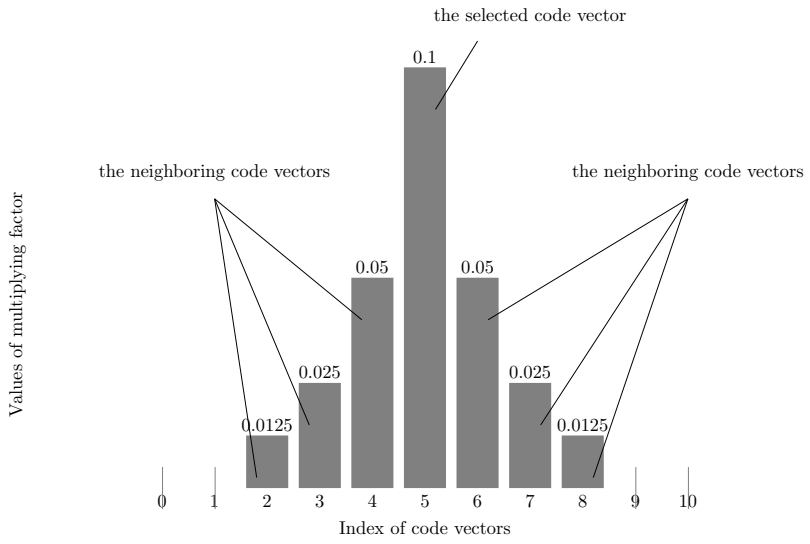


Figure 4-3: An representative example of the value of the multiplying factor $\theta(\mathbf{c}_i(t))$. The index of the selected code vector is 5. The maximum index difference of the neighboring code vector and the selected code vector is 3. The multiplying factor of each code vector to be updated decreases with the index difference.

4.3.3 Code vector update

Let $\mathbf{r}(t)$ denote the residual vector encoded by a code vector $\mathbf{c}_i(t)$, and $\hat{\mathbf{r}}(t)$ denotes the reconstructed residual vector. For the selected code vector $\mathbf{c}_i(t)$ and its neighboring code vectors, the update step defined in Equation (3.11) is changed into:

$$\mathbf{c}_i(t+1) = \mathbf{c}_i(t) + \alpha(t) \cdot \theta(\mathbf{c}_i(t)) \cdot (\hat{\mathbf{r}}(t) - \mathbf{c}_i(t)) \tag{4.5}$$

In Section 4.4.1, we describe how to determine the adequate value of the parameters D and A_1 in a heuristic manner.

4.3.4 Implementation issues

In HEVC, coding a CTU block is divided into two stages: estimation and encoding. In the estimation stage, the encoder splitting the CTU block recursively in to 4 sub-block each of which is a smaller CU block. At each recursive level, the encoder estimates the best coding mode for each CU block, including the how to of splitting a CU into PUs, the best intra prediction mode of each PU block, the best way of splitting a CU into TUs as well as the residue coding mode of each TU block. All this coding options will be stored during the estimation stage. Then, in the encoding stage, with these coding options, the encoder conduct the encoding of the each CU in the CTU.

Obviously, during the coding mode estimation, at the end of coding a TU block with the AMDVQ mode, we can update the relative code vectors with the reconstructed residues in this TU block and a modified codebooks will be used for coding the next TU block. However, when a coding mode is not validated by the encoder, all code vectors updated when coding TUs with this mode need to be recovered to its previous values. Indeed, it requires a complex mechanism to keep the codebook update in line with the actual coding option of each TU block during the coding mode estimation.

To resolve this problem, our AMDVQ residual coding has some restrictions on codebook updates. During the estimation stage of a CU, the encoder determines for each TU block of this CU the better coding mode between the AMDVQ mode and the

conventional residual coding mode. But the codebook update is not performed in this stage, so that the codebook is unchanged. Then, in the encoding stage of this CU, each TU block are coded with the decision made before, and all relative code vectors of TUs coded with AMDVQ, including the selected code vectors and the neighboring code vectors will be updated sequentially according to the TU processing order.

4.4 Experimental results

In HEVC standard, each intra predicted TU block contains a vector of residual samples which can be encoded with Adaptive MDVQ. In our experiments, as in MDVQ residual coding approach, the proposed AMDVQ was used for coding TU blocks of size 8×8 and TU blocks of size 4×4 which reside inside a 8×8 CU block. The residual coding mode signaling of adaptive MDVQ is the same as that of MDVQ presented in Chapter 3.

The Adaptive MDVQ residual coding has been implemented in the reference software of HEVC standard HM8.0. In addition to the test configurations defined in MDVQ, another configuration of the codebook update process is needed to be emphasized because it differs in the encoding mode. When the test sequence is encoded in All-Intra (AI) mode, the codebook update is performed within a frame (picture) of a sequence and codebooks are reinitialized at the beginning of coding each frame. When the test sequence is encoded with Random Access (RA) mode, the codebooks do not need to be initialized and can be updated continuously within a Group Of Pictures (GOP).

4.4.1 Adaptive codebook parameter optimization

The codebook update parameters optimization is more important for the All-Intra mode than for the Random-Access mode. In All-Intra mode, a Group Of Pictures of the sequence contains only Intra coded frames, and each frame is encoded independently. This means that the code vectors update step stops after coding the last residual

block in a frame, and the reordered k-means codebooks have to be reinitialized at the beginning of encoding the next frame. However, in Random-Access mode, there are not only I frames, but also B frames and P frames which contain inter prediction residual blocks. The B frames or P frames are encoded with reference to previously encoded frames. Thus, for the intra prediction residual blocks in these frames encoded with the AMDVQ residual coding, the code vector update step can be continued in a Group Of Picture (GOP). As a result, in All-Intra mode, the parameters in Equation (4.5) have to be optimized so that the code vectors are adapted to the test sequence with a limited number of updates, whereas in Random-Access mode, there is a sufficient quantity of reconstructed residual vectors in a GOP for updating the code vectors.

A. The number of neighboring code vectors

We first examine the impact of the number of neighboring code vectors. The multiplying factors are kept to the optimum values in the codebook construction process described in Section 3.4.3. It should be noted that the values of the multiplying factors in the codebook update process refer to the values in the adjustment stage of codebook construction process. We tested in our simulations the values of 1, 7, and 16 neighboring code vectors. Larger values are out of consideration due to the increase in the computation complexity. Fig. 4-4 shows the influence of these parameters in the coding performance.

The experimental results show that, for natural sequences, the number of neighboring code vectors to be updated is preferred to be a small value, whereas for sequences of screen content, a larger number of neighboring code vectors is more suitable. In fact, the reordered k-means codebooks used in our simulation are constructed with training sequences which are natural sequences as well, so they are more suitable for encoding natural sequences than for encoding the sequence of screen content. As a result, the number of neighboring code vectors involved in the code vector update differs with the type of sequence. For sequence of screen content, enlarging the scope of the impact of a reconstructed residual vector on the codebook can accelerate the codebook adaptation.

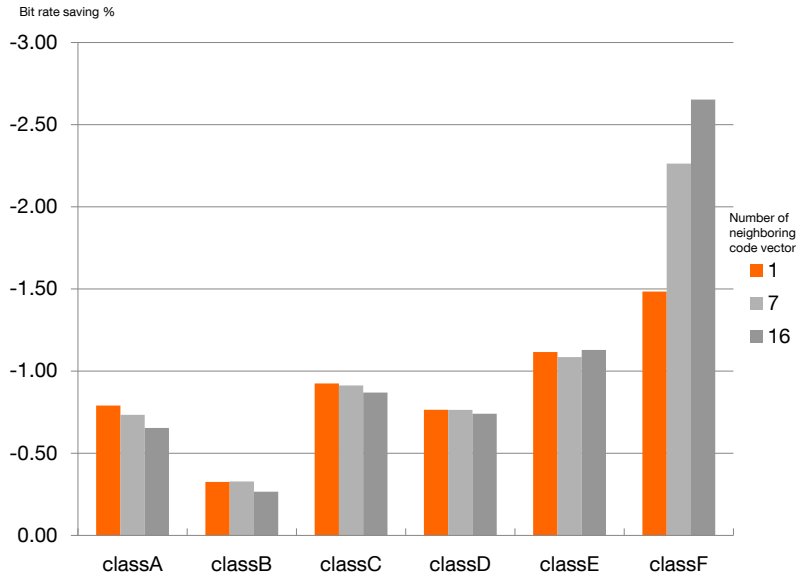


Figure 4-4: The impact of parameter Neighboring Code Vector of coding test sequences of six classes using reordered k-means codebook. Here the learning rate it set to 0.1.

In other words, when the codebook has characteristics that differ from those of the test sequences, a larger number of neighboring code vectors in the codebook update step is preferred.

B. The value of multiplying factors

Then we examine the impact of the multiplying factors on the encoding performance. As the reordered k-means codebooks are more suitable for encoding natural sequences, we limit our experiments to the natural sequences of class A, B, C and D. We first checked the best value of multiplying factor when the number of neighboring code vectors is 1 and 16 respectively. Fig. 4-5 and Fig. 4-6 illustrate how the bit rate saving changes when the value of the multiplying factor increases from 0.1 to 0.9. The experimental results show that the best performance is obtained with a multiplying factor of 0.1 and the number of neighboring code vectors set to 1.

To further investigate the impact of the multiplying factor on the bit rate saving, we tested a sequence of value going from 0.01 to 0.1 for the multiplying factor with the number of neighboring code vectors set to 1. The experimental results are shown

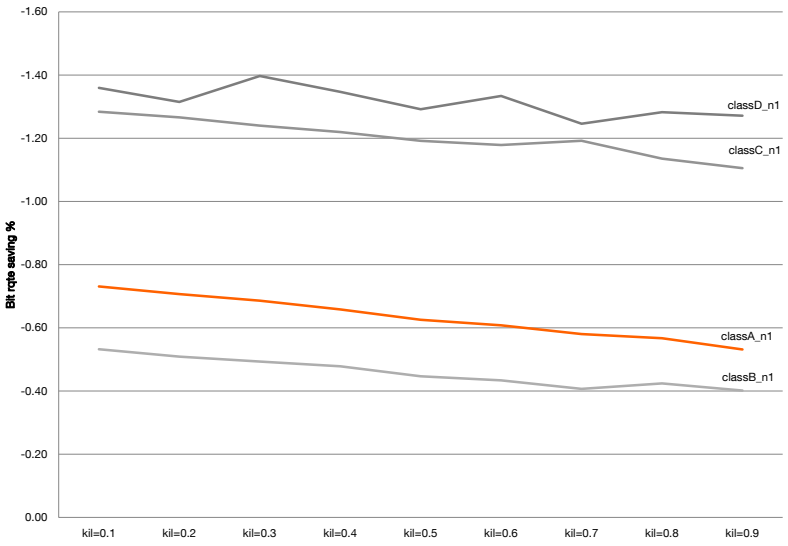


Figure 4-5: The impact of the multiplying factor on coding performance with test sequences of class A, B, C and D. AMDVQ residual coding using reordered k-means codebooks. The number of neighboring code vectors is set to 1. The multiplying factor vary from 0.1 to 0.9.

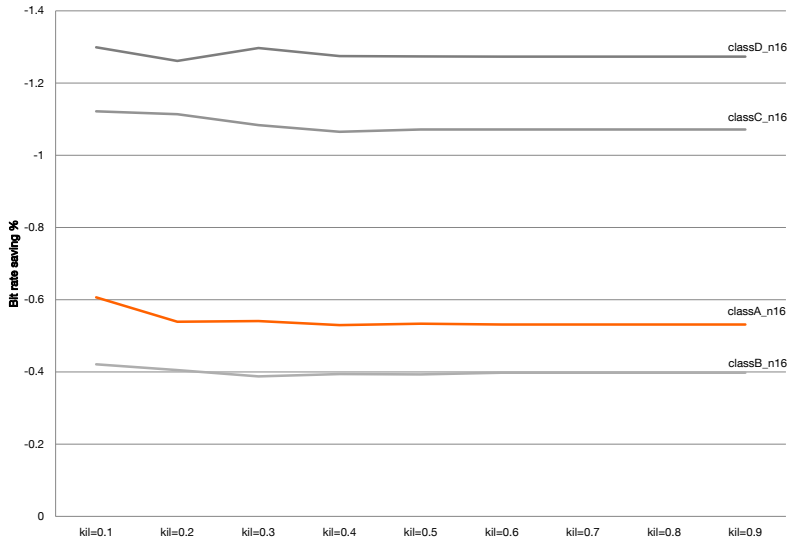


Figure 4-6: The impact of the multiplying factor on coding performance with test sequences of class A, B, C and D. AMDVQ residual coding using reordered k-means codebooks. The number of neighboring code vectors is set to 16. The multiplying factor vary from 0.1 to 0.9.

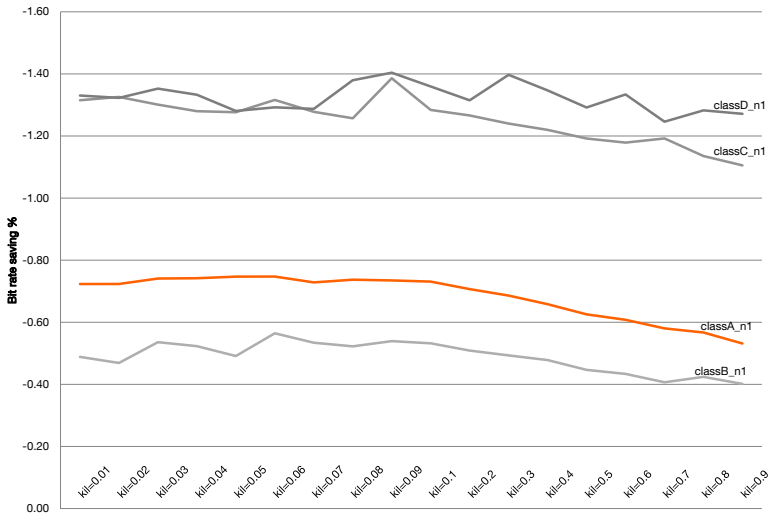


Figure 4-7: The impact of the multiplying factor on coding performance with test sequences of class A, B, C and D. AMDVQ residual coding using reordered k-means codebooks. The number of neighboring code vectors is set to 1. The multiplying factor vary from 0.01 to 0.9.

in Fig. 4-7. We found that a smaller multiplying factor provides higher bit rate saving for sequences of larger resolution, whereas a larger value of the multiplying factor is preferred for sequences in smaller resolution.

4.4.2 Performance of Adaptive MDVQ

In order to evaluate the gain contributed by the adaptive codebooks, the simulations with adaptive codebooks and fixed codebooks are both examined. The simulations are performed with optimized parameters. The number of neighboring code vectors is set to 1 for natural sequences of class A, B, C and D, and this parameter is set to 16 for classes E and F which contain sequences of video conference and sequences of screen content. The value of the multiplying factor is set to 0.06 for sequences of class A and B, and it is set to 0.09 for sequences of class C and D or sequences of screen content, this parameter is set to 0.1. It should be noted that the multiplying factor is determined by the resolution of the video sequence. As such, it is not really adapted

to the class of video sequences.

The performance of adaptive MDVQ residual coding in All Intra mode using re-ordered k-means codebooks is shown in Table 4.1. By comparing the bit rate saving between the approach of AMDVQ and the approach of MDVQ, we concluded that the coding efficiency of vector quantization has been improved by adapting the code vectors to test sequences.

The performance of adaptive MDVQ residual coding in All Intra mode using the Kohonen codebooks is shown in Table 4.2. In MDVQ residual coding, the Kohonen codebooks are less efficient than the k-means codebooks, but the codebook update process can adapt the code vectors to the test sequence in order to enhance the bit rate saving. This also illustrates the fact that the adaptivity of codebook during the vector quantization can improve the coding efficiency, and this adaptivity can be achieved with the Kohonen algorithm.

Furthermore, Table 4.3 shows the performance of AMDVQ residual coding in Random Access mode using Kmeans and Kohonen codebooks respectively. In Random Access mode, only the intra prediction residue is further compressed with AMDVQ approach, and the codebooks are updated continuously with the intra prediction residue actually coded with AMDVQ in a GOP. From the experimental results show that the video coding efficiency has been significantly improved by only coding the part of intra prediction residue with AMDVQ. We conclude that, the coding efficiency can be further improved by extending the AMDVQ residual coding approach to inter prediction residue as well.

4.4.3 Complexity aspect

Both of the proposed MDVQ and AMDVQ residual coding require that the codebooks are stored at the encoder and decoder. In general, vector quantization tends to impose a much larger complexity burden on the encoder because of the search for the best code vector than it does on the decoder where a simple table look-up is needed. Therefore, it can be viewed as a tool that the encoder can optionally choose to use to further improve

Sequence	**Bit rate saving (%) MDVQ**	**Bit rate saving (%) AMDVQ**
NebutaFestival	−0.07	−0.17
PeopleOnStreet	−1.66	−1.74
SteamLocomotiveTrain	0.39	0.35
Traffic	−1.40	−1.43
Average of class A	**-0.7**	**-0.8**
BasketballDrive	0.08	−0.11
BQTerrace	−1.17	−1.28
Cactus	−0.98	−1.02
Kimono	0.01	−0.05
ParkScene	−0.26	−0.36
Average of class B	**-0.5**	**-0.6**
BasketballDrill	−2.37	−2.57
BQMall	−1.25	−1.32
ParkScene	−0.93	−0.98
RaceHorse	−0.79	−0.67
Average of class C	**-1.3**	**-1.4**
BasketballPass	−1.39	−1.58
BlowingBubbles	−1.25	−1.34
BQSquare	−1.00	−1.00
RaceHorse	−1.53	−1.70
Average of class D	**-1.3**	**-1.4**
FourPeople	−1.45	−1.32
Johnny	−1.61	−1.54
KristenAndSara	−1.57	−1.54
Average of class E	**-1.5**	**-1.5**
BasketballDrillText	−2.37	−2.17
ChinaSpeed	−1.31	−1.91
SlideEditing	−0.98	−5.88
SlideShow	−2.55	−4.47
Average of class F	**-1.8**	**-3.6**

Table 4.1: Experiments of coding sequence in All Intra mode with AMDVQ approach. Bit rate savings using sequence-independent **reordered** k-means codebooks. Codebook update parameters are optimized.

Sequence	**Bit rate saving (%) MDVQ**	**Bit rate saving (%) AMDVQ**
NebutaFestival	−0.23	−0.32
PeopleOnStreet	−1.81	−1.52
SteamLocomotiveTrain	0.38	0.21
Traffic	−1.48	−0.99
Average of class A	**-0.8**	**-0.7**
BasketballDrive	0.02	0.00
BQTerrace	−0.84	−0.62
Cactus	−0.79	−0.70
Kimono	0.14	0.13
ParkScene	−0.07	−0.14
Average of class B	**-0.3**	**-0.3**
BasketballDrill	−1.38	−1.62
BQMall	−0.65	−0.67
ParkScene	−0.57	−0.59
RaceHorse	−0.51	−0.59
Average of class C	**-0.8**	**-0.9**
BasketballPass	−0.77	−0.57
BlowingBubbles	−0.98	−0.81
BQSquare	−0.38	−0.46
RaceHorse	−1.22	−1.12
Average of class D	**-0.8**	**-0.7**
FourPeople	−1.32	−1.32
Johnny	−0.88	−1.07
KristenAndSara	−0.90	−0.99
Average of class E	**-1.0**	**-1.1**
BasketballDrillText	−1.73	−1.61
ChinaSpeed	−0.64	−1.13
SlideEditing	−0.22	−4.12
SlideShow	−1.54	−3.74
Average of class F	**-1.0**	**-2.7**

Table 4.2: Experiments of coding sequence in All Intra mode with AMDVQ approach. Bit rate savings using sequence-independent Kohonen codebooks. Codebook update parameters have identical values for all sequences.

Sequence	**Bit rate saving (%) Adaptive Kmeans CB**	**Bit rate saving (%) Adaptive Kohonen CB**
NebutaFestival	0.01	−0.01
PeopleOnStreet	−0.11	−0.14
SteamLocomotiveTrain	0.08	0.23
Traffic	−0.50	−0.62
Average of class A	**-0.1**	**-0.1**
BasketballDrive	0.35	0.32
BQTerrace	−0.84	−0.70
Cactus	−0.79	−0.75
Kimono	0.08	−0.07
ParkScene	−0.15	−0.07
Average of class B	**-0.3**	**-0.3**
BasketballDrill	−1.55	−1.30
BQMall	−0.90	−0.70
ParkScene	−0.73	−0.55
RaceHorse	0.12	0.15
Average of class C	**-0.8**	**-0.6**
BasketballPass	−0.63	−0.38
BlowingBubbles	−0.60	−0.25
BQSquare	−1.08	−0.57
RaceHorse	−0.17	−0.14
Average of class D	**-0.6**	**-0.3**
FourPeople	−3.14	−3.13
Johnny	−2.23	−1.89
KristenAndSara	−3.37	−3.20
Average of class E	**-2.9**	**-2.7**
BasketballDrillText	−1.79	−1.63
ChinaSpeed	−2.34	−2.06
SlideEditing	−11.06	−9.96
SlideShow	−4.19	−5.02
Average of class F	**-4.9**	**-4.7**

Table 4.3: Experiments of coding sequence in Random Access mode with AMDVQ approach. Bit rate savings using sequence-independent Kmeans (left column) and Kohonen (right column) codebooks. Codebook update parameters have identical values for all sequences.

the compression, at the expense of some complexity, while the decoder can always easily support it. On the decoder side, the proposed vector quantization approach introduces simple additional steps: retrieving, adding the code vector and codebook updated when using adaptive MDVQ. Overall, the execution time of decoding increases of only 3% on average for MDVQ approach compared with the reference software HM8.0. Due to the codebook update step, the execution time of decoding increases of 25% for AMDVQ method. We have not investigated the complexity on the encoder side, since the encoding can be accelerated by using fast vector quantization methods such as Tree-Structured VQ [GG92] or other encoder-side optimization. For instance, to reduce the complexity, the use of MDVQ and AMDVQ can be limited to the most frequent intra prediction modes.

4.5 Conclusion

This chapter presents a novel approach of residual coding, the AMDVQ, which further enhances the intra coding performance compared with the MDVQ approach. This is achieved by employing the Kohonen algorithm for codebooks update during the coding of the test sequences. The Kohonen algorithm gradually modifies the code vectors by the reconstructed residual vectors which are encoded by the AMDVQ approach. In this way, the codebooks are adapted to the intra prediction residue of coding test sequences and a higher bit rate saving is obtained. We have also investigated the impact of the initial codebooks on the coding efficiency of AMDVQ approach. The codebooks constructed with the k-means algorithm and the codebooks constructed with the Kohonen algorithm are used as initial codebooks respectively. What we emphasized is that, different from the Kohonen codebooks which is used directly as the initial codebooks in AMDVQ, the k-means codebooks have to be reordered for updating the code vectors by the Kohonen algorithm. Thus, a method of reorganizing k-means codebooks by the LK Heuristic algorithm is proposed as well. Furthermore, during the code vectors update step, the selection of the parameters of code vectors

update has been also discussed. From the experimental results, we observed that the AMDVQ approach using the Kohonen codebooks and the reordered k-means codebooks can both improve the intra coding efficiency compared with the MDVQ approach. The reordered k-means codebooks are even better than the Kohonen codebooks for AMDVQ approach. We conclude that the proposed AMDVQ residual coding improves significantly the intra coding efficiency compared with the ordinary HEVC standard.

Chapter 5

Conclusion and Future works

Obtained results

The objective of the research work described in this thesis was to develop new techniques to further improve the coding efficiency of the current HEVC video coding standard. To achieve this we have concentrated on techniques which reduce the correlations remaining in the residual data, after the intra prediction is performed. From the studies of bit rate consumption of the HEVC standard, we found that there are still room for prediction residual compression. We proposed three prediction residual compression algorithms which reduce the remaining redundancy in the residual domain.

The algorithm of Intra Residual Prediction (IRP) with template matching extends the scope of prediction from the pixel domain to the residual domain. The experimental results show that by exploiting the correlations in the residual domain, an upper bound of 11% on bit rate saving can be achieved, assuming that the cost of signaling the prediction is zero. To code efficiently the residual predictor, we investigated several aspects including how to restrict the search region and how to construct a candidate list. Our residual prediction algorithm provides 0.01% to 0.12% on bit rate saving compared with the HEVC standard. Even though the upper bound of the performance is encouraging, and even if there is a relatively high percentage of selection of our

method by the rate-distortion optimization, the overall gain is modest. We conclude that the amount of information is too high and negates the possible gains. If we want to reduce correlations in the residues efficiently, the past decoded residual information is not a pertinent source.

The second algorithm consists in applying the well-known vector quantization on residue of prediction. Vector quantization is an efficient data compression technique which has been widely used in image compression and speech coding. But in the field of video coding, the requirement of large dimension codebook is usually considered as too complex to be implemented in practical applications. However, the proposed algorithm, residual coding using mode dependent vector quantization, is able to provide significant coding gain with a generic codebook of small dimension. The experimental results show that our algorithm provides a bit rate reduction of 1.1% on average compared with the HEVC video coding standard. This coding performance is achieved by treating residuals of different prediction modes individually and optimizing codebook in a sense of rate-distortion optimization. Furthermore, we also demonstrated that the codebook of residual vector quantization is independent of the quantization parameter. This characteristic makes the approach suitable for real applications, since a single codebook per mode covers the whole compression range. Furthermore, the impact on decoding complexity is very limited.

Finally, we proposed an algorithm of residual coding using adaptive vector quantization. We designed an adaptive vector quantizer that gradually modifies the codebook in order to adapt to the input video sequences. The adaptivity of vector quantizer further improves the coding efficiency over the previous vector quantization algorithm from 1.1% to 1.5% on average. In particular when coding screen content video sequences, a bit rate saving of 3% is observed. For this algorithm, we used a modified version of the Kohonen update step, which is usually applied only for codebook construction. We also discussed the necessity of constructing an adequate initial codebook for adaptive vector quantization.

Perspectives and open problems

The algorithms developed during the thesis show that it is possible to obtain compression gain by exploiting the correlations remaining in the intra prediction residuals. Several interesting perspectives can be proposed to further continue the work done in this thesis.

The proposed approaches based on vector quantization have shown their ability to improve the compression performance while maintaining a decoder of low complexity. However, they still suffer from two disadvantages. First, they require the storage of several codebooks at the encoder and the decoder. Second, the encoder complexity is still considerable since a good matching code vector has to be found from the codebook. These disadvantages can be resolved by using a technique of Lattice vector quantization [CS82] which does not requires codebook storage at both the encoder and decoder. When applying the mode dependent vector quantization on inter prediction residuals, the main concern is how to estimate a intra prediction mode for an inter predicted block so that the same codebook can be used also for inter prediction residuals. Furthermore, Lattice vector quantization does not require a full search at the encoder side. Due to the structure of the Lattice, the nearest neighbor is obtained by a simple algorithm. Here, the challenge would consist in following the probabilities of the points of the Lattice for the different modes.

Another possible improvement consists in extending the proposed approach to inter prediction residuals. This would require to identify "classes" of inter prediction residuals similar to the intra prediction modes, if we want to share the same codebooks.

Appendix A

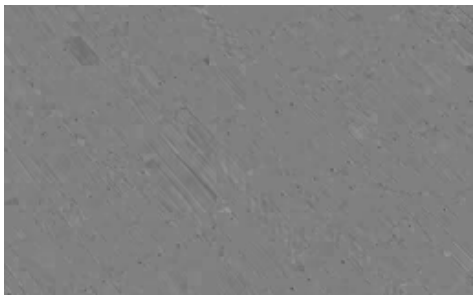


Figure A-1: Part of residue image of sequence BasketBallDrillText, coded with HEVC model.

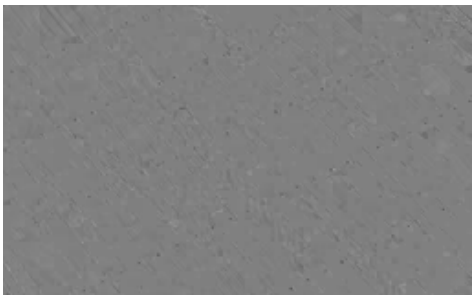


Figure A-2: Part of residue image of sequence BasketBallDrillText, coded with MDVQ.

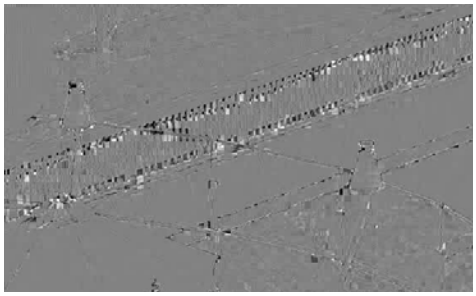


Figure A-3: Part of residue image of sequence BQTerrace, coded with HEVC reference model.

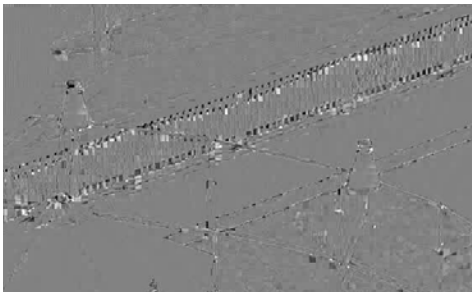


Figure A-4: Part of residue image of sequence BQTerrace, coded with MDVQ.

Bibliography

[AD12] E.A. Ayele and S.B. Dhok. Review of proposed high efficiency video coding (hevc) standard. *International Journal of Computer Applications*, 59:1–9, 2012.

[Ash01] M. Ashikhmin. Synthesizing natural textures. *ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[BFOS84] L. Breiman, J. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[Bjø08] G. Bjøntegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33, ITU-T SG16 Q.6 Document*, 2008.

[Bos08] F. Bossen. Common test conditions and software reference configurations. *Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG*, 2008.

[CGT+11] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, and P. Perez. Epitome-based image compression using translational sub-pel mapping. *IEEE International Workshop on Multimedia Signal Processing, MMSP*, 2011.

[CHJ11] G. Clare, F. Henry, and J. Jung. Sign data hiding. *JCTVC-G271, 7th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland*, 2011.

[CS82] J. Conway and N. Sloane. Fast quantizing and decoding and algorithms for lattice quantizers and codes. *IEEE Transactions on Information Theory*, 28:227–232, 1982.

[GG92] A. Gersho and R.M. Gray. Vector quantization and signal compression. *Kluwer Academic Press, Springer*, pages 649–654, 1992.

[GWL08] Y. Guo, Y. K. Wang, and H. Li. Priority-based template matching intra prediction. *Proc. IEEE International Conference on Multimedia and Expo*, pages 1117–1120, 2008.

[HH93] C.M. Huang and R.W. Harris. A comparison of several vector quantization codebook generation approaches. *IEEE Transactions on Image Processing*, 2:108–112, 1993.

[IT93] ITU-T. Video codec for audiovisual services at px64 kbits/s. *ITU-T Recommendation H.261 - Version 2*, 1993.

[IT94] ITU-T and ISO/IEC JTC 1. Generic coding of moving pictures and associated audio information - part 2: Video. *ITU-T Recommendation H.262 - ISO/IEC 13818-2 (MPEG-2)*, 1994.

[KML+12] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park. Block partitioning structure in the hevc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1697–1706, 2012.

[Koh90] T. Kohonen. The self-organizing map. *IEEE Proceedings*, 78:1464–1480, 1990.

[LCW13] T. Lin, X. Chen, and S. Wang. Pseudo-2d-matching based dual-coder architecture for screen contents coding. *IEEE International Conference on Multimedia and Expo Workshops, ICMEW*, pages 1–4, 2013.

[LK73] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operation Research*, 21:498–516, 1973.

[oIT03] Joint Video Team of ITU-T and ISO/IEC JTC 1. Draft itu-t recommendation and final draft international standard of joint video specification (itu-t rec. h.264 iso/iec 14496-10 avc). *JVT-G050, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG*, 2003.

[PSG+13] C. Pang, J. Sole, L. Guo, M. Karczewicz, and R. Joshi. Non-rce3: Intra motion compensation with 2-d mvs. *JCTVC-N0256, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11*, 2013.

[SG88] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing, ICASSP*, pages 1445–1453, 1988.

[SJK11] J. Sole, R. Joshi, and M. Karczewicz. Ce11: Parallel context processing for the significance map in high coding efficiency. *JCTVC-E338, 5th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland*, 2011.

[SJN+12] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clare, F. Henry, and A. Duenas. Transform coefficient coding in hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1765–1777, 2012.

[SKS+04] K. Sugimoto, M. Kobayashi, Y. Suzuki, S. Kato, and C.S. Boon. Inter frame coding with template matching spatio-temporal prediction. *IEEE International Conference on Image Processing, ICIP*, pages 465–468, 2004.

[SOHW12] G.J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1649–1668, 2012.

[TBS06] T.K. Tan, C.S. Boon, and Y. Suzuki. Intra prediction by template matching. *IEEE International Conference on Image Processing, ICIP*, pages 1693–1696, 2006.

[TBS07] T.K. Tan, C.S. Boon, and Y. Suzuki. Intra prediction by averaged template matching predictors. *IEEE Consumer Communications and Networking Conference, CCNC*, pages 405–409, 2007.

[TG09] M. Turkan and C. Guillemot. Sparse approximation with adaptive dictionary for image prediction. *IEEE International Conference on Image Processing (ICIP)*, pages 25–28, 2009.

[TG10] M. Turkan and C. Guillemot. Image prediction: Template matching vs. sparse approximation. *IEEE International Conference on Image Processing (ICIP)*, pages 789–792, 2010.

[TG12] M. Turkan and C. Guillemot. Image prediction based on neighbor-embedding methods. *IEEE Transactions on Image Processing*, 21:1885–1898, 2012.

[VB93] R. Veldhuis and M. Breeuwer. *An Introduction to Source Coding*. Prentice-Hall, 1993.

[WG93] S.-W. Wu and A. Gersho. Enhanced video compression with standardized bit stream syntax. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 36:103–106, 1993.

[WL00] L.Y. Wei and M. Levoy. Fast texture synthesis using tree-structure vector quantization. *Proceeding of SIG-GRAPH*, pages 479–488, 2000.

[WLM+96] T. Wiegand, M. Lightstone, D. Mukherjee, T.G. Campbell, and S.K. Mitra. Rate-distortion optimized mode selection for very low bit rate video coding and the emerging h.263 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:182–190, 1996.

[YK08] Y. Ye and M. Karczewicz. Improved h.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient

scanning. *IEEE International Conference on Image Process, ICIP*, pages 2116–2119, 2008.

[YWH+12] X. Yu, J. Wang, D. He, G. Martin-Cocher, and S. Campbell. Multiple sign bits hiding. *JCTVC-H0481, 8th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, San Jose, CA*, 2012.

[ZCW+11] Y. Zheng, M. Coban, X. Wang, J. Sole, R. Joshi, and M. Karczewicz. Ce11: Mode dependent coefficient scanning. *JCTVC-D393, 4th Joint Collaborative Team on Video Coding (JCT-VC) Meeting, Geneva, Switzerland*, 2011.

[ZYE+08] Y. Zheng, P. Yin, O.D. Escoda, X. Li, and C. Gomila. Intra prediction using template matching with adaptive illumination compensation. *IEEE International Conference on Image Processing, ICIP*, pages 125–128, 2008.