Multiview video plus depth coding for new multimedia services

Elie-Gabriel Mora

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2014-ENST-0007

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Signal et Images »

présentée et soutenue publiquement par

Elie Gabriel MORA

le 4 février 2014

Codage multi-vues multi-profondeur pour de nouveaux services multimédia

Directeur de thèse : **Béatrice PESQUET-POPESCU**
Co-directeur de thèse : **Marco CAGNAZZO**
Encadrement de thèse : **Joël JUNG**

Jury

Mme Christine GUILLEMOT, Directeur de Recherches, INRIA	Rapporteur
M. Laurent LUCAS, Professeur, IUT de Reims	Rapporteur
Mme Béatrice PESQUET-POPESCU, Professeur, Télécom ParisTech	Directeur de thèse
M. Marco CAGNAZZO, Directeur de Recherches, Télécom ParisTech	Co-directeur de thèse
M. Joël JUNG, Ingénieur R&D, Orange Labs	Encadrant de thèse
M. Marc ANTONINI, Directeur de Recherches, Laboratoire I3S Sophia-Antipolis	Président
M. Pierrick PHILIPPE, Ingénieur R&D, Orange Labs	Invité

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

To Khaled. To Raymonda.

Remerciements

Je souhaite en premier lieu exprimer toute ma reconnaissance à Joël Jung, ingénieur recherche à Orange Labs, pour son encadrement au cours de ces trois années de thèse. Son partage d'expériences et de connaissances, ses critiques pointilleuses, ses conseils et son suivi continu de l'avancement des travaux ont fortement contribué au succès de cette thèse.

Je souhaite également exprimer toute ma reconnaissance à Béatrice Pesquet-Popescu, professeur à l'Institut Mines-Télécom / Télécom ParisTech, pour la direction de mes travaux de thèse. Le succès de cette thèse a fortement dépendu de ses idées innovantes, de son soutient continu, et surtout de ses encouragements tout au long de ces trois dernières années. J'exprime aussi toute ma reconnaissance à Marco Cagnazzo, directeur de recherche à l'Institut Mines-Télécom / Télécom ParisTech pour ses idées, son pragmatisme, et ses critiques constructives lors de la co-direction de mes travaux.

Je souhaite remercier à présent Laurent Lucas et Christine Guillemot pour avoir assuré la charge de rapporteur. Leurs analyses ainsi que l'esprit critique dont ils ont fait preuve lors de l'évaluation de mes travaux ont été fortement appréciés. Je souhaite aussi remercier Pierrick Philippe pour la lecture de mon manuscrit, et bien sûr, Marc Antonini pour avoir présidé le jury.

Côté Orange Labs, je souhaite remercier mon manager Didier Gaubil pour ses encouragements, ainsi que tous les membres de l'équipe Compression Vidéo Avancée pour leurs conseils précieux : les deux Patrick, Rémi, Loic, Pierrick, Stéphane, Gordon, Félix, Gilles, Christophe et Joël. Un grand merci aussi à Régine pour le volet administratif. Une pensée très spéciale aux collègues que j'ai côtoyé à Issy-les-Moulineaux pendant une brève période mais qui ont laissé chacun une marque dans cette aventure : Jean-Marc, Julien, et Kartik, ainsi qu'à mes collègues actuellement en thèse : Khoa et Antoine à qui je souhaite courage et bonne chance pour le reste. A Télécom ParisTech, je souhaite remercier tout d'abord les permanents : Florence Besnard, Laurence Zelmar, Fabrice Planche, et Yves Grenier, ainsi que mes ex-collègues du groupe MMA : Marc, Azza, Olivier, Manel, Houssem, Abdalbassir, Irina et Rafael. Je salue aussi les doctorants et les post-doctorants toujours

présents : Maxime, Giuseppe, Giovanni, Claudio, Paul, Yafei, Aniello, Marco, Hamlet, Giovanni, Francesca, Marwa, Mireille, et enfin Andrei, dont le travail de stage a initié un des travaux amonts de cette thèse. Je les remercie pour leur convivialité, leur gentillesse, leurs échanges, leurs rires, leurs pleurs, leurs confidences, et leurs conseils. La thèse c'est aussi une expérience humaine, et elle n'aurait jamais été complète sans eux.

Un grand merci à mes trois meilleurs amis : Farid, Samy et Haithem pour avoir toujours été là à mes côtés. Je garderai toujours dans mon coeur le souvenir des apéros chez Samy, des brunchs chez Farid, des cinés / starbucks avec Haithem, de nos vacances à Barcelone... Je les remercie aussi pour leur aide dans la préparation du pot de soutenance. C'est grâce à eux que c'était un franc succès ! Je remercie spécialement Farid pour tous les bons moments qu'on a partagés ensemble, pour son grand coeur, pour le simple fait d'être soi-même ! Bien sûr, je remercie aussi mes amis de la Maison du Liban, notamment Joe, mon ami d'enfance, et plus généralement mes ex-voisins du 2$^{\text{ème}}$ étage, trop nombreux pour être tous nommés.

Finalement, je souhaite remercier ma famille en commençant par ma mère Raymonda, et mes deux soeurs : Anita et Mirella. Sans leur soutient et leur amour, cette thèse n'aurait jamais vu le jour, je les remercie donc du fond du coeur.

Résumé

Introduction

La vidéo 3D connaît un développement rapide ces dernières années, des sorties de plus en plus fréquentes de films 3D à l'émergence de nouveaux services 3D comme la télévision 3D (3DTV) et la Free Viewpoint TV (FTV). Bien que la vidéo 3D n'a pas encore atteint le succès attendu (dû principalement aux inconforts visuels et à la nécessité de porter des lunettes 3D tout au long de la séance), elle est attendue à conquérir rapidement le marché avec le visionnage auto-stéréoscopique, qui requiert un nombre important de vues à multiplexer simultanément au récepteur.

Le format stéréo classique (mettant en jeu deux vues uniquement) étant du coup insuffisant, le format multi-vue (MVV pour Multi-View Video), composé de plusieurs vues représentant la même scène mais légèrement décalées les unes par rapport aux autres, permet de subvenir aux exigences de la 3DTV et la FTV. Or le coût de transmission des différentes vues dans le format MVV peut toutefois rester exorbitant, même si les corrélations inter-vues sont exploitées avec des codeurs tels que Multi-view Video Coding (MVC). Le format Multi-view Video + Depth (MVD) est une alternative intéressante à MVV car il introduit des vidéos de profondeur qui sont moins coûteuses à coder que des vidéos de texture, et qui peuvent être utilisées pour synthétiser autant de vues nécessaires au récepteur avec la technique de Depth Image Based Rendering (DIBR).

Or il n'y a actuellement aucun standard conçu pour coder efficacement des vidéos 3D dans un format MVD. Le groupe MPEG s'est penché sur le sujet en 2011, et parmi les différentes voies de standardisation qui en ont découlé, l'une concerne une extension 3D du standard HEVC (3D-HEVC) qui permettrait un codage efficace des données MVD. C'est dans ce contexte que se situent justement les travaux de cette thèse. L'objectif est de développer des outils qui permettent d'augmenter l'efficacité de codage des vues dépendantes, des vidéos de profondeur, et des vidéos synthétisées dans 3D-HEVC. Le fil conducteur des approches proposées est l'amélioration de la prédiction des informations de codage transmises dans un flux binaire 3D-HEVC.

Les travaux sont divisés en deux catégories, la première regroupant des approches

conventionnelles orientées vers la normalisation et la deuxième regroupant des approches amont, en rupture avec l'état de l'art actuel. Les approches de la première catégorie sont soumises à des contraintes strictes (en matière de complexité, de nombre de calculs, de mémoire RAM utilisée...) afin d'être facilement intégrables dans des standards 3D tel que 3D-HEVC. Les approches de la deuxième catégorie par contre, ne sont soumises à aucune contrainte. L'accent est plus mis sur l'innovation et la recherche pure, et ceci permet d'obtenir des gains de codage plus significatifs que ceux obtenus dans la première catégorie.

1 Représentation et codage vidéo 3D

Dans le premier chapitre de cette thèse, nous introduisons les différents indices à partir desquels le système visuel humain perçoit la profondeur. On distingue les indices physiologiques des indices monoculaires et binoculaires. Ces derniers permettent en particulier de recréer une sensation de profondeur avec des moyens artificiels, d'où la stéréoscopie qui domine le marché de la vidéo 3D actuellement. Bien sûr, d'autres formats 3D existent aussi. Ils peuvent être divisés en deux catégories principales : les formats à base de texture uniquement, et les formats à base de profondeur, dont MVD.

Le chapitre introduit ensuite des outils de codage de vidéos 3D sous un format MVD. En l'occurrence, il s'agit d'outils qui permettent de coder plus efficacement les vues de texture et de profondeur en exploitant les corrélations inter-vues mais aussi inter-composantes (texture-profondeur). Certains outils spécifiques au codage de la profondeur y sont détaillés également.

Finalement, ce chapitre présente différents standards vidéo capables d'encoder des vidéos 3D sous un format MVD. Parmi les plus efficaces, on note HEVC, MVC, une extension multi-vue de HEVC (MV-HEVC), et 3D-HEVC. MVC est une extension multivue du standard H.264/AVC qui permet d'exploiter les corrélations inter-vues pour augmenter l'efficacité de codage. MV-HEVC est similaire, sauf que basé HEVC et non AVC. MVC et MV-HEVC permettent de coder des vidéos 3D sous un format MVD en codant séparément les textures et les profondeurs. 3D-HEVC, en cours de standardisation, permet un codage conjoint des textures et des profondeurs pour augmenter encore plus l'efficacité de codage. Des outils spécifiques au codage de la profondeur y sont inclus également.

HEVC, MVC, MV-HEVC et 3D-HEVC sont ensuite comparés. Deux scénarios sont considérés, où doivent être codées 9 vues de texture dans l'un (format MVV), et 3 vues de texture (parmi les 9) ainsi que leurs profondeurs dans l'autre (format MVD), sachant que dans ce scénario, après décodage, 6 vues sont synthétisées pour

revenir aux 9 initiales. Dans le scénario 1, MVC donne des gains de -7.5% en moyenne par rapport à un codage HEVC simulcast des différentes vues. En réalité, MVC ne commence à donner des gains qu'à partir de la 4$^{\text{ème}}$ vue codée, où l'exploitation des corrélations inter-vues devient plus efficace que le codage mono-vue de HEVC qui est meilleur que celui de MVC basé AVC. MV-HEVC ne souffre pas d'un codage mono-vue inefficace car l'extension est basée elle-même sur HEVC. Du coup l'exploitation des corrélations inter-vue améliore l'efficacité de codage sans pour autant être contrée par des pertes, ce qui justifie un gain moyen de -37.1%. Dans le scenario 2, MVC donne des pertes importantes (73.2%) par rapport à HEVC simulcast car moins que 4 vues sont codées. MV-HEVC donne -28.8% de gains, alors que 3D-HEVC donne -45.6% à cause de l'exploitation des corrélations inter-composantes (texture-profondeur) et des outils de codage de la profondeur. En comparant les deux scénarios par codec utilisé, on remarque que le scénario 2 est plus efficace que le premier dans tous les cas (-23.7% de gains avec MVC, -31.2% avec MV-HEVC, et -55.5% avec HEVC), ce qui prouve que MVD est plus rentable que MVV pour représenter des vidéos 3D.

2 Outils de codage dans 3D-HEVC

Dans le deuxième chapitre, nous présentons les outils de codage dans 3D-HEVC. Une vue dite vue de base est en premier codée avec HEVC pour maintenir une compatibilité arrière avec le standard. Puis, les autres vues de texture, dites vues dépendantes, et les vidéos de profondeur sont codées avec des outils additionnels pour un codage plus efficace. Dans 3D-HEVC, on code par unité d'accès. Une unité d'accès comporte toutes les trames de texture et de profondeur à un instant donné t. La structure de codage est détaillée à la Figure 1.

La vue de base est codée avec HEVC sans aucun outil additionnel pour garantir la compatibilité arrière. La structure de codage dans HEVC est une structure en arbres quaternaires. Elle est illustrée à la Figure 2. L'image est découpée en Coding Tree Units (CTU). Chaque CTU peut être divisée en quatre Coding Units (CU), et chaque CU peut elle-aussi être divisée en quatre autres CU et ainsi de suite jusqu'à arriver à une taille minimale autorisée de CU. Chaque CU non splittée est divisée en une ou plusieurs Prediction Units (PU). Différentes tailles de partitionnement existent : carrées (2N×2N et N×N), rectangulaires (2N×N, N×2N) mais aussi asymétriques (2N×nU, 2N×nD, nL×2N, nR×2N) appellées Asymetric Motion Partition (AMP). Une PU ne peut être partitionnée en d'autres PU. Notons que c'est au niveau d'une CU qu'est sélectionné un mode de codage (Intra, Inter ou Merge), et que c'est au niveau des PUs que sont déterminés les informations de prédiction qui correspondent

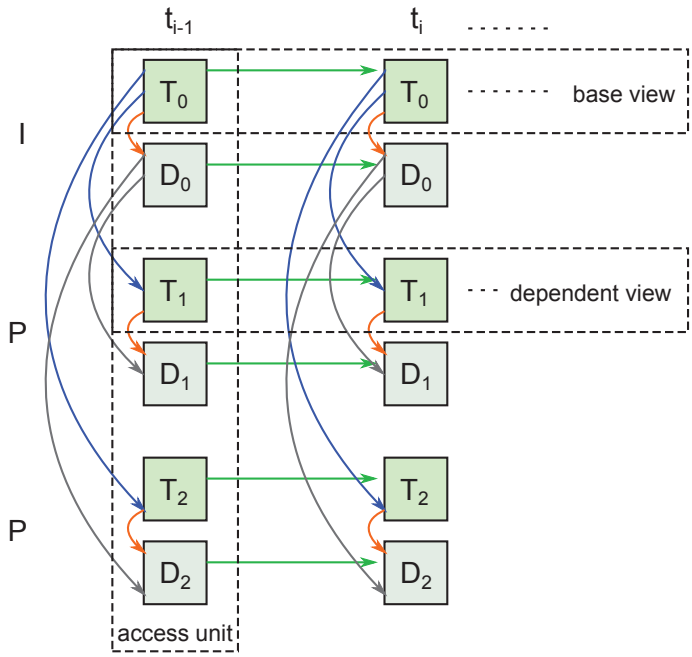


Figure 1: Structure de codage dans 3D-HEVC

au mode de codage choisi (directions Intra si la CU est codée en Intra, vecteurs de mouvement si la CU est codée en Inter, ...). Indépendamment du partitionnement en PUs, une CU peut aussi être divisée en plusieurs Transform Units (TU). C'est au niveau des TUs qu'est appliquée la transformée après le calcul du résiduel. La division en TUs définit donc ce qu'on appelle un arbre de transformée ou Residual QuadTree (RQT). Un processus d'optimisation débit-distorsion ou Rate Distortion Optimization (RDO) va tester, pour chaque CU, toutes les combinaisons Mode de codage / Partitionnement en PUs / RQT et va aussi tenter de splitter la CU en 4 où il va refaire la même chose pour les 4 CUs ainsi générées. La configuration qui résulte en le moindre coût pour la CU donnée va être choisie pour le codage de cette CU.

Une CU peut être codée en Intra, Inter, et Merge. Dans le mode Intra, une PU est prédite à partir de ses pixels voisins causaux dans la même trame. La ligne au-dessus et la colonne de gauche forment la liste des pixels de référence. Dans HEVC, 34 modes Intra existent dont 2 non-angulaires : le DC et le Planar, comme l'illustre la Figure 3.

Le mode Intra est en général codé sur 6 bits. Pour réduire ce coût de signalisation, HEVC introduit l'outil Most Probable Mode (MPM) qui consiste à établir une liste de deux candidats qui sont les modes Intra des PUs à gauche et au-dessus de la PU courante. Si le mode Intra de la PU courante est égal à l'un des deux candidats MPM, seuls un bit qui signale qu'une correspondance a été trouvée, et un autre qui signale lequel des deux correspond au mode courant sont transmis dans le flux

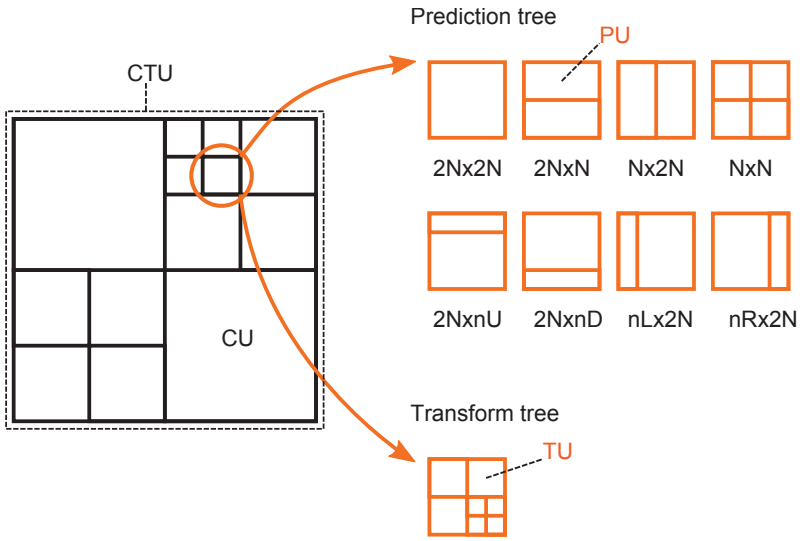


Figure 2: Structure de codage en arbres quaternaires dans HEVC

binaire. Le mode courant est du coup codé sur 2 bits au lieu de 6.

Le mode Inter consiste en une estimation de mouvement où un vecteur de mouvement (MV) est déterminé pour une PU donnée. Un MV fait correspondre la PU courante à la PU qui minimise un critère de distorsion dans une trame précédemment codée. L'estimation de mouvement est suivie d'une compensation de mouvement qui consiste à utiliser le vecteur de mouvement pour construire la prédiction et pouvoir ainsi calculer un résiduel pour la PU courante. Le MV doit être transmis au décodeur. Pour réduire le coût de signalisation, il va être prédit et seul un résiduel de MV va finalement être transmis. Le processus de prédiction du MV dans HEVC s'appelle Advanced Motion Vector Prediction (AMVP). Il consiste à établir une liste de deux candidats qui sont les MV de PU voisines à la PU courante.

Finalement, le mode Merge permet à une PU courante d'hériter les paramètres de mouvement (MV + indices de référence temporelle) d'une PU voisine. Les paramètres de mouvement des PU voisines forment la liste des candidats du Merge. Il y a 5 candidats dans cette liste, 4 spatiaux (A_1, B_1, B_0, A_0) et un temporel (C_0), comme le montre la Figure 4. Des candidats combinés et des candidats zéro viennent remplir la liste s'il manque des candidats. RDO vérifie ensuite les paramètres de mouvement de chacun des 5 candidats et l'indice de celui qui minimise le plus le coût RD va être transmis au décodeur.

Les vues dépendantes sont aussi codées avec HEVC auquel viennent s'ajouter des outils qui vont exploiter les corrélations inter-vues. Le premier est la prédiction en compensation de disparité (DCP). DCP ajoute de la syntaxe additionnelle afin que les trames d'une autre vue au même instant temporel soient insérées dans la

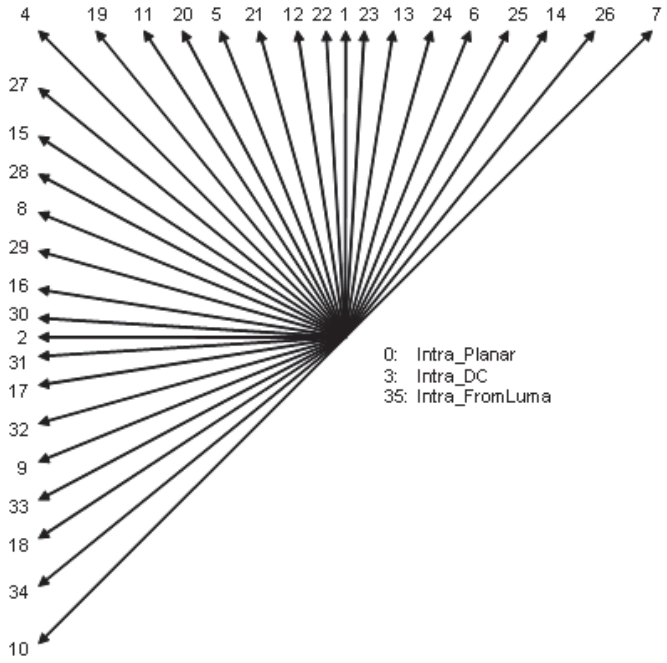


Figure 3: Liste des modes Intra dans HEVC

liste des trames de référence d'une trame courante. Un MV pointant vers un bloc dans une trame de référence inter-vue n'est plus appelé MV mais plutôt vecteur de disparité (DV). Une PU qui a été codée avec un DV est dite être codée en DCP (à l'opposé des PU codées avec un MV qui sont dites être codées en MCP).

Le deuxième outil est la prédiction du mouvement inter-vues ou IVMP. Dans IVMP, un DV est dérivé pour une PU courante de la même manière à l'encodeur et au décodeur. Le processus de dérivation de ce DV est le Neighboring Disparity Vector (NBDV). Ce DV pointe vers une PU qui correspond à la PU courante dans une vue de référence, comme le montre la Figure 5. Le MV de cette PU de référence

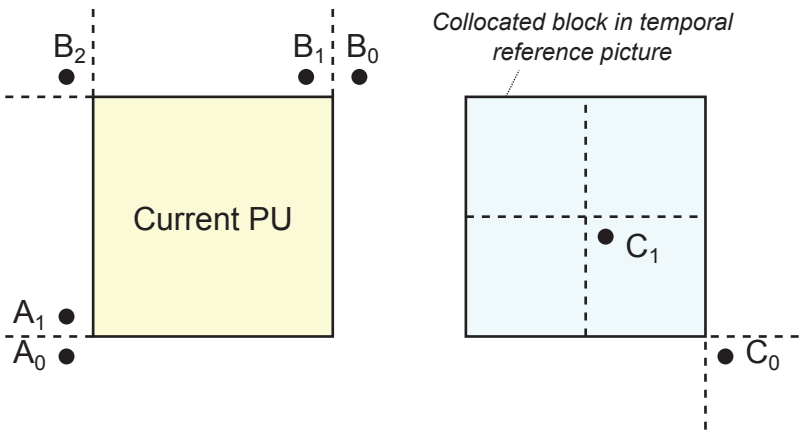


Figure 4: Candidats du Merge dans HEVC

est ensuite ajouté à la liste des candidats AMVP et Merge de la PU courante. Plus précisément, le MV est ajouté comme candidat multi-vue à la première position de la liste des candidats du Merge, liste qui comprendra donc 6 candidats au lieu de 5. Si le MV n'existe pas (si la PU de référence est codée en Intra par exemple), c'est le DV lui-même qui sera ajouté comme candidat multi-vue.

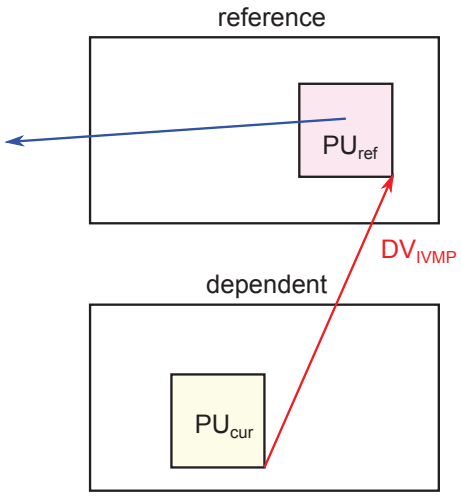


Figure 5: Prédiction du mouvement inter-vues (IVMP)

La prédiction du résiduel inter-vue (IVRP) est un troisième outil pour coder les vues dépendantes, qui consiste à prédire le résiduel de la PU courante avec le résiduel de la PU de référence pointée par le DV dérivé avec NBDV.

NBDV, quant à lui, est un simple processus de recherche d'un DV parmi des PU voisines. Les mêmes positions spatio-temporelles voisines que celles utilisées pour le Merge sont parcourues, et le DV de la première PU trouvée codée en DCP est choisi comme DV final utilisé pour IVMP et IVRP.

Par ailleurs, plusieurs outils de codage de vidéos de profondeur sont inclus dans 3D-HEVC : les modes de modélisation de la profondeur (DMM), l'héritage des paramètres de mouvement (MPI), le codage simple de la profondeur (SDC), et enfin le codage Region Boundary Chain (RBC).

3 Codage des vidéos de profondeur par héritage des modes Intra de texture

Dans ce troisième chapitre, nous présentons notre outil de codage des vidéos de profondeur basé sur l'héritage des modes Intra de texture. En analysant les coûts de signalisation de différents éléments transmis dans un flux binaire 3D-HEVC dans la version 0.3 du software de référence de 3D-HEVC : HTM-0.3, on remarque que la signalisation des modes Intra représente 25% du débit total alloué à la profondeur.

Des gains significatifs peuvent être alors obtenus avec une meilleure prédiction de ces modes Intra. L'idée initiale proposée est de prédire systématiquement les modes Intra de profondeur avec les modes Intra des blocs colocalisés en texture. Or une analyse montre que les modes ne sont identiques que dans 18% des PUs et que la correspondance est plus marquée dans les zones où on a des contours bien marqués en texture. En effet, un contour bien marqué en texture définit une structure géométrique dans la scène qui, en général, existerait aussi en profondeur. Comme les modes Intra sont directionnels, ils vont suivre la direction de cette structure aussi bien en texture qu'en profondeur, et par conséquent, les modes Intra de texture et de profondeur choisis par RDO vont être identiques. L'idée proposée dans ce chapitre est donc de prédire le mode Intra d'une PU de profondeur avec celui de la PU de texture colocalisée, mais uniquement si cette dernière comprend un contour bien marqué.

La méthode proposée est une méthode de codage inter-composantes (où l'on code une composante à partir d'une autre, en l'occurrence ici la profondeur à partir de la texture). D'autres méthodes inter-composantes incluent DMM (modes 3 et 4), MPI, View Synthesis Prediction (VSP), etc. Une méthode similaire a été proposée par ETRI où le mode Intra de texture est systématiquement ajouté à la liste des candidats prédicteurs MPM de la PU de profondeur colocalisée. Si la liste est déjà pleine, un des deux candidats MPM spatiaux présents est forcément remplacé par le candidat de texture. Or rien ne garantit que ce dernier soit meilleur que le candidat remplacé, et si ce n'est effectivement pas le cas, ceci résultera en des pertes en efficacité de codage. Notre méthode consiste donc à ajouter le mode Intra de texture à la liste des candidats MPM de la PU de profondeur colocalisée uniquement si la PU comprend un contour bien marqué en texture, où la similarité entre les deux modes est la plus probable.

Notre algorithme est illustré à la Figure 6. La première étape consiste à trouver la PU de texture colocalisée à une PU de profondeur courante. Dans une deuxième étape, on calcule un critère sur cette PU de texture qui estime la probabilité que les modes Intra de texture et de profondeur soient identiques. D'après notre analyse précédente, ce critère devra détecter s'il y a un contour bien marqué dans la PU de texture. Dans une troisième étape, on va comparer le critère calculé à un seuil. Si la valeur du critère est supérieure au seuil, on va hériter le mode Intra de texture pour la profondeur, ce qui veut dire concrètement qu'il sera ajouté dans la liste des candidats MPM de la PU de profondeur courante, remplaçant ainsi un candidat spatial si la liste est déjà pleine.

Nous proposons deux critères différents pour notre méthode qu'on évalue indépendamment. Les deux reposent sur un filtrage au préalable de la PU de texture

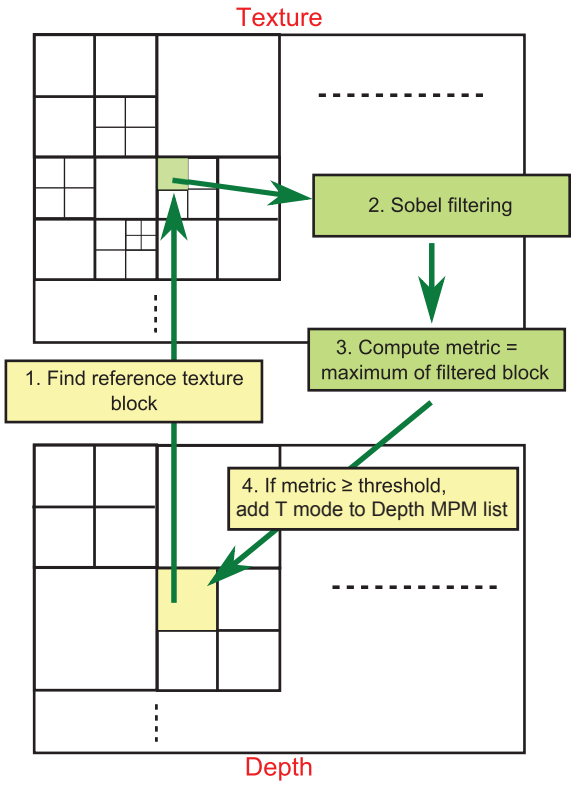
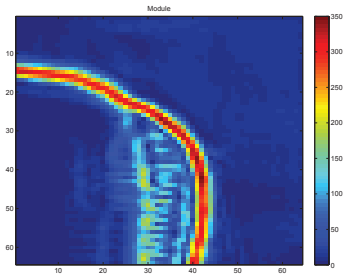


Figure 6: Algorithme de la méthode proposée

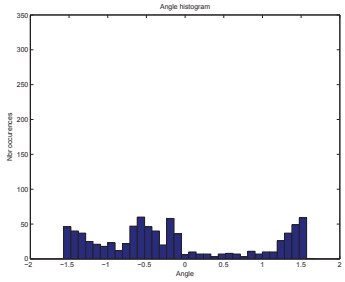
avec un filtre de détection de contour comme Sobel. Le filtrage donne les deux composantes (horizontale et verticale) du gradient : G_x et G_y à partir desquelles peuvent être calculés le module du gradient : $M = \sqrt{G_x^2 + G_y^2}$ et la matrice des angles : $A = \arctan\left(\frac{G_y}{G_x}\right)$. Le maximum du module du gradient nous donne un premier critère : GradientMax, et l'exploitation de l'histogramme des angles dont le module est supérieur à un certain seuil nous donne un deuxième critère : DominantAngle. GradientMax permet de détecter s'il y a un contour bien marqué en texture, alors que DominantAngle détecte s'il y a un contour bien marqué mais aussi directionnel, c'est-à-dire s'il suit une direction bien définie.

Prenons l'exemple de la Figure 7. Dans le premier cas, la PU de texture comprend un contour bien marqué mais non directionnel. La CU correspondante sera probablement splittée à l'encodeur car il n'y a aucun mode Intra qui permet de prédire efficacement la structure géométrique présente dans la PU. Du coup le mode Intra de cette PU de texture ne sera pas pertinent pour la profondeur, et dans ce cas, il vaudrait mieux ne pas l'hériter. Or le critère GradientMax dans ce cas va donner une valeur élevée (346) car un contour marqué est bien présent dans la PU. DominantAngle par contre détectera que le contour en question n'est pas directionnel et donnera une valeur faible (4). Si on spécifie un seuil égal à 200, on va hériter dans le premier cas mais pas dans le deuxième, et c'est exactement ce qui est requis.

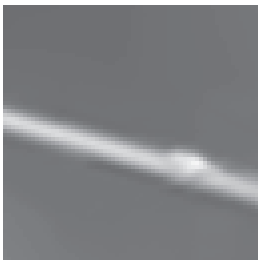
(a) PU de texture comprenant un contour bien marqué non directionnel (CMND)



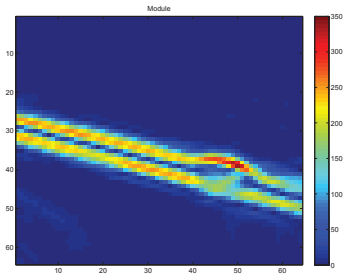
(b) Module de la PU de texture comprenant un CMND



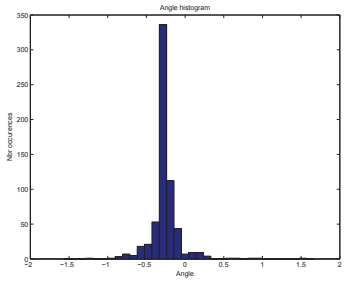
(c) Histogramme des angles de la PU de texture comprenant un CMND



(d) PU de texture comprenant un contour bien marqué directionnel (CMD)



(e) Module de la PU de texture comprenant un CMD



(f) Histogramme des angles de la PU de texture comprenant un CMD

Figure 7: Deux types de PU de texture comprenant des contours bien marqués

Bien sûr, quand la PU de texture comprend un contour bien marqué et directionnel, comme à la Figure 7(d), les deux critères vont donner des valeurs élevées (337 et 344) impliquant un héritage dans les deux cas.

Nous avons implémenté notre méthode et les deux critères dans le HTM-0.3. Nous avons respecté les conditions communes de test (CTC) définies par le groupe JCT-3V qui dirige les activités de standardisation de 3D-HEVC, sauf que nous avons choisi une configuration Intra (où toutes les trames sont codées en Intra) pour mieux évaluer notre méthode. Les séquences ont été codées avec quatre couples de QP ($QP_{texture}$ / $QP_{profondeur}$) : 25/34 30/39 35/42 40/45. Nous avons testé notre méthode sur les 7 séquences décrites dans les CTC. Il s'agit de séquences de 10 secondes chacune (entre 250 et 300 trames) mais nous n'avons choisi de coder que 0.5 secondes de vidéo pour accélérer les simulations. Nous donnons ainsi les résultats sur 0.5 secondes de vidéo mais aussi sur la première trame codée. En effet, le seuil auquel on compare la valeur du critère a besoin d'être optimisé. Or l'optimiser sur toutes les trames serait long et complexe. Par conséquent, dans notre méthode, nous optimisons le seuil uniquement sur la première trame codée et nous l'utilisons tel quel pour les autres trames.

Les Tableaux 1 et 2 donnent les gains sur les vues synthétisées et sur les vidéos de profondeur avec notre méthode, pour les deux critères GradientMax et DominantAngle, et pour, respectivement, la première trame codée et les 0.5 secondes de vidéo. Notons que ces gains sont mesurés avec la métrique de Bjontegaard (BD-Rate). DominantAngle, comme prévu, donne de meilleurs gains que GradientMax car il permet de prendre en compte la direction des contours.

Séquence	**DominantAngle**		**GradientMax**	
	Gains sur VS (in %)	**Gains sur profondeur (en %)**	**Gains sur VS (en %)**	**Gains sur profondeur (en %)**
GT Fly	−1.4	−1.4	−3.0	+6.1
Newspaper	−1.4	−2.9	−2.0	−0.7
PoznanHall2	−3.1	−7.8	−1.4	−2.7
Kendo	−0.6	+1.1	−1.0	+0.3
Balloons	−0.9	−1.9	−0.7	−0.1
Dancer	−2.4	−2.4	−0.6	−1.4
PoznanStreet	−1.2	−1.2	−0.2	−0.4
Moyenne	**−1.6**	**−2.3**	**−1.3**	**+0.2**

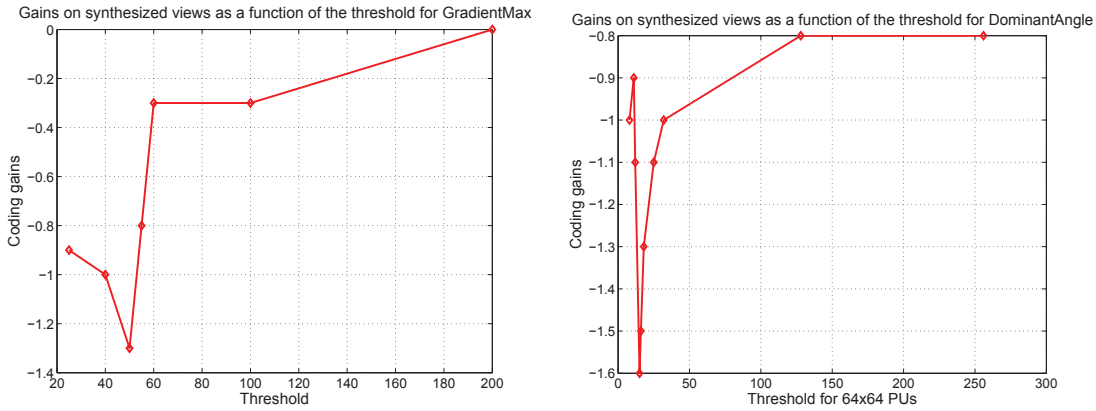
Table 1: Gains de codage (en BD-Rate) sur la première trame des vues synthétisées et des vidéos de profondeur

Séquence	**DominantAngle**		**GradientMax**	
	Gains sur VS (en %)	**Gains sur profondeur (en %)**	**Gains sur VS (en %)**	**Gains sur profondeur (en %)**
GT Fly	−0.5	+1.6	−0.6	0.0
Newspaper	−0.9	−1.3	−0.9	−0.8
PoznanHall2	−2.1	−0.6	−1.3	−0.5
Kendo	−0.9	−0.8	−1.2	−1.2
Balloons	−0.6	−0.6	−1.1	−0.4
Dancer	−1.2	−1.2	−0.5	−0.6
PoznanStreet	−0.7	−2.0	−0.7	−1.3
Moyenne	**−1.0**	**−0.7**	**−0.9**	**−0.7**

Table 2: Gains de codage (en BD-Rate) sur 0.5 secondes de vues synthétisées et de vidéos de profondeur

L'écart-type des gains entre les différentes séquences est de 0.5 pour DominantAngle et 0.29 pour GradientMax. Les deux écarts-types sont assez faibles, et on peut donc en conclure que les deux critères sont statistiquement stables. En comparant la variation des gains en fonction du seuil choisi pour les deux critères, illustrée à la Figure 8, on remarque que dans DominantAngle, si on s'éloigne de la valeur optimale du seuil (200 au lieu de 15 par exemple), la moitié du gain maximum est conservée (-0.8% au lieu de -1.6%), alors qu'avec GradientMax, on perd tous les gains en passant du seuil optimal 50 à 200 par exemple. En conclusion,

DominantAngle donne plus de gains en moyenne que GradientMax et est aussi plus robuste en matière de sélection du seuil. Notre méthode a été le sujet d'un article de journal publié dans APSIPA Transactions on Signal and Information Processing en 2013.



(a) GradientMax (b) DominantAngle

Figure 8: Gains sur les vues synthétisées en fonction du seuil

4 Amélioration de IVMP dans 3D-HEVC

En analysant les modes de codage des PUs des vues dépendantes dans le HTM-4.1, on remarque qu'il y a peu de PUs codées en DCP (16% uniquement). La Figure 9 montre une trame d'une vue dépendante. Les blocs roses représentent des PUs codées en DCP, minoritaires par rapport aux blocs gris et verts codés en MCP. Une des causes possibles de ce déséquilibre serait une mauvaise estimation des DV. En effet, si un bloc contient deux objets de profondeur différentes, ces deux objets auraient donc une disparité différente, et du coup l'estimation d'un seul vecteur de disparité pour ce bloc ne permettrait pas d'obtenir une prédiction efficace. Une autre cause serait la mauvaise prédiction des DV eux-mêmes. En effet on remarque aussi qu'il y a rarement de DVs dans la liste des candidats du Merge des PUs des vues dépendantes. Or le Merge est sélectionné pour 92% des PUs dans le HTM-4.1, donc s'il y avait des candidats DV dans cette liste, on aurait une bien meilleure répartition entre DV et MV et cela augmenterait les gains de codage.

Actuellement, pour une PU donnée, un DV et un MV sont estimés. Le DV peut donner une meilleure prédiction que celle du MV mais comme il y a rarement de candidats DV dans la liste du Merge, pour coder le DV, AMVP devra être utilisé et un résiduel de DV devra être transmis. Le MV par contre pourrait se retrouver dans la liste des candidats du Merge et dans ce cas, son codage se résumerait à l'envoi d'un simple indice. L'encodeur va alors sûrement préférer coder la PU en

MCP plutôt qu'en DCP, créant ainsi le déséquilibre MV/DV. Notre idée est donc de rééquilibrer la donne en insérant un DV dans la liste des candidats du Merge.

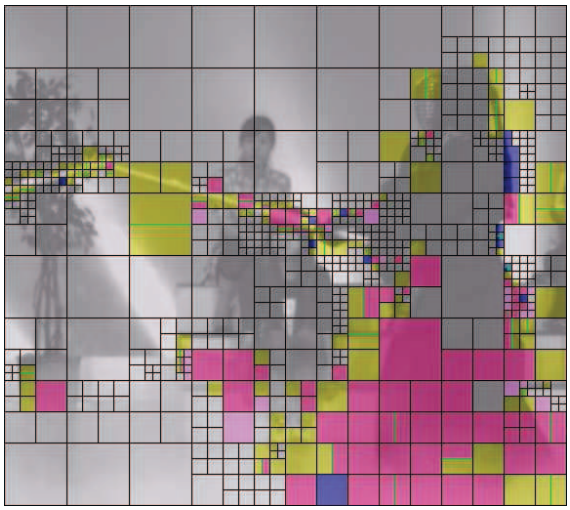


Figure 9: Mode de codages des PUs dans une trame d'une vue dépendante codée avec le HTM-4.1

Différentes méthodes dans la litérature proposent de modifier la liste des candidats du Merge. Dans une contribution au 2ème meeting JCT-3V, la liste primaire des candidats est parcourue et le premier DV trouvé est utilisé pour calculer 2 candidats additionnels en ajoutant un offset positif et négatif à ce DV. Ces candidats sont ensuite utilisés pour compléter la liste s'il manque des candidats. Cette méthode n'était pas très efficace à l'époque car il y avait rarement des DV dans la liste primaire. Dans une autre contribution, une vérification de redondance entre le candidat multi-vue et les deux premiers candidats spatiaux a été ajoutée, ce qui a donné -0.3% de gain et une adoption conséquente dans 3D-HEVC. Différentes autres méthodes ont été proposées aussi mais aucune n'a été spécialement conçue pour équilibrer la sélection de DCP par rapport à MCP. Et c'est justement ce qu'on propose dans ce chapitre.

Dans notre méthode qu'on appelle Additional Inter-View Merge Candidate (AIMC), on va ajouter un DV à la liste des candidats du Merge. Dans 3D-HEVC, un DV est dérivé avec NBDV pour chaque PU d'une vue dépendante pour construire le candidat multi-vue du Merge (voir Section 2). Ce dernier correspond au MV de la PU de référence pointée par ce DV s'il existe. Sinon, il correspondait directement au DV. Nous proposons ici d'insérer ce DV en tant que candidat inter-vue si le candidat multi-vue est un MV.

On propose deux méthodes d'insertion. Dans la première (AIMC-1) on insère le DV dans la liste de candidats secondaires. Le DV va venir compléter la liste primaire uniquement s'il y a des positions vides. Sinon, le DV ne va pas être ajouté. Dans une deuxième méthode d'insertion (AIMC-2), le candidat est toujours inséré avant le 4ème candidat spatial. Du coup c'est le candidat temporel qui est déplacé

à la liste secondaire et ne sera donc rajouté dans la liste primaire que s'il y a des positions vides. Notons que dans les deux cas, avant d'insérer le candidat inter-vue, une vérification de la redondance du candidat avec tous les autres candidats qui le précèdent dans la liste est effectuée, et s'il est redondant, il ne sera pas ajouté.

On a testé notre méthode dans le HTM-4.1 qui nous a aussi servi de référence et on a respecté les CTCs. Ici on a testé notre méthode sur l'ensemble des trames des différentes séquences (10 secondes de vidéo). Dans le Tableau 3 qui représente les gains en BD-Rate, les 2è, 3è, et 4è colonnes représentent respectivement les gains de codage sur la vue de base et sur les deux vues dépendantes. Comme on n'applique pas notre méthode à la vue de base, on a 0% de gain évidemment. Pour la 5è colonne, on considère la moyenne des trois PSNR et la somme des débits des 3 vues de texture pour calculer le BD-rate. La 6è colonne donne les gains sur les vues synthétisées. On synthétise au total 6 vues donc 3 entre la vue gauche et la centrale (qui est la vue de base) et trois autres entre la centrale et la droite. On considère la moyenne des 6 PSNR et la somme des 3 débits de texture et les 3 débits de profondeur pour calculer ce BD-Rate. La 7è colonne considère la moyenne des PSNR des 9 vues (donc 6 vues synthétisées et 3 vues codées) et la somme des 3 débits de texture et les 3 débits de profondeur pour calculer le BD-Rate. Enfin, les deux dernières colonnes donnent les runtimes à l'encodeur et au décodeur. On remarque des gains de -0.5% et -0.6% sur les deux vues dépendantes avec AIMC-1 et -0.6% sur les deux vues dépendantes avec AIMC-2. On a aussi des gains en runtime de 3-4% avec la méthode proposée. On remarque aussi que la sélection de DCP a été augmentée de 8.3% dans les deux méthodes ce qui explique les gains. Ceci est visible d'ailleurs sur la Figure 10 qui montre plus de blocs roses par rapport à la référence. En proposant un candidat additionnel, on réduit aussi le nombre de candidats secondaires à construire de 9%. Or cette construction est un peu complexe en général ce qui explique pourquoi on a gagné également en runtime. Enfin, on remarque qu'en enlevant l'étape de vérification de la redondance qui est l'élément le plus complexe de la méthode proposée, on réduit les gains de codage sans pour autant gagner en runtime, donc il vaut mieux laisser cette étape. Les deux méthodes ont été proposées au 2$^{\text{ème}}$ meeting JCT-3V et AIMC-2 a été adoptée dans le texte et dans le logiciel de référence de 3D-HEVC. AIMC a aussi fait le sujet d'un article accepté et présenté à la conférence Multi Media Signal Processing (MMSP) en 2013.

Toujours dans le but d'améliorer IVMP, on s'attaque à présent directement à NBDV, le processus de dérivation du DV (qui, à présent, correspond au candidat inter-vue ajouté dans AIMC). On remarque au fait que NBDV est sous-optimal. En effet dans NBDV, le premier DV trouvé est utilisé directement pour IVMP et

Méthode	Vidéo				Synt.	Codé +Synt	Runtimes	
	0	1	2	Avg			Enc	Dec
AIMC-1	0.0	-0.5	-0.6	-0.2	-0.2	-0.2	97	100
AIMC-1-NORC	0.0	-0.4	-0.4	-0.1	-0.1	-0.1	98	101
AIMC-2	0.0	-0.6	-0.6	-0.2	-0.2	-0.2	96	99
AIMC-2-NORC	0.0	-0.5	-0.4	-0.2	-0.1	-0.1	98	100

Table 3: Moyenne des gains de codage (en BD-Rate) de AIMC-1 et AIMC-2 et de la variante qui enlève l'étape de vérification de la redondance (NORC)

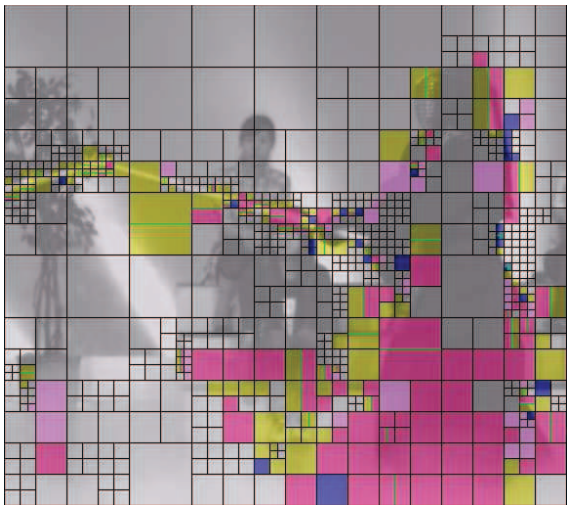


Figure 10: Mode de codages des PUs dans une trame d'une vue dépendante codée avec AIMC

IVRP et le processus de recherche s'arrête là, sans aucune garantie d'optimalité. On pourrait au fait avoir des voisins qui possèdent un meilleur DV et qui ne seront jamais parcourus. Bien qu'il y ait plusieurs contributions dans la littérature qui ont proposé de modifier NBDV, aucune n'a tenté de résoudre ce problème particulier.

Notre méthode qu'on appelle MedianNBDV consiste au fait à parcourir tous les voisins et à sauvegarder leurs DV/DDVs dans une liste. Le processus s'arrête uniquement après le parcours du dernier voisin. Dans une deuxième étape, les vecteurs redondants sont enlevés de cette liste. Enfin, dans une troisième étape, le médian des vecteurs restants est sélectionné comme vecteur final qui sera utilisé pour IVMP. On utilise toujours NBDV pour IVRP. L'algorithme de la méthode est illustré à la Figure 11.

Dans MedianNBDV, le nombre maximal de vecteurs sur lesquels le médian est calculé (par la suite, on appellera ce nombre *MaxCand*) est de 14 vecteurs (5 voisins spatiaux * 2 liste de référence + 2 voisins temporaux dans 2 trames de référence temporelles = 14) . Or le calcul du médian de 14 vecteurs est difficile en hardware, donc pour que MedianNBDV puisse être adopté dans 3D-HEVC, *MaxCand* doit être réduit. Pour cela, nous avons pensé à 3 variantes : 1Ref où une seule liste de

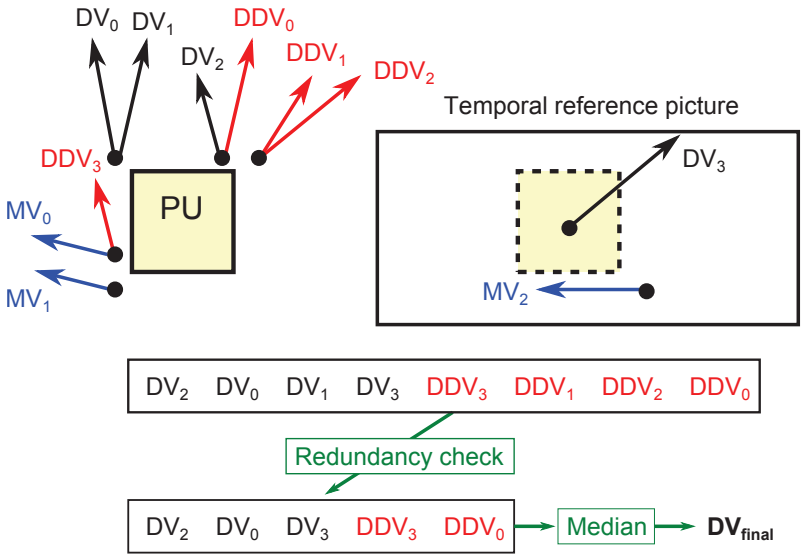


Figure 11: Algorithme de MedianNBDV

référence est parcourue pour les voisins spatiaux (cela réduit *MaxCand* à 9), RMPOS où une position spatiale du parcours est carrément enlevée (cela permet d'économiser deux vecteurs, ou un seul si combiné avec 1REF) et enfin LIMIT-X où la taille de la liste est limitée à X vecteurs (*MaxCand* = X). Par exemple, dans LIMIT-4, les voisins sont parcourus et seuls les 4 premiers vecteurs trouvés sont sauvegardés (le processus de recherche s'arrête là). Nous avons aussi pensé à d'autres variantes dont les résultats permettront d'arriver à des conclusions intéressantes : NODDV où aucun DDV n'est ajouté dans la liste, ALLOWRED où la vérification de la redondance à l'étape 2 est enlevée, NOAMVP où MedianNBDV n'est pas utilisé pour IVMP-AMVP, APPLYRES où MedianNBDV est utilisé également pour IVRP et enfin, MEAN où le calcul du médian est remplacé par une moyenne.

On a testé notre méthode dans le HTM-5.0.1 et on a respecté les CTC. On obtient -0.6% et -0.8% de gain avec MedianNBDV pour les vues dépendantes, comme le montre le Tableau 4. L'augmentation de la sélection des candidats multi-vue et inter-vue du Merge (qui dépendent directement du DV) de 1.9% et 31.1% respectivement justifie ces gains. Notons aussi que le nombre moyen de vecteurs sur lequels le médian est calculé est de 1.9 à l'encodeur et 2.2 au décodeur ce qui explique pourquoi il n'y a pas eu d'augmentation apparente du runtime avec notre méthode. Visuellement, nous pouvons souligner une différence en comparant une partie d'une trame codée avec MedianNBDV et la même partie codée avec une référence HTM-5.0.1 dans deux séquences : Kendo et Dancer (Figure 12). En effet, dans Kendo, l'épée qui était coupée en deux avec la référence est rétablie avec MedianNBDV. Dans Dancer, l'arrière de la tête du danseur est mieux représentée avec MedianNBDV qu'avec la référence.

Variante	Max Cand	Vidéo				Synt.	Codé +Synt	Runtimes	
		0	1	2	Avg			Enc	Dec
MedianNBDV	14	0.0	-0.6	-0.8	-0.3	-0.2	-0.2	100	99
1REF	9	0.0	-0.6	-0.7	-0.2	-0.2	-0.2	97	98
1REF+RMPOS	7	0.0	-0.5	-0.6	-0.2	-0.2	-0.2	99	99
LIMIT-4	4	0.0	-0.5	-0.7	-0.2	-0.2	-0.2	103	98
NODDV	14	0.0	-0.3	-0.3	-0.1	-0.1	-0.1	97	99
ALLOWRED	14	0.0	-0.2	-0.3	-0.1	-0.1	-0.1	98	98
MEAN	14	0.0	-0.3	+0.1	0.0	0.0	0.0	100	98
NOAMVP	14	0.0	-0.6	-0.7	-0.3	-0.2	-0.2	101	98
APPLYRES	14	0.0	-0.6	-0.8	-0.3	-0.2	-0.2	108	98

Table 4: Moyennes des gains de coage (en BD-Rate) avec MedianNBDV et ses différentes variantes

(a) Kendo V1 QP40 - référence (b) Kendo V1 QP40 - MedianNBDV

(c) Dancer V2 QP35 - référence (d) Dancer V2 QP35 - MedianNBDV

Figure 12: Parties de trames codées avec MedianNBDV et avec une référence HTM-5.0.1

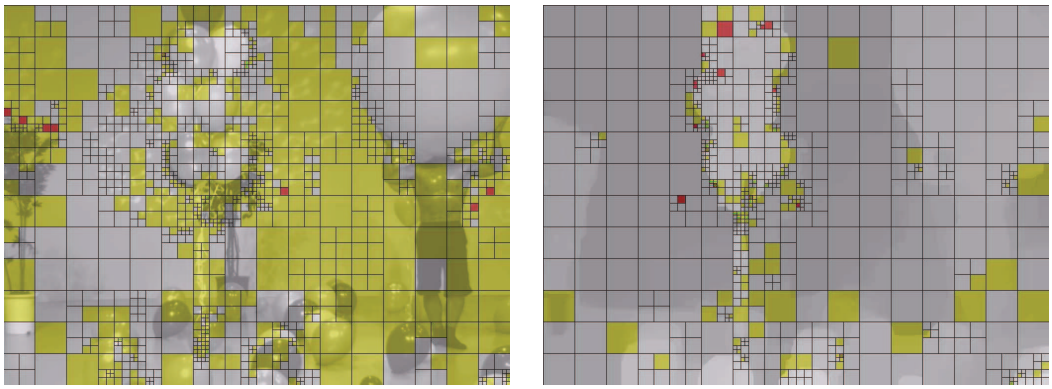
Parmi les 3 variantes que nous avons proposées pour réduire *MaxCand*, LIMIT-4 le réduit le plus moyennant une très légère perte (0.2% sur la vue 1, 0.1% sur la vue 2) c'est donc la meilleure variante dans cette catégorie. On remarque aussi que si aucun DDV n'est ajouté dans la liste, si le médian est remplacé par une moyenne, ou si la vérification de redondance est enlevée, les gains de codage diminuent significativement. Ne pas appliquer MedianNBDV pour IVMP-AMVP diminue très légèrement les gains (0.1% sur la vue 2) ce qui prouve que l'efficacité de la méthode vient plutôt de l'amélioration du mode Merge ce qui est normal vu sa sélection fréquente. Par ailleurs, utiliser MedianNBDV pour IVRP ne modifie pas les gains mais augmente sensiblement le runtime (108%) vu l'application plus fréquente d'un calcul de médian. MedianNBDV a fait le sujet d'un article accepté et présenté à la

conférence International Conference on Image Processing (ICIP) en 2013.

5 Initialisation et limitation des arbres quaternaires de texture et de profondeur dans 3D-HEVC

En analysant les arbres quaternaires des deux composantes de texture et de profondeur codées avec 3D-HEVC, on remarque que la texture est plus partitionnée en général que la profondeur. La Figure 13 illustre cette observation. En effet, la texture et la profondeur représentent la même scène à un même instant donné. Supposons qu'on ait un objet bien texturé dans la scène, les CUs comportant cet objet seront splittées dans la composante de texture. Mais puisque tous les détails de l'objet sont à la même profondeur, l'objet sera représenté dans la composante de profondeur comme une surface plate et les CUs correspondantes n'auront pas besoin d'être splittées pour obtenir une prédiction efficace. Nous formulons ainsi l'hypothèse suivante : "une CU de texture est au moins autant partitionnée qu'une CU de profondeur colocalisée".

Le Tableau 5 montre le pourcentage de CUs où cette hypothèse n'est pas vérifiée, pour diverses séquences codées à différents QPs. On en conclut que l'hypothèse est en général valide, elle l'est particulièrement pour les séquences Dancer et GT Fly qui sont des séquences générées par ordinateur et qui ne comprennent donc pas d'artefacts liés à une mauvaise estimation des cartes de profondeur. En effet, les autres séquences ont ce problème là et ce dernier se manifeste par des faux contours dans la profondeur qui n'existent pas en texture. Dans ces cas-là, les CUs de profondeur correspondantes à ces contours seront splittées alors qu'ils ne le seront pas en texture, faussant ainsi l'hypothèse. Cet effet s'estompe à bas débit comme le montre le Tableau 5.



(a) Arbre quaternaire de texture (b) Arbre quaternaire de profondeur

Figure 13: Arbre quaternaire de texture et de profondeur d'une trame Inter de la séquence Balloons codée à QP 25

Sequence	QP			
	25	**30**	**35**	**40**
Kendo	38	15	6	2
Newspaper	56	31	16	6
Balloons	40	19	7	4
Dancer	15	16	10	4
GT Fly	21	11	4	2
Poznan Hall2	32	11	4	2
Poznan Street	31	13	6	2

Table 5: Pourcentage de CUs où l'hypothèse formulée n'est pas vérifiée

Par ailleurs, les informations de partitionnement : bit signalant un split (split flag) + taille de partitionnement représentent 8.1% du débit total alloué à la texture et à la profondeur, ce qui donne une marge de gain intéressante si une prédiction efficace de ces informations était mise en place.

Par conséquent, il serait bénéfique d'imposer l'hypothèse formulée lors du codage effectif des deux composantes : si la texture est codée avant la profondeur, l'arbre quaternaire de la profondeur serait limité par celui de la texture, et dans le cas inverse, l'arbre quaternaire de la texture serait initialisé par celui de la profondeur. Dans les deux cas, on gagnerait en temps d'encodage car dans le premier, cela évitera de tester les partitions plus fines, alors que dans le second, cela évitera de tester les grandes partitions. On augmente aussi l'efficacité de codage en n'ayant plus à envoyer les informations de partitionnement pour une composante dans certains cas, car ils pourront être dérivés par l'arbre quaternaire décodé de l'autre composante.

Notre méthode n'est pas vraiment un raccourci d'encodeur car un raccourci d'encodeur cherche en général à obtenir des gains en runtime moyennant une certaine perte en efficacité de codage, à travers le choix de ne pas tester certains modes de prédiction. Dans notre méthode, on cherche à obtenir des gains en runtime mais aussi des gains de codage avec une meilleure prédiction des informations de partitionnement. Quoi qu'il en soit, il serait intéressant de comparer notre méthode aux raccourcis d'encodeur suivants, qu'on peut trouver dans le HM ou le HTM : CBF-based early termination, Early CU termination (ECU) et Enhanced Depth CU (EDCU), qui est un raccourci appliqué uniquement en profondeur.

La première méthode proposée est l'initialisation de l'arbre quaternaire de texture (QTI), où on force l'arbre quaternaire de la texture à être au moins autant partitionné que celui de la profondeur. Ceci bien sûr suppose que la profondeur est codée avant la texture dans les vues dépendantes. Si la CU de profondeur est splittée (cas 'a' dans la Figure 14), on force le split au niveau de la CU de texture colocalisée et on n'envoie pas donc de split flag. Si la CU de profondeur est partitionnée (cas 'b' et 'c'), on autorise la texture à être partitionnée de la même manière ou à être

splittée. Donc on envoie uniquement un split flag car si la CU de texture n'est pas splittée, son partitionnement en PUs est identique à celui de la profondeur (donc pas besoin de le transmettre). Si la CU de profondeur n'est ni splittée ni partitionnée (cas 'd'), la CU de texture peut être quelconque (on envoie le split flag et la taille du partitionnement). Notons que la partie de la méthode qui ne transmet pas les informations de partitionnement selon le cas est appelée codage prédictif (PC). Notons aussi que la probabilité des cas 'a', 'b' et 'c' où on applique effectivement QTI+PC est égale à 9.2%. Comme on n'applique pas QTI+PC très souvent, les gains potentiels en runtime et en codage seront limités.

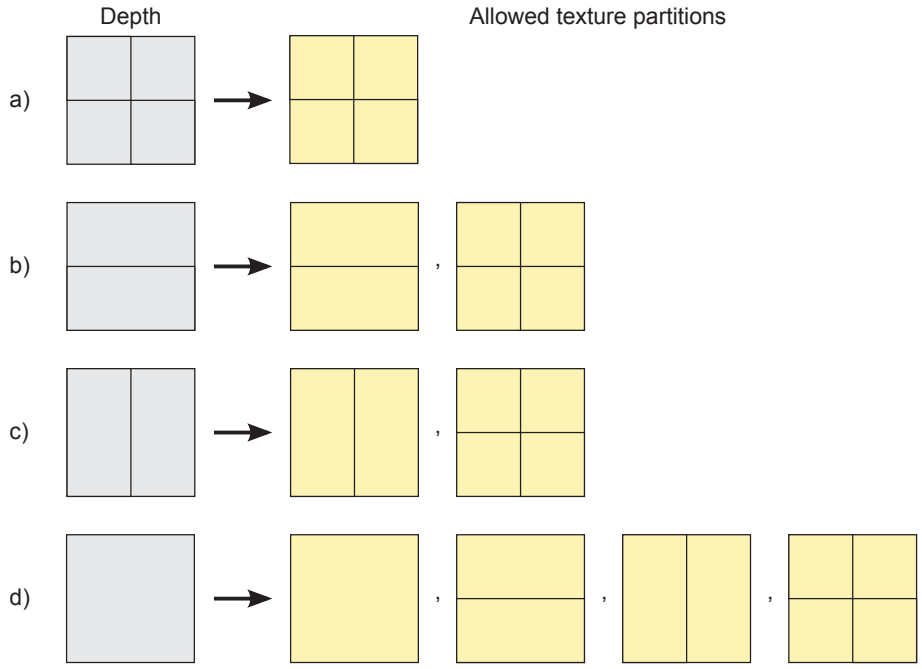


Figure 14: Partitions de texture autorisées dans QTI

Dans une variante à QTI+PC qu'on appelle QTI+1+PC, on autorise la CU de texture à être au maximum un niveau de profondeur moins partitionnée que la CU de profondeur colocalisée. C'est à dire que si la CU de profondeur est splittée au niveau N+2 comme le montre la Figure 15, on va forcer le split de la CU de texture au niveau N+1, et on n'envoie pas de split flag ni de taille de partitionnement. Sinon, dans tous les autres cas, y compris si la CU de profondeur est splittée au niveau N+1, la CU de texture peut être quelconque donc on envoie les deux infos de partitionnement. Dans cette variante un peu moins stricte que QTI+PC, on applique l'initialisation en texture moins souvent, donc on altère l'arbre quaternaire de texture moins souvent par rapport à celui choisi par RDO, diminuant ainsi les pertes de codage, mais on réduit aussi les gains du PC car on envoie les infos de partitionnement plus souvent. Ceci crée un nouveau compromis.

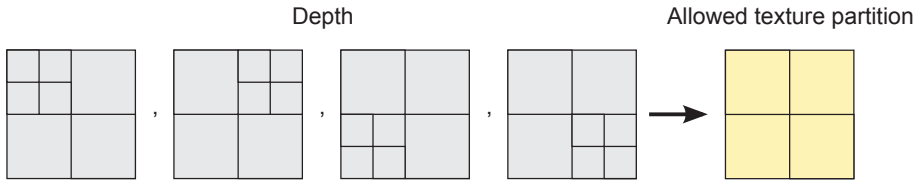


Figure 15: La variante QTI+1

La deuxième méthode proposée est la limitation de l'arbre quaternaire de la profondeur (QTL) où l'on force l'arbre quaternaire de la profondeur à être au plus autant partitionné que celui de la texture. Ceci suppose que la texture est codée avant la profondeur dans les vues dépendantes, ce qui est d'ailleurs l'ordre habituel dans 3D-HEVC. Si la CU de texture est partitionnée ou ni partitionnée ni splittée (cas 'a' dans la Figure 16), la CU de profondeur est forcée à être ni partitionnée, ni splittée (on envoie ni split flag ni taille de partitionnement). Si la CU de texture est splittée par contre, la CU de profondeur peut être quelconque (on envoie les deux infos de partitionnement). La probabilité du cas 'a' où on applique QTL+PC est égale à 84.9%. On applique donc la méthode plus souvent que QTI+PC et par conséquent, les gains en runtime et en codage seront plus importants.

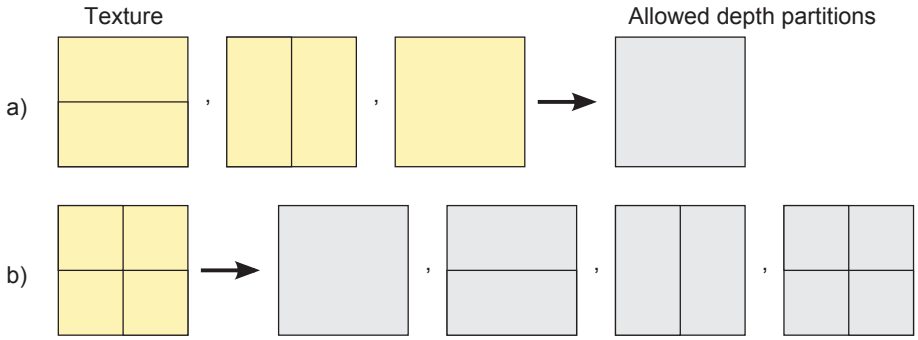


Figure 16: Partitions de profondeur autorisées dans QTL

Nous avons implémenté QTI+PC et QTI+1+PC dans le HTM-4.0, où on a activé l'outil Flexible Coding Order (FCO) pour coder les profondeurs avant les textures dans les vues dépendantes (sinon QTI+PC ne peut être appliqué). Nous avons respecté les CTC et codé toutes les trames de toutes les séquences. Le Tableau 6 résume les gains moyens obtenus avec QTI, QTI+PC, QTI+1+PC et les raccourcis d'encodeur ECU et CBF appliqués uniquement en texture (ECU-T et CBF-T). QTI donne 10% de gain en runtime moyennant une perte de 3.0% et 2.9% sur les vues dépendantes et 0.9% sur les vues synthétisées. L'ajout du PC réduit considérablement ces pertes (1.7% sur les vues dépendantes et 0.6% sur les vues synthétisées) mais réduit aussi le gain en runtime qui passe de 10% à 5%. Enfin, la variante QTI+1+PC réduit presque totalement les pertes de codage tout en maintenant un

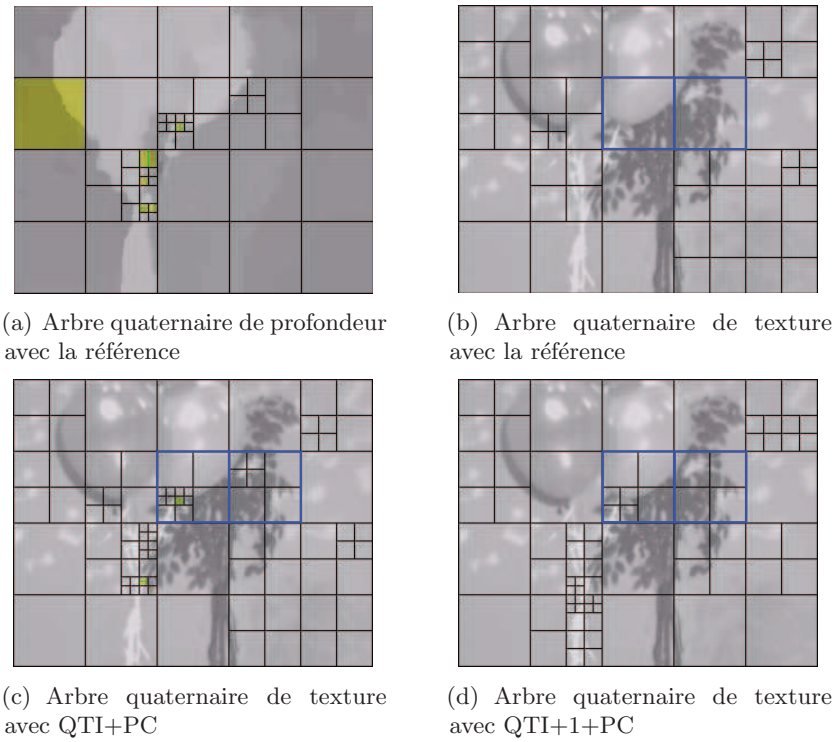
Méthode	Vidéo				Synt.	Codé +Synt	Runtimes	
	0	1	2	Avg			Enc	Dec
QTI	0.0	3.0	2.9	1.0	0.9	0.9	90	99
QTI+1	0.0	1.7	1.7	0.6	0.5	0.5	95	100
QTI+1+PC	0.0	0.1	0.2	0.0	0.1	0.0	94	101
ECU-T	0.3	-1.2	-1.1	0.5	-0.1	N/A	63	100
CBF-T	0.9	0.5	0.5	0.9	0.7		72	101

Table 6: Moyenne des gains (en BD-Rate) obtenus avec QTI, QTI+PC, QTI+1+PC, et d'autres raccourcis d'encodeur pour la texture

gain en runtime de 6%. En effet, QTI+1 est plus souple que QTI dans le sens où l'arbre quaternaire de texture est altéré moins souvent, réduisant ainsi les pertes. Les Figure 17(a) et 17(b) montrent respectivement les arbres quaternaires de texture et de profondeur d'une partie d'une trame codée avec la référence HTM-4.0. Les CU marquées en bleu en texture ne sont ni splittées ni partitionnées, c'est le meilleur choix selon RDO. Or avec QTI, ces mêmes CU sont forcées à être au moins autant partitionnées que les CU de profondeur colocalisées d'où le partitionnement fin visible à la Figure 17(c), qui induit certainement des pertes étant trop éloigné du meilleur choix. Avec QTI+1+PC, les CU de texture peuvent être un niveau de profondeur moins partitionnées que les CU de profondeur colocalisées, comme le montre la Figure 17(d). Le partionnement résultant est plus proche du meilleur choix, d'où la réduction des pertes.

Comparé à d'autres raccourcis d'encodeur, QTI+1+PC ne donne aucune perte sur les vues codées, contrairement à ECU et CBF qui donnent respectivement 0.5% et 0.9% de pertes. Par contre, les gains en runtime que donnent ces deux raccourcis d'encodeur sont supérieurs à ceux obtenus avec QTI+1+PC.

Nous avons implémenté QTL+PC dans le HTM-4.0 également. Nous avons respecté les CTC et codé toutes les trames de toutes les séquences. Les résultats obtenus sont résumés dans le Tableau 7. QTL donne des gains de codage significatifs (-20.1% sur les vidéos de profondeur, et -1.4% sur les vues codées) ainsi que des gains en runtime assez significatifs également de 34%. Ceci est accompagné d'une perte sur les vues synthétisées de 0.7%. QTL+PC augmente les gains de codage sur les vidéos de profondeur et sur les vues codées, diminue les pertes sur les vues synthétisées, mais réduit aussi légèrement le gain en runtime qui passe à 31%. Il est important de noter que les pertes sur les vues synthétisées ne sont pas toutes de vraies pertes. Une partie de ces pertes est liée au lissage de faux contours dans la profondeur, qui est en réalité un avantage apporté par la méthode. En effet, considérons l'exemple de la Figure 18. Cette figure montre une partie d'une trame de texture et de profondeur. Nous remarquons des faux contours en profondeur, marqués en



(a) Arbre quaternaire de profondeur avec la référence

(b) Arbre quaternaire de texture avec la référence

(c) Arbre quaternaire de texture avec QTI+PC

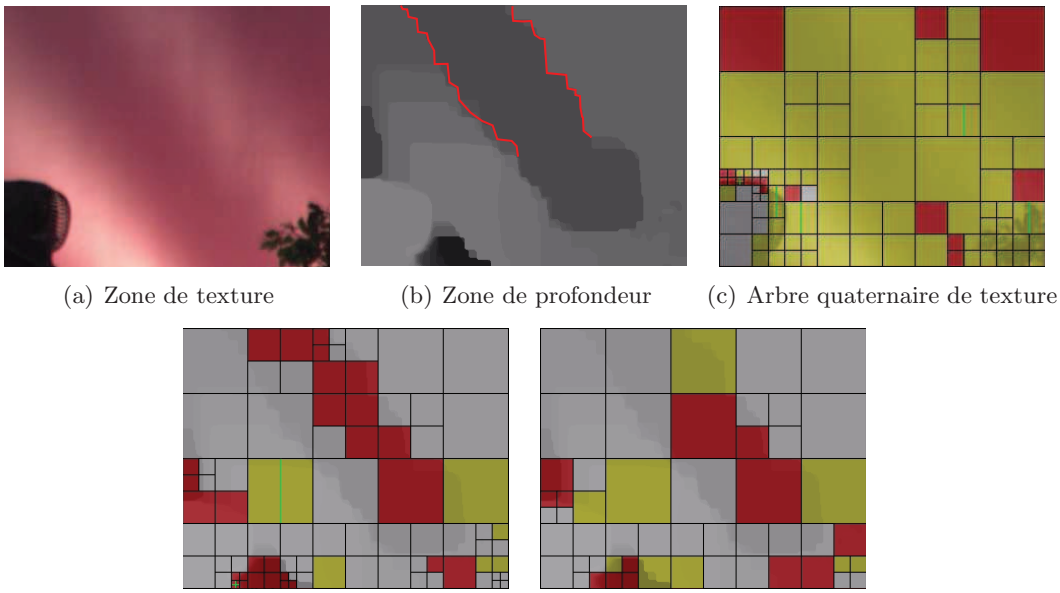
(d) Arbre quaternaire de texture avec QTI+1+PC

Figure 17: Arbre quaternaire de texture avec la référence HTM-4.0, avec QTI+PC, et avec QTI+1+PC

rouge à la Figure 18(b) qui n'existent pas en texture. Les CU correspondantes ne seront pas splittées en texture, comme le montre la Figure 18(c) mais le seront en profondeur (Figure 18(d)) avec un codage de référence. Avec QTL, les CU de profondeur sont limitées aux CUs de texture. Elles ne sont ni splittées ni partitionnées, comme le montre la Figure 18(e), lissant ainsi les faux contours sous-jacents. Ceci est bénéfique car des vues synthétisées avec ces cartes de profondeur plus "propres" seront d'une meilleure qualité. En effet, lors du 2$^{\text{ème}}$ meeting JCT-3V, des tests subjectifs ont été effectués et leurs résultats prouvent qu'avec QTL+PC, il n'y a aucune dégradation visible sur les vues synthétisées. Il y a même au contraire une très légère amélioration. Enfin, comparé à d'autres raccourcis d'encodeur, QTL+PC donne le plus de gains de codage et le plus de gains en runtime. QTL+PC a été présenté au 2$^{\text{ème}}$ meeting JCT-3V et a été adopté dans le texte et dans le logiciel de référence de 3D-HEVC. QTI+PC et QTL+PC ont été les sujets d'un article de journal publié dans IEEE Transactions on Circuits and Systems for Video Technology (CSVT) en 2013.

Méthode	Profondeur	Vidéo total	Synt.	Codé +Synt	Runtimes	
					Enc	Dec
QTL	-20.1	-1.4	0.7	0	66	97
QTL+PC	-23.1	-1.8	0.4	-0.3	69	97
ECU-D	N/A	-2.2	1.1	N/A	76	N/A
EDCU		-1.4	0.4		82	
CBF-D		-0.2	0.3		85	

Table 7: Moyenne des gains (en BD-Rate) obtenus avec QTL, QTL+PC et d'autres raccourcis d'encodeur pour la profondeur



(a) Zone de texture (b) Zone de profondeur (c) Arbre quaternaire de texture

(d) Arbre quaternaire de profondeur avec la référence (e) Arbre quaternaire de profondeur avec QTL

Figure 18: Arbres quaternaires de profondeur avec la référence et avec QTL

6 Codage vidéo 3D en utilisant le flot optique warpé

Le flot optique (OF) est une méthode utilisée pour dériver un champ dense de vecteurs de mouvement (DMVF) entre deux trames. Le DMVF réduit davantage l'énergie résiduelle en comparaison avec un champ de vecteurs de mouvement par blocs estimé par le HTM (CMVF). Le Tableau 8 montre en effet que l'énergie résiduelle avec le CMVF (E_2^C) est supérieure à celle obtenue avec le DMVF (E_2^D) et ce pour toutes les séquences et QPs testés.

Par ailleurs, le DMVF estimé entre deux trames reconstruites (pour simuler une estimation de mouvement pouvant être réalisée au décodeur) reste toujours meilleur qu'un CMVF classique estimé à l'encodeur entre une trame originale et une autre reconstruite. En effet, le Tableau 8 montre que l'énergie résiduelle avec ce DMVF bruité par le bruit de quantification (E_2^{DQ}) est inférieure à E_2^C pour toutes

les séquences et QPs testés, même à bas débit (bruit de quantification élevé). C'est sur cette observation que va se baser l'approche proposée dans ce chapitre.

Séquence	**QP**	$\mathbf{E_2^C}$	$\mathbf{E_2^D}$	$\mathbf{E_2^{DQ}}$
Balloons	25	5.7	4.1	4.2
	30	6.3	4.8	5.0
	35	8.1	6.3	6.9
	40	11.8	8.9	10.3
Kendo	25	28.1	8.5	8.8
	30	15.8	8.7	9.3
	35	16.6	9.3	10.2
	40	17.9	10.0	12.1
Newspaper	25	8.6	5.8	6.0
	30	9.5	7.6	8.0
	35	12.8	10.5	11.4
	40	18.0	15.0	16.8
PoznanHall2	25	9.3	6.3	6.4
	30	8.2	5.9	6.2
	35	7.3	6.3	6.8
	40	8.3	7.2	7.9

Table 8: Energie résiduelle par méthode d'estimation de mouvement pour 4 séquences et 4 QPs testés

Dans la littérature, l'OF est en général utilisé dans des applications d'indexation vidéo, de détection / reconnaissance d'objets dans des séquences vidéo. L'OF a rarement été proposé pour le codage vidéo à cause du coût élevé lié à la transmission des multiples vecteurs de mouvement qui composent le DMVF. Quelques solutions ont été proposées pour réduire ce coût mais aucune n'a été suffisamment efficace.

Par ailleurs, les approches décodage intelligent permettent de réduire le coût de signalisation de certaines informations transmises dans le flux binaire en déplaçant quelques opérations au décodeur. Des exemples d'approches décodage intelligent incluent le template matching ou la réduction du nombre de prédicteurs. L'idée proposée dans ce chapitre est d'utiliser une approche décodage intelligent conjointement avec l'OF pour coder les vues dépendantes dans 3D-HEVC, en évitant de transmettre les vecteurs qui constituent le DMVF. Concrètement, cela consiste à estimer le DMVF sur une vue de base reconstruite au décodeur, et l'utiliser pour coder les vues dépendantes.

La Figure 19 illustre notre méthode, qu'on appelle Warped Optical Flow (WOF). On suppose qu'on désire coder une vue dépendante courante à l'instant t, ayant déjà codé la vue de base aux instants $t-1$, t et $t+1$, et la vue dépendante aux instants $t-1$ et $t+1$. Dans une première étape, avant le début du codage de la vue dépendante courante, deux DMVFs sont calculés au niveau de la vue de

base entre les trames reconstruites à l'instant t et $t-1$ pour l'un, et t et $t+1$ pour l'autre. Dans une deuxième étape, en codant une PU courante dans la vue dépendante courante, un DV est dérivé en utilisant NBDV qui pointe vers une PU correspondante à la PU courante dans la vue de base à l'instant t. Dans une troisième étape, les deux DMVFs de cette PU de référence sont hérités pour la PU courante, où ils seront ajoutés comme un candidat dense du Merge à la première position de la liste. L'avantage de cette méthode c'est que les PUs de la vue dépendante courante vont bénéficier de prédictions plus précises apportées par les DMVFs hérités, au prix d'une simple transmission d'un indice du Merge. Le deuxième avantage est que comme avec les DMVFs, chaque pixel d'une PU a son propre MV, la CU n'a plus besoin d'être splittée pour obtenir une prédiction plus efficace. Un gain sur les informations de partitionnement est par conséquent possible avec WOF.

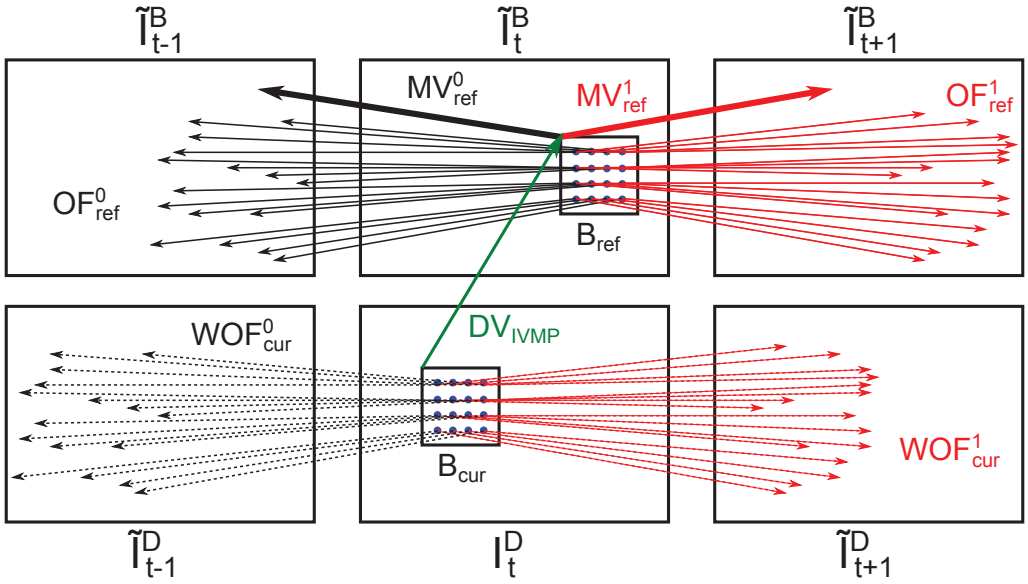


Figure 19: Méthode proposée pour dériver un candidat dense du Merge

Dans WOF, un seul vecteur de disparité estimé avec NBDV compense la PU courante, même si celle-ci contient deux objets de profondeurs différentes. Ceci à son tour faussera l'association des MVs aux pixels de la PU courante et la prédiction ne sera plus efficace. Pour résoudre ce problème, nous proposons deux variantes : Flexible Coding Order WOF (FCO-WOF) et Depth Oriented WOF (DO-WOF). FCO-WOF suppose qu'on code la profondeur avant la texture dans les vues dépendantes. Dans ce cas, au lieu de compenser la PU courante avec un seul DV, chaque pixel de la PU va être compensé avec un DV calculé à partir de la valeur du pixel colocalisé dans la trame de profondeur déjà codée, comme le montre la Figure 20. La variante DO-WOF suppose qu'on code la texture avant la profondeur dans les vues dépendantes. Dans ce cas, le DV donné par NBDV va être utilisé pour pointer

à la PU correspondante dans la trame de profondeur déjà codée associée à la vue de base. Chaque valeur de profondeur dans cette PU de référence est transformée en un vecteur de disparité qui va compenser le pixel correspondant dans la PU courante, comme le montre la Figure 21.

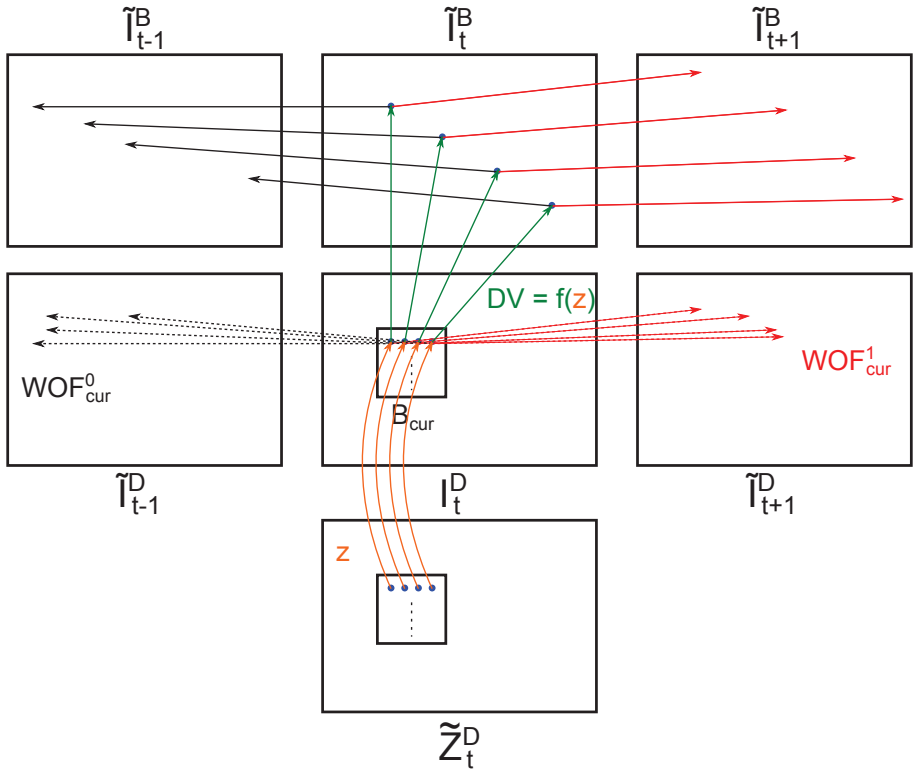


Figure 20: La variante FCO-WOF

Nous avons implémenté WOF, FCO-WOF et DO-WOF dans le HTM-5.1. Nous avons respecté les CTC et codé la moitié des trames de toutes les séquences pour accélérer les simulations (5 secondes de vidéo). Le Tableau 9 montre les résultats obtenus avec WOF. On note des gains significatifs de -7.3% et -6.9% sur les vues dépendantes et -2.5% sur les vues synthétisées. Les gains vont jusqu'à -21.1% sur une vue dépendante de la séquence GT Fly. Ces gains sont accompagnés d'une augmentation du runtime à l'encodeur de 33% et de 11487% au décodeur. L'augmentation est accrue au décodeur car ce dernier est passé d'un décodeur passif qui lit simplement des informations du flux binaire à un décodeur actif qui estime des DMVFs, une opération qui reste assez complexe.

Les gains de codage avec WOF proviennent d'une réduction de l'énergie résiduelle bien sûr, mais aussi d'une réduction du nombre de split et de skip flags transmis dans le flux binaire de 38% et 44% respectivement, comme le montre le Tableau 10.

Le Tableau 11 résume les résultats des variantes DO-WOF et FCO-WOF et les

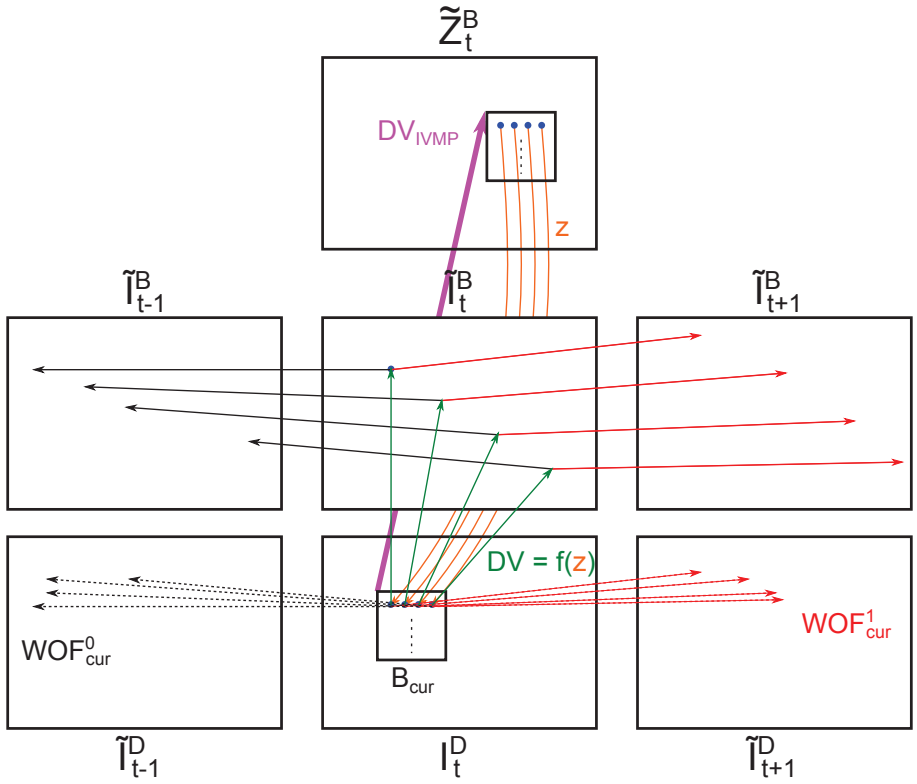


Figure 21: La variante DO-WOF

compare à ceux obtenus avec WOF. On remarque que si les résultats de WOF et DO-WOF sont similaires, FCO-WOF donne légèrement plus de gains avec -7.5% et -7.6% en moyenne sur les vues dépendantes et -2.7% sur les vues synthétisées. Ceci est dû à la compensation dense en disparité, plus précise que la compensation d'un bloc entier avec un seul DV.

Le Tableau 12 montre les résultats obtenus avec WOF à bas débit, à moyen débit et à haut débit. On remarque que les gains diminuent quand le débit augmente. En effet, le Tableau 13 montre que le candidat dense du Merge ajouté dans WOF est plus sélectionné par les PUs à bas débit (grands QPs). Ceci est dû au fait que le mode Merge est lui-même plus sélectionné à bas débit.

Par ailleurs, le runtime peut être réduit par plusieurs méthodes. Premièrement, certains paramètres de l'OF peuvent être modifiés sans réelle influence sur la performance de codage. Le Tableau 14 montre par exemple qu'en diminuant la valeur d'un des paramètres (le ratio), les gains de codage restent pratiquement inchangés, alors que le runtime à l'encodeur et au décodeur diminue significativement. Deuxièmement, le code de l'OF peut être optimisé pour réduire le runtime en évitant d'utiliser des structures de données complexes par exemple. Finalement, un calcul OF par PU et non par trame peut être mis en place car nos expériences montrent qu'en moyenne, seuls 68% des pixels de la vue de base sont référencés par les PUs

Séquence	Vidéo				Vidéo total	Synt.	Runtimes	
	0	1	2	Moy			Enc	Dec
Balloons	0.0	-7.8	-7.0	-3.2	-3.1	-3.1	127	10514
Kendo	0.0	-5.4	-4.5	-1.9	-1.8	-2.0	126	11417
Newspaper	0.0	-3.1	-2.6	-1.1	-1.0	-1.0	141	12968
GT Fly	0.0	-21.1	-20.4	-6.0	-5.7	-5.9	135	11160
PoznanHall2	0.0	-7.9	-9.4	-3.8	-3.7	-3.7	124	11515
PoznanStreet	0.0	-3.6	-4.2	-1.1	-1.1	-1.4	142	12159
Dancer	0.0	-1.9	-0.4	-0.5	-0.4	-0.5	136	11373
Moyenne	**0.0**	**-7.3**	**-6.9**	**-2.5**	**-2.4**	**-2.5**	**133**	**11587**

Table 9: Gains de codage (en BD-Rate) avec WOF

Sequence	WOF	
	RSP	RSK
Balloons	40	47
Kendo	28	32
Newspaper	29	33
GT Fly	66	76
PoznanHall2	40	42
PoznanStreet	39	48
Dancer	23	28
Average	**38**	**44**

Table 10: Réduction du nombre de split flags (RSP) et de skip flags (RSK) envoyés dans le flux binaire avec WOF

des vues dépendantes. Cela veut dire que des MVs estimés pour 32% des pixels de la vue de base ne sont pas utilisés. Un gain en runtime est alors possible en évitant de les estimer en premier lieu.

Méthode	Vidéo				Vidéo total	Synt.	Runtimes	
	0	1	2	Moy			Enc	Dec
WOF	0.0	-7.3	-6.9	-2.5	-2.4	-2.5	133	11587
DO-WOF	0.0	-7.1	-7.2	-2.5	-2.5	-2.5	136	11968
FCO-WOF	0.0	-7.5	-7.6	-2.7	-2.4	-2.7	139	11022

Table 11: Moyenne des gains de codage (en BD-Rate) avec WOF, DO-WOF et FCO-WOF

Test	Vidéo				Vidéo total	Synt.
	0	1	2	Avg		
Bas débit	0.0	-5.9	-5.8	-2.0	-2.0	-2.0
Moyen débit	0.0	-4.8	-4.4	-1.5	-1.5	-1.6
Haut débit	0.0	-2.7	-2.3	-0.8	-0.8	-0.8

Table 12: Moyenne des gains de codage (en BD-Rate) obtenus avec WOF à bas, moyen et haut débit

Séquence	Pourcentage de sélection					
	20	25	30	35	40	45
Balloons	36	67	82	90	93	96
Kendo	24	36	49	59	70	78
Newspaper	36	63	80	89	96	98
GT Fly	18	33	81	90	93	94
PoznanHall2	18	55	76	85	90	95
PoznanStreet	19	36	51	66	79	89
Dancer	8	16	37	67	81	89
Average	**23**	**44**	**65**	**78**	**86**	**91**

Table 13: Pourcentage de sélection du candidat dense du Merge par séquence et par QP

Ratio	Vidéo				Vidéo total	Synt.	Runtimes	
	0	1	2	Moy			Enc	Dec
$R = 0.85$	0.0	-4.7	-4.3	-1.5	-1.5	-1.6	275	52355
$R = 0.75$	0.0	-4.8	-4.4	-1.5	-1.5	-1.6	187	29142
$R = 0.65$	0.0	-5.0	-4.6	-1.6	-1.6	-1.5	175	22294
$R = 0.5$	0.0	-4.9	-4.6	-1.6	-1.6	-1.5	138	12023
$R = 0.4$	0.0	-4.9	-4.5	-1.6	-1.5	-1.5	134	10136

Table 14: Moyennes des gains de codage obtenus avec WOF pour différentes valeurs de ratio

7 Synthèse de vues exploitant les prédictions temporelles

Plusieurs méthodes existent pour synthétiser des vues à partir de cartes de profondeur. Certaines sont implémentées dans les logiciels de référence : View Synthesis Reference Software (VSRS) et VSRS-1DFast (utilisé pour la synthèse des vues dans le contexte de la normalisation 3D-HEVC). D'autres sont proposées dans la littérature (par exemple, les méthodes qui tentent de réduire les trous en prétraitant les cartes de profondeur ou en post-traitant les vues synthétisées directement).

Quoi qu'il en soit, toutes ces méthodes synthétisent une vue centrale à partir de deux vues de référence (gauche et droite). La synthèse exploite ainsi les corrélations inter-vues. L'idée proposée dans ce chapitre est d'utiliser les trames de référence temporelles pour synthétiser une trame centrale pour augmenter la qualité de la synthèse finale. Ce sont les corrélations temporelles qui seront donc exploitées dans notre méthode qu'on appelle View Synthesis Temporal Prediction (VSTP).

Dans VSTP, on suppose que les vues de référence gauche et droite et leurs vidéos de profondeur associées, ainsi que les deux trames de la vue synthétisée aux instants $t-1$ et $t+1$ sont disponibles. Pour synthétiser la trame à l'instant t, on calcule d'abord un DMVF avec l'OF au niveau de la vue de référence entre t et $t-1$, qu'on appelle $\mathbf{V}_r$. Ayant les cartes de profondeur de la vue de référence à l'instant $t-1$ et t on peut aussi calculer des champs de vecteurs de disparité $\mathbf{D}_{t-1}$ et $\mathbf{D}_t$ en transformant les valeurs de profondeur en disparité. Ayant $\mathbf{V}_r$, $\mathbf{D}_{t-1}$ et $\mathbf{D}_t$, un DMVF noté $\mathbf{V}_s$ liant, au niveau de la vue synthétisée, la trame à l'instant $t-1$ à la trame courante peut être calculé en utilisant une contrainte épipolaire. Illustrée à la Figure 22, la contrainte épipolaire stipule qu'à partir d'un point S dans la vue de référence à l'instant $t-1$, on peut atteindre la projection de ce point dans la vue synthétisée à l'instant t par deux chemins différents, l'un en compensant en disparité puis en mouvement et l'autre en compensant en mouvement puis en disparité. Ceci permet d'écrire l'équation suivante :

$$\mathbf{V}_r(S) + \mathbf{D}_t(S + \mathbf{V}_r(S)) = \mathbf{D}_{t-1}(S) + \mathbf{V}_s(S + \mathbf{D}_{t-1}(S)) \quad (1)$$

En utilisant $\mathbf{V}_s$, une compensation en mouvement de la trame synthétisée à l'instant $t-1$ nous donnera une première prédiction de la trame courante. Le même processus peut être répété pour une autre vue de référence. Le DMVF peut aussi être calculé entre les instants t et $t+1$ (trame de référence temporelle future au lieu de passée). Ceci donne au final 4 prédictions comme le montre la Figure 23 : passé / gauche, passé / droite, futur / gauche, futur / droite, qui seront ensuite combinées en une seule trame synthétisée. Les éventuels trous dans cette trame sont remplis

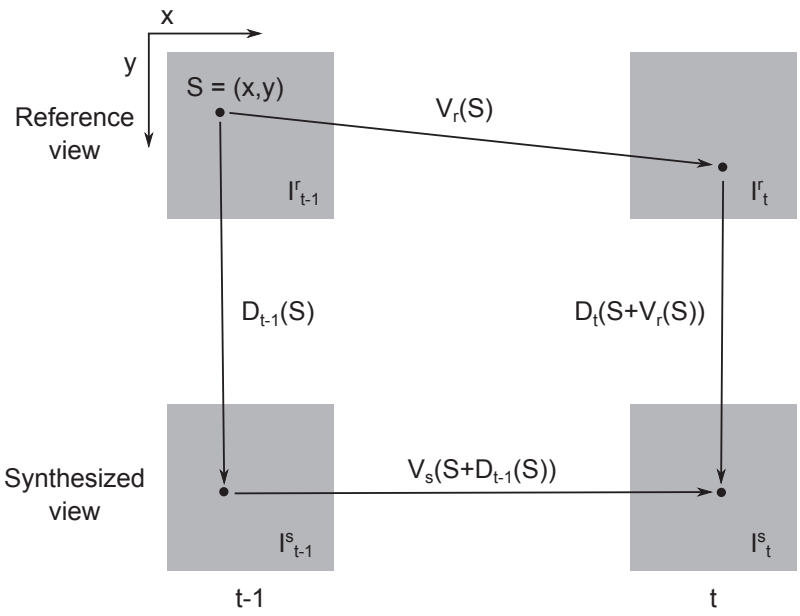


Figure 22: Contrainte épipolaire

avec un inpainting linéaire (le même que celui utilisé dans VSRS-1DFast).

Notre algorithme a besoin d'être initialisé en envoyant certaines trames, qu'on appelle trames clés, de la vue synthétisée dans le flux binaire. Les trames clés correspondent dans VSTP à la première trame de chaque GOP, le GOP étant le même que celui utilisé pour coder les vues de référence, comme le montre la Figure 24. Les autres trames sont synthétisées avec VSTP selon deux schémas de prédiction, illustrés à la Figure 25 : un schéma direct où les trames de référence temporelles sont uniquement des trames clés, et un autre schéma hiérarchique où les trames de référence temporelles peuvent être des trames précédemment synthétisées au sein d'un même GOP. L'avantage du schéma hiérarchique est que la distance de prédiction moyenne est réduite.

Nous avons implémenté VSTP dans MATLAB. Les deux vues de référence ont été codées avec le HTM-7.0. Nous avons respecté les CTC et codé seulement 3 secondes de 4 séquences vidéo pour accélérer les simulations. Nous nous sommes comparés à VSRS-1DFast pour évaluer VSTP.

Le Tableau 15 donne les gains en dB, mesurés avec la métrique de Bjontegaard BD-PSNR obtenus avec VSTP dans les deux schémas de prédiction proposés. En moyenne, VSTP augmente le PSNR de 0.717 dB et de 1.391 dB dans les schémas Direct et Hiérarchique respectivement. Les courbes RD correspondantes à chacune des 4 séquences sont données à la Figure 26.

Le Tableau 15 et les courbes de la Figure 26 montrent que le schéma Hiérarchique dépasse en performances le schéma Direct. Ceci est dû au fait que les distances de prédiction temporelles sont plus courtes avec le schéma Hiérarchique, augmentant

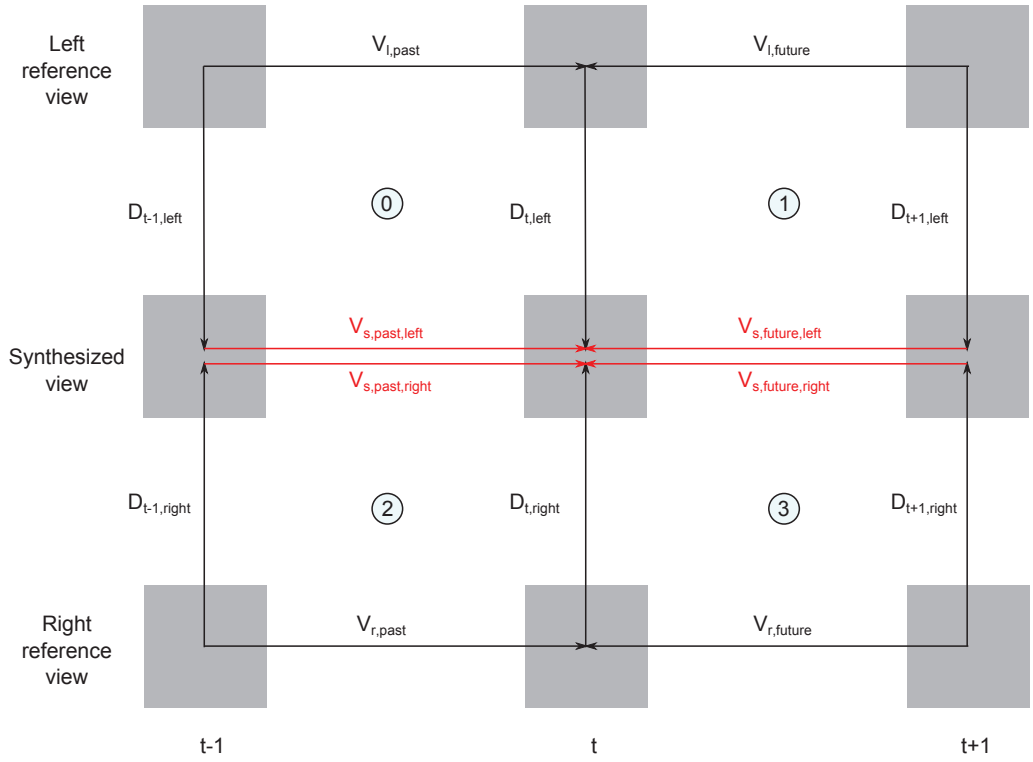


Figure 23: Quatre prédictions en utilisant la contrainte épipolaire

ainsi la qualité des 4 prédictions qui seront combinées.

En général, VSTP donne des gains par rapport à VSRS-1DFast car la méthode permet d'atteindre le bon niveau d'illumination en envoyant les trames clés, et de le maintenir pour toutes les trames de la vue synthétisée. Par ailleurs, la combinaison de 4 prédictions réduit significativement le nombre de trous à remplir car il est fort probable que des trous dans une prédiction soient disponibles dans une autre.

La Figure 27 montre l'évolution au fil du temps du PSNR da la vue synthétisée, pour la référence et pour VSTP dans les deux schémas de prédiction. On remarque que même si VSTP est en général meilleur que VSRS-1DFast, sur certaines trames,

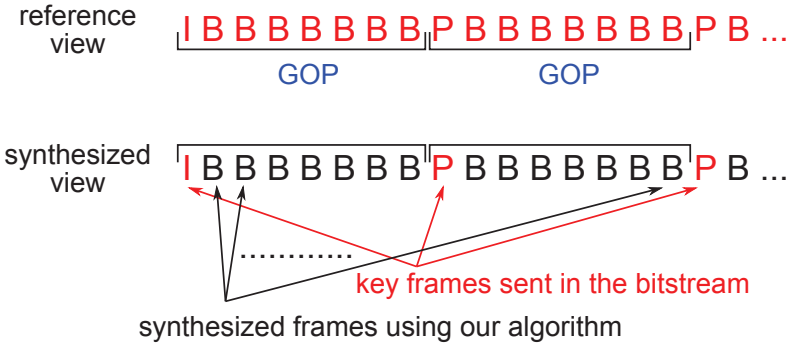
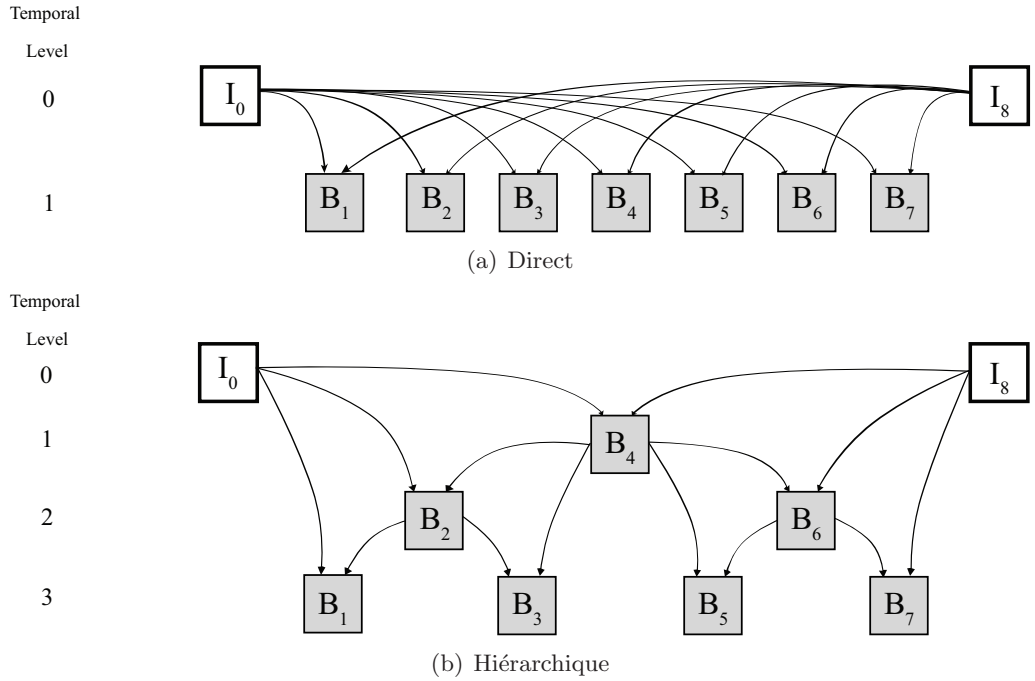


Figure 24: Synthèse par GOP dans notre algorithme



(a) Direct

(b) Hiérarchique

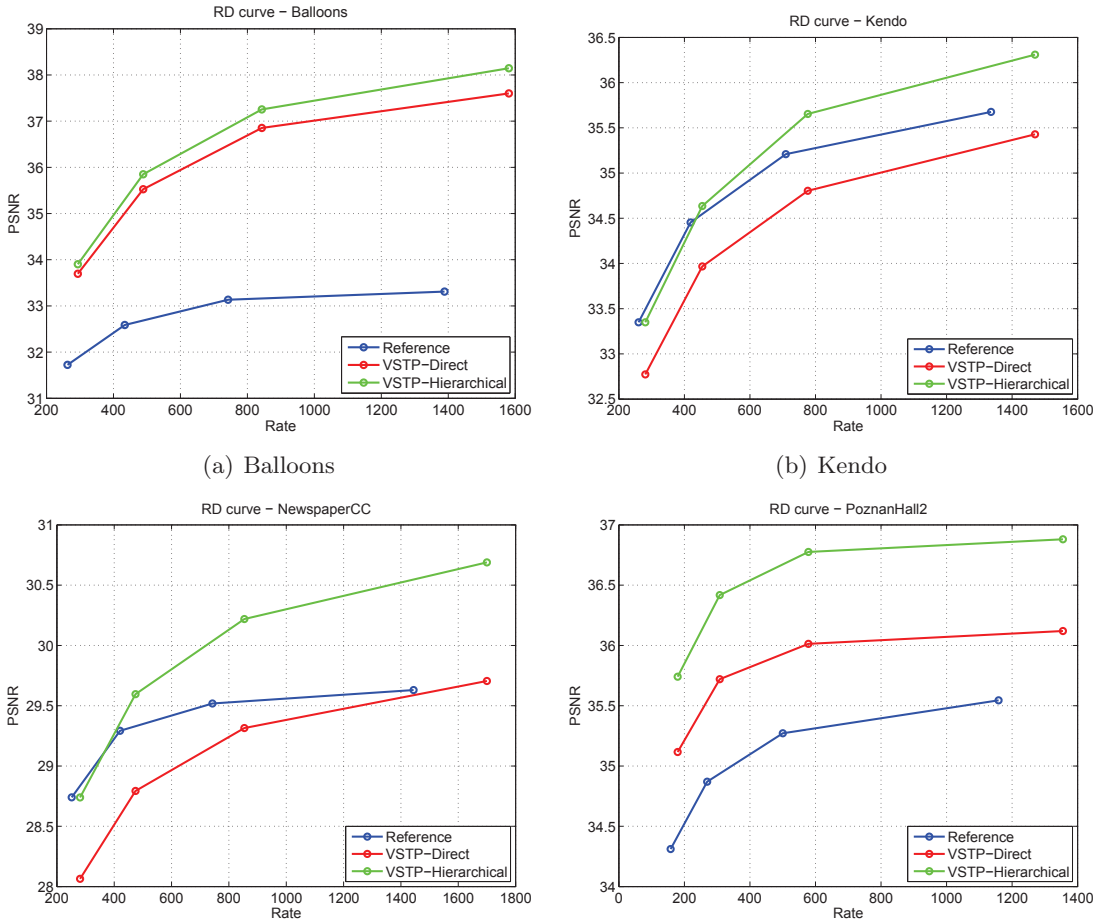
Figure 25: Schémas de prédiction dans un GOP

Séquence	**BD-PSNR (en dB)**	
	Direct	Hiérarchique
Balloons	3.150	3.511
Kendo	-0.556	0.205
Newspaper	-0.404	0.442
PoznanHall2	0.678	1.406
Moyenne	**0.717**	**1.391**

Table 15: Gains en BD-PSNR obtenus avec VSTP dans les deux schémas de prédiction proposés

c'est l'inverse. Ceci est dû à l'échec de l'OF à donner un bon DMVF lorsqu'il y a des mouvements rapides dans la vidéo. Le problème peut être résolu en modifiant les paramètres de l'OF pour rendre le calcul du DMVF plus précis, au prix par contre d'une augmentation de la complexité générale de VSTP.

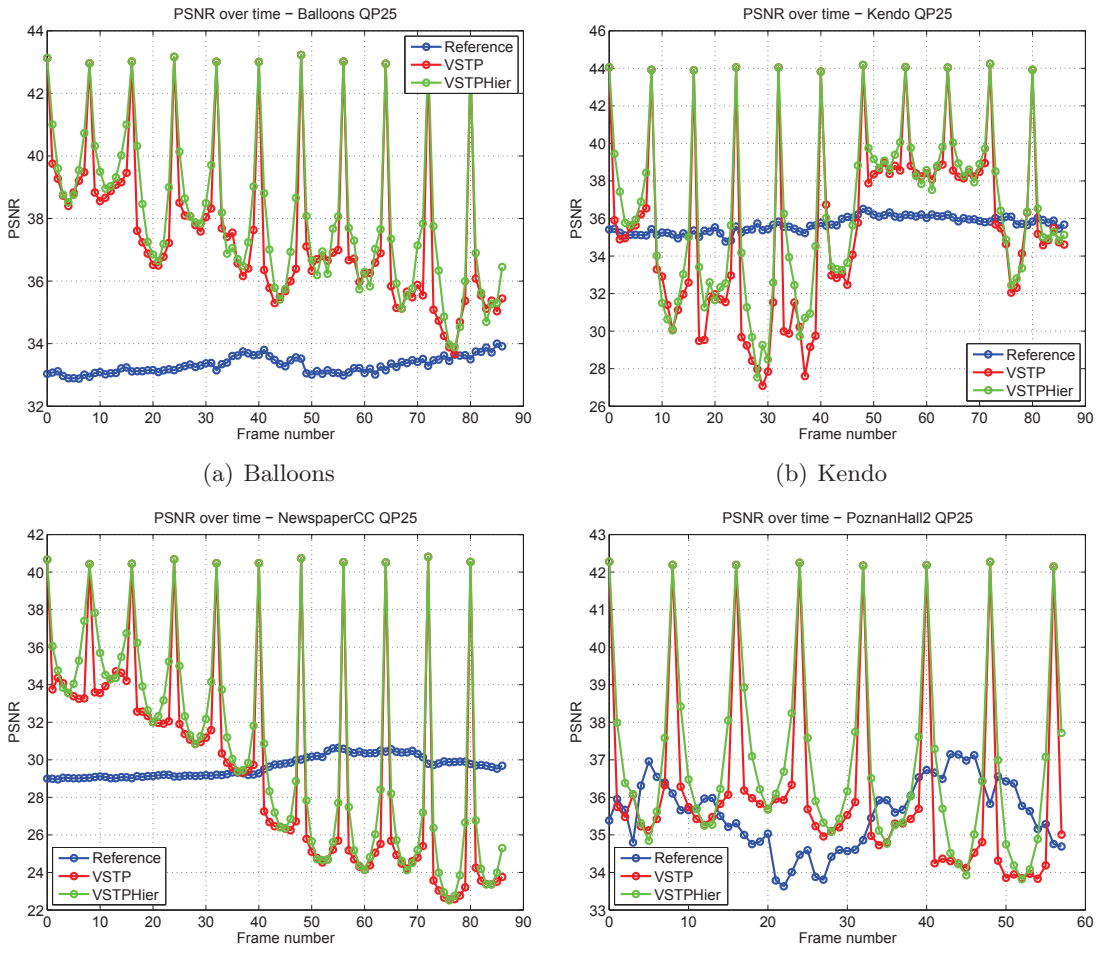
Même s'il est difficile de comparer la complexité de VSTP et de VSRS-1DFast car implémentés dans deux langages différents (MATLAB pour l'un, C++ pour l'autre), il est clair que VSTP est plus complexe que VSRS-1DFast à cause de l'estimation et de la compensation de mouvement denses qui doivent être effectuées 4 fois par trame. Pour réduire la complexité de VSTP, moins de prédictions peuvent être utilisées (par exemple prendre uniquement futur / gauche et futur / droite). Une estimation de mouvement par bloc beaucoup moins complexe qu'une estimation de mouvement dense peut aussi être mise en place.



(a) Balloons (b) Kendo

(c) Newspaper (d) PoznanHall2

Figure 26: Courbes RD de la référence et de VSTP dans les deux schémas de prédiction proposés pour les 4 séquences testées



(a) Balloons (b) Kendo

(c) Newspaper (d) PoznanHall2

Figure 27: Evolution du PSNR de la vue synthétisée au fil du temps pour la référence et la méthode proposée à QP 25

8 Conclusion

Dans ces travaux de thèse, nous avons étudié différents aspects du modèle 3D-HEVC. La prédiction de différentes informations transmises dans un flux binaire 3D-HEVC a été améliorée, donnant ainsi des gains de codage. Plusieurs méthodes conventionnelles qui suivent des contraintes strictes ont été proposées : l'héritage des modes Intra de texture pour la profondeur, AIMC, MedianNBDV, QTI / QTL + PC. Deux approches en rupture basées sur le flot optique ont aussi été proposées : WOF et VSTP. Les gains significatifs obtenus pour ces deux méthodes prouvent le potentiel de ces solutions innovantes.

Dans le futur, nous développerons une méthode adaptative VSTP / VSRS afin de sélectionner VSRS dans les zones à fort mouvement où VSTP échoue. Pour l'héritage des modes Intra de texture pour la profondeur, un schéma d'héritage progressif peut être implémenté, où le mode Intra de texture est directement utilisé pour coder la PU de profondeur si la dépendance statistique entre les deux modes est jugée importante. Dans AIMC, l'ajout du nouveau candidat peut être conditionné par un critère qui va en prédire le bénéfice. Pour MedianNBDV, l'ensemble des DVs sauvegardés dans la liste peuvent être évalués par RDO et l'indice du meilleur DV pour le bloc courant peut être signalé au décodeur. Pour QTI, nous pouvons trouver un schéma moins sévère que QTI mais plus sévère que QTI+1 afin que le PC ait plus de poids. Pour QTL, nous pouvons étudier son intéraction avec AMP qui a été récemment activé dans les CTC. Enfin, pour WOF, nous pouvons utiliser une contrainte épipolaire pour corriger le DMVF hérité, comme dans VSTP.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3D-HEVC	Three-dimensional High Efficiency Video Coding
3DTV	Three-dimensional Television
AIMC	Additional Inter-View Candidate
AMVP	Advanced Motion Vector Prediction
AVC	Advanced Video Coding
BD	Bjontegaard Delta
CfP	Call for Proposals
CTC	Common Test Conditions
CTU	Coding Tree Unit
CU	Coding Unit
dB	Decibel
DCP	Disparity-Compensated Prediction
DIBR	Depth Image Based Rendering
DMM	Depth Modeling Modes
DMVF	Dense Motion Vector Field
DO-WOF	Depth Oriented Warped Optical Flow
DV	Disparity vector
FCO	Flexible Coding Order
FTV	Free viewpoint Television
GOP	Group Of Pictures

HEVC	High Efficiency Video Coding
HTM	High Test Model (reference software of 3D-HEVC)
IVMP	Inter-View Motion Prediction
IVRP	Inter-View Residual Prediction
JCT-3V	Joint Collaborative Team on 3D Video
MCP	Motion-Compensated Prediction
MPI	Motion Parameter Inheritance
MPM	Most Probable Mode
MV	Motion Vector
MVD	Multiview Video plus Depth
NBDV	Neighboring Disparity Vector
OF	Optical Flow
PC	Predictive Coding
PSNR	Peak Signal to Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
QTI	Quadtree Initialization
QTL	Quadtree Limitation
RD	Rate-Distortion
RDO	Rate-Distortion Optimization
VSO	View Synthesis Optimization
VSP	View Synthesis Prediction
VSRS	View Synthesis Reference Software
VSTP	View Synthesis exploiting Temporal Predictions
WOF	Warped Optical Flow

Introduction

Context

The invention of the stereoscope by Sir Charles Wheatstone in 1833 marked the first milestone in the history of 3D video. The device consisted in a pair of mirrors oriented at a 45° angle to the viewer's eyes, reflecting two images depicting the same scene but from a slightly different view point. This resulted in a compelling sense of 3D perception of the scene. In the beginning of the 20^{th} century, different stereoscopic viewing techniques were progressively introduced such as the anaglyph glasses, the shutter based technique and the polarized projection. The 1950s are considered as the golden era of 3D movies. Indeed, Hollywood invested in 3D back then to counter the dropping box office receipts of 2D movies due to the ever-increasing popularity of a competing technology: the television. However, this outbreak was short-due because 3D viewing was uncomfortable for the cinema audience at that time. Since then, 3D has come and gone in certain "waves". In 2004, the release of the animation movie "Polar Express" and later, in 2009, the blockbuster "Avatar" marked a significant comeback of 3D.

Over the years, the cinema audience became more and more demanding, expecting higher picture resolutions, more vibrant colors, multi-sensorial interaction, animations, and finally depth perception, because it exists in real life. This is the reason behind these periodic outbursts in 3D cinema history. Indeed, 3D cinema has the ability to generate a compelling sense of physical space, and allows images to emerge from the screen and enter further into the spectator's space, more than what is possible with conventional 2D or "flat" cinema.

In the most recent 3D "wave", new multi-media services such as Three-Dimensional TeleVision (3DTV) or Free viewpoint TeleVision (FTV) are being explored. 3DTV, on the one hand, is expected to offer depth perception of broadcasted TV programs without the need to wear special glasses. As a general consensus, 3DTV should provide a level of image quality and viewing comfort at least comparable to standard 2DTV. Furthermore, 3DTV should offer monoscopic compatiblity for 2DTV sets, for a gradual system transition. FTV, on the other hand, should allow the viewer

to switch through the available viewpoints of a 3D scene using a special controller. The transition between the different viewpoints is required to be as fluid as possible for a successful deployment of FTV.

Conventional stereo, one of the most popular 3D video format available on the market today, is not sufficient however to support deployment of 3DTV and FTV. To alleviate the burden of wearing special glasses for 3DTV, an auto-stereoscopic display must be used and it requires inputting multiple views to reduce flipping effects when switching from one stereo pair to another, and to provide at least some horizontal head motion parallax. The fluidity in navigation in FTV is also only possible if numerous views are available at the receiver side. In this context, the multiview video-plus-depth (MVD) 3D video format is particularly interesting, as it can provide multiple views at the receiver at the cost of transmitting only a handful of texture views and corresponding depth videos. Indeed, the other views can be interpolated using Depth Image Based Rendering (DIBR) algorithms.

Furthermore, MPEG has been planning to standardize a 3D video coding standard as a second phase of the FTV project since 2009, following the first phase which consisted in the standardization of the MVC extension of H.264/AVC. For that purpose, a Call for Proposals on 3D video coding technologies was issued in March 2011, and answered in November 2011. MVD turned out to be the most popular choice of 3D formats amongst the different contributions, hence proving its potentials. Between the various standardization tracks created after the answer to the CfP, one particular track aimed at standardizing an HEVC-compatible 3D video coding standard, which is basically an MVC extension of HEVC that includes depth. This extension, called MV-HEVC, allows to exploit inter-view redundancies in both texture and depth using high-level syntax only. The coding efficiency of dependent views and depth data is further increased using additional block-level coding tools in another extension called 3D-HEVC. The work on 3D-HEVC started in the beginning of 2012 at the MPEG side, then ITU joined in July 2012 and a joint ISO / ITU collaborative team for 3D video (JCT-3V) was created. Since it implies a redesign of the decoders, 3D-HEVC is not expected to be finalized before early 2015.

The work done in this thesis falls in the 3D-HEVC standardization context. The goal is to develop new coding tools on top of the 3D-HEVC reference software to further improve the coding efficiency of coded and synthesized views. Indeed, Orange follows 3D-HEVC standardization as a continuity of its previous activities in HEVC. The work in this thesis was thus performed in the Advanced Video Coding (CVA) team of Orange Labs, and in the Multimedia (MMA) group of the Signal and Image processing department (TSI) of Telecom ParisTech and the LTCI laboratory (UMR 5141).

Contributions

Various methods aimed at increasing the coding efficiency in 3D-HEVC were developped during this thesis. They can be divided into two categories. The first category includes methods that comply to real-world constraints in terms of complexity, memory usage, and practicality. These are thus more oriented towards standardization. We identify three contributions in this category:

- A method that exploits the statistical Intra direction dependency between texture and depth. The texture Intra directions are conditionally inherited by depth blocks and used to predict the depth Intra directions.
- A method for improving Interview Motion Vector Prediction (IVMP) in 3D-HEVC which adds a Disparity Vector (DV) candidate in the Merge list for a better Motion Vector (MV) / DV equilibrium in the list. IVMP is also improved by modifiying the Neighboring DV (NBDV) process for DV derivation.
- A method that exploits the link between the texture and depth quadtrees to save on both encoder runtime and coding of partition information. The depth quadtree is limited to the texture quadtree or inversely, the texture quadtree is initialized from the depth quadtree.

The methods in the second category are unconstrained coding approaches where the aim is to achieve significant coding gains regardless of the introduced complexity. A novel optical flow approach is used to either increase the coding efficiency of dependent texture views in 3D-HEVC, or to improve the view synthesis after decoding. There are thus two contributions in this category:

- A method that combines a smart decoder approach and an optical flow computation to benefit from a dense motion vector field that significantly improves the block predictors at no additional cost. The optical flow is applied on the reconstructed base view after decoding (this can be done at both the encoder and decoder sides) and then the resulting dense motion vector field is inherited by blocks in dependent views as a Merge candidate.
- A view synthesis method that blends four different frame predictions obtained through motion compensation with four dense motion vector fields. These are computed using optical flow at the level of two reference views (using a past and a future reference), then warped to the synthesized view.

Structure of the manuscript

This manuscript starts with a state-of-the-art in 3D video representation and coding, and a detailed overview of the coding tools in 3D-HEVC. Then, it comprises two parts which describe, respectively, coding approaches aimed towards standardization, and more innovative approaches using optical flow. More precisely, the manuscript is organized as follows:

- Chapter 1 presents a state-of-the-art in 3D video representation and coding. It starts by presenting depth perception in 3D displays. 3D video formats are described next, followed by tools developped to specifically code 3D video. A list of 3D video standards and a summary of recent standardization activites in this field are detailed next. A comparison of 3D-HEVC with other 3D video coding standards concludes this chapter.

- Chapter 2 lists the coding tools present in 3D-HEVC at the time of developping our methods. The tools used to code the HEVC-compatible base view are detailed first, followed by the ones used to code the dependent views and the depth videos. A presentation of non-normative encoder controls concludes this chapter.

Part one

- Chapter 3 details our depth video coding method based on the Intra mode inheritance from texture. First, a preliminary study of the texture/depth Intra mode matchings is given. Then, the method is presented with a first criterion used to condition the inheritance: GradientMax. The results of our method with GradientMax are reported next. In an aim to further increase coding efficiency, another criterion, DominantAngle, is presented next. Its associated results are reported to allow for a direct comparison with GradientMax, which concludes the chapter.

- Chapter 4 presents two methods to improve IVMP in 3D-HEVC. The first adds an inter-view candidate in the Merge candidate list of dependent view PUs to achieve a better equilibrium between MVs and DVs in the list. The second tackles the sub-optimality issue of the NBDV process by considering multiple neighboring DVs and selecting the median as final DV. For both methods, a state-of-the-art, a motivation, a detailed description, and a reporting of the results are given.

- Chapter 5 presents the depth quadtree limitation using texture (QTL), the texture quadtree initialization using depth (QTI), and the predictive coding

(PC) algorithm which can be applied in both methods. A state-of-the-art of existing encoder shortcuts is first given. The two methods are then motivated by analyzing the percentage of CUs where the texture quadtree is more partitionned than the depth quadtree, and the percentage of bits used to code the partition information in the bitstream. QTI and QTL are then respectively presented, along with their variants. The objective and subjective coding results of the methods, and the runtime savings they bring are reported, then analyzed thoroughly.

Part two

- Chapter 6 presents a coding method that uses optical flow and a smart decoder approach to increase the coding efficiency of dependent texture views in 3D-HEVC. A state-of-the-art on these two aspects is presented to put the method in context. Then, the method is motivated by showing that the dense motion vector field (DMVF) provided by optical flow gives more accurate predictions than the conventional block-based coarse motion vector field in HTM, even if the DMVF is computed between reconstructed frames and is thus affected by quantization noise. The method is then presented: basically an optical flow computation is performed, at both encoder and decoder sides, between the reconstructed base view frame and one of its temporal reference frames. The resulting DMVF is then inherited at the level of dependent view PUs as a dense Merge candidate. The significant coding gains of the method are then reported and interpreted.

- Chapter 7 presents a novel view synthesis method that uses an optical flow approach as well to increase the quality of synthesized views. A state-of-the-art on view synthesis techniques is first given, followed by a presentation of the method. Basically, four DMVFs are computed at the level of two coded reference views using two temporal reference frames (one in the past, the other in the future). These DMVFs are then warped to the level of the synthesized view using an epipolar constraint, then used for motion compensation. The four resulting predictions are blended as a final step of the algorithm. The significant coding gains on synthesized views are then given and analyzed.

We end this manuscript with a summary of the proposed methods and their associated results, as well as some perspectives for future work in this field.

Chapter 1

3D video representation and coding

Contents

We begin this thesis manuscript by introducing the mecanisms by which the viewer can experience 3D while watching a video on a 3D display. Different types of display technologies will be discussed, underlining the advantages and disadvantages of each.

Furthermore, the work done in this thesis is aimed at improving the coding efficiency of 3D video data for stereoscopic and autostereoscopic displays, represented in a specific Multiview Video-plus-Depth (MVD) format. In a second section of this chapter, we thus position the MVD format along side other 3D video representations. We first discuss texture-only formats, and then introduce depth-based formats which include MVD.

The 3D information, regardless of the format it is represented in, is costly in terms of bitrate; it thus needs to be efficiently compressed. A myriad of tools are proposed

for this purpose in literature and in 3D video codecs. In a third section, these tools will first be categorized and detailed. Then, a list of existing video standards suitable for compressing 3D data will be established. New standardization activities in this field, leading to the draft 3D extension of HEVC (3D-HEVC) on which our work is based, will also be presented. A performance comparison between different 3D video coding schemes is also performed in this section, in order to prove the superiority of 3D-HEVC in efficiently coding MVD data.

1.1 Depth perception in 3D displays

1.1.1 Depth cues

The human visual system uses several depth cues to build a mental model of a perceived 3D scene [RHFL10, LRL13]. Depth cues can be divided into two main categories: oculomotor and visual.

Oculomotor depth cues include accomodation, convergence and myosis. Accomodation is a change of the focal length of the eye's lens in order to bring objects at different distances into sharp focus on the retina. Convergence is a rotation of the eyes toward each other for closer objects. Myosis is the constriction of the pupil size relative to the amount of light the pupil receives. Occulomotor depth cues are relatively weak, they are only effective for short distances (less than 10m).

Visual depth cues include monocular cues and binocular cues. Monocular cues can be either static or dynamic. Static monocular cues are classic pictorial cues such as shadow, illumination, relative size differences, aerial perspective, linear perspective, etc., while dynamic monocular cues correspond to motion parallax. Indeed, motion cues are created when the viewer moves his eyes or head. A relative object motion around a fixation point can serve as a depth cue. Monocular cues are important depth cues which span a large range of scene depths. Binocular cues allow a viewer to experience the sensation of depth (stereopsis) through the existence of different retinal images captured by each eye. Those are only effective for medium viewing distances (from a few centimeters to nearly 100 meters). Figure 1.1 compares the depth perception layers of different types of depth cues [Ebr12].

1.1.2 3D displays

A natural real world scene provides multiple depth cues to the viewer, allowing him to mentally build the model of the scene and experience 3D. In order to be able to view 3D video, a 3D display should do the same. However, not all depth cues can be provided by 3D displays.

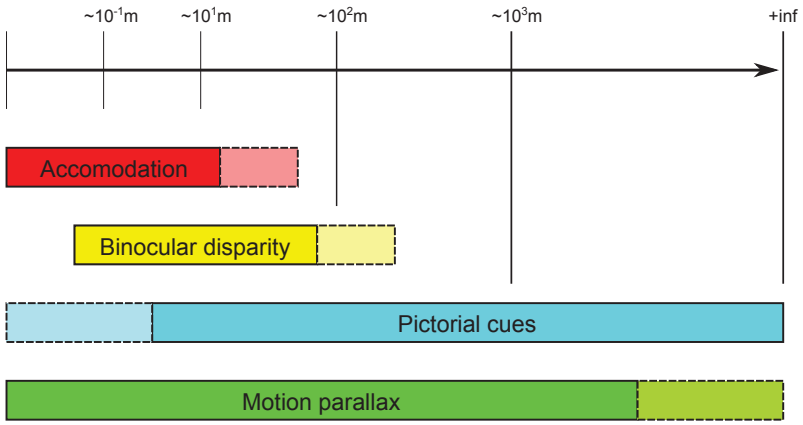


Figure 1.1: Depth perception layers of different types of depth cues

(a) Anaglyph (b) Polarized (c) Shutter

Figure 1.2: Types of glasses used for stereoscopic viewing

The most common and commercially popular 3D displays provide only binocular disparity. These so-called *stereoscopic* displays multiplex two slightly different images (called views) of the same scene captured at the same time instant. In order to provide the binocular disparity depth cue, each image must be matched to a different eye of the viewer. This can be done either using special glasses (anaglyph, polarization or shutter glasses) as illustrated in Figure 1.2 [Ali] or without the need of special equipment, as in the case of *autostereoscopic* displays where the dissociation between views is done using a parallax barrier or lenticular arrays [Dod05], as shown in Figure 1.3. Autostereoscopic displays can multiplex more than two views and can thus provide motion parallax, at some extent. Indeed, as the viewer moves around the display, different stereo pairs are projected to his eyes, and hence, he can explore different elements of the 3D scene. These displays are reported to cause eye strain because of a discrepancy between the accomodation and the convergence distance, as shown in Figure 1.4. Indeed, the convergence and binocular parallax will place an object in front of the display, but the eyes must still focus on the display in order to make the object sharp in the retina. Another disadvantage is the lack of fluid motion parallax due to a limited number of views. In case there is no sufficient views

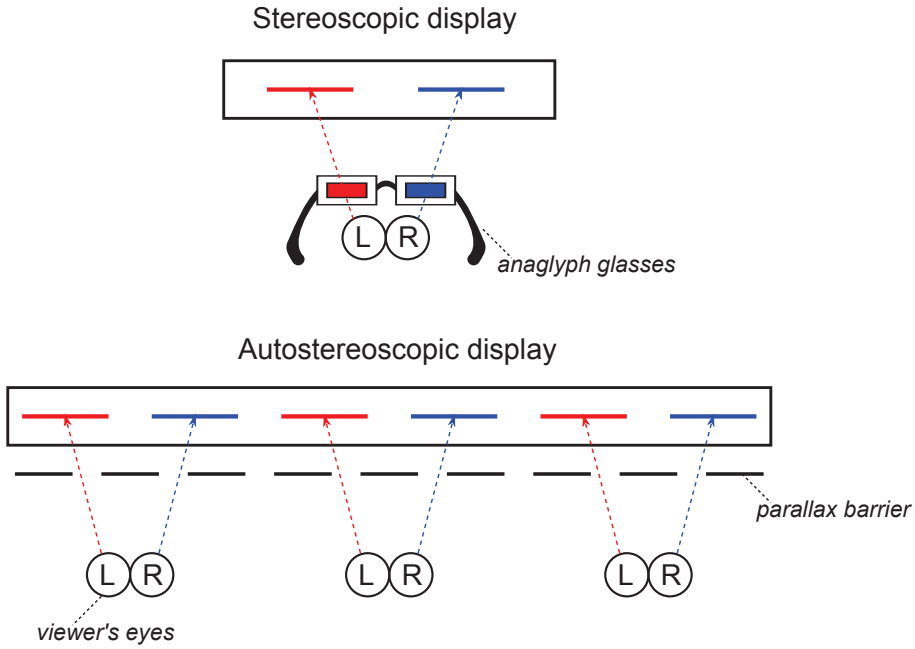


Figure 1.3: Stereoscopic and autostereoscopic displays

projected on the 3D display, the viewer will experience discomfort since he will not be able to perceive a slightly different perspective of the projected 3D scene with the natural movement of the head [Ols08]. In addition, when switching from one stereo pair to the next, an unpleasant flipping effect usually occurs in autostereoscopic displays. These two inconveniences have hindered the successful commercialization of stereoscopic displays.

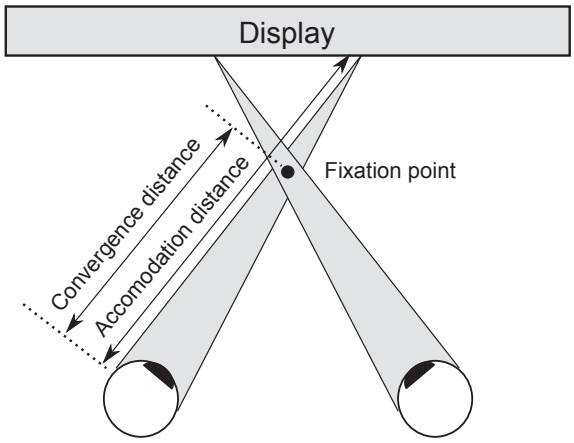


Figure 1.4: Discrepancy between the convergence and accomodation distances

To counter these issues, more advanced 3D display technologies are currently being explored. These include holoscopic and holographic displays. Holoscopic or Integral Imaging 3D displays consist of a display panel and a lens array placed in front of the panel. The display panel projects several elemental images, each corres-

ponding to a different viewpoint of the same scene. The light rays from the elemental images are integrated by the lens array in order to form a 3D image of the captured scene [PHL09]. Figure 1.5 [3Dv] shows a 3D image formed using a holoscopic display. The 3D images formed present continuous parallax throughout the viewing zone. This continuity eliminates the flipping effect discussed earlier [3Dv]. Head motion parallax is fully provided by these displays. In addition, the accomodation and convergence distances are the same, and thus eye strain is greatly reduced [3Dv]. However, at the time of writing this thesis manuscript, the display, with its complex optical system, is still relatively expensive to be commercially popular. Plus, there is little content available today for holoscopic viewing. Indeed, the capture technology is not mature, and specific compression schemes for integral images are yet to be established.

Figure 1.5: Holoscopic display

Another solution that counters most of the issues of classical stereoscopy is the holographic display technology, where a light distribution is physically duplicated into a volume of interest [YKO10]. Specifically, the wave field of a 3D scene is recorded onto a hologram, and the display reconstructs it in space by modulating coherent light, e.g. with a Spatial Light Modulator (SLM). Consequently, holographic displays can provide nearly all depth cues and are thus labeled as "true 3D" displays. Figure 1.6 [SCS05] shows an example of a 3D image produced by computer generated holograms (CGH). The commercialization of holographic displays is not expected any time soon however since the computational and bandwidth cost associated with holographic data is, to this day, too high.

Each display technology has its own associated video formats. The following section will detail the 3D video formats for stereoscopic and autostereoscopic displays. Holoscopic and holographic display technologies are not in the scope of this thesis, and thus, they will be disregarded in the remainder of this manuscript.

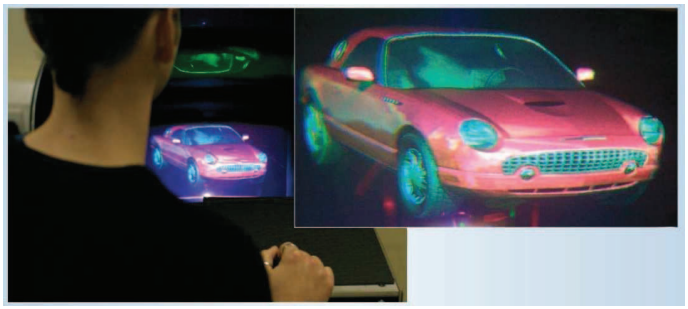


Figure 1.6: Replay of a spatial-multiplexed, 3×8 billion-pixel, full-parallax, fullcolor, 3D image with a holographic display

1.2 3D video formats

1.2.1 Texture-only 3D video

The most common 3D video format is the Conventional Stereo Video (CSV) format, in which two views representing the same scene but captured by two different cameras separated by a certain distance (called the baseline), are multiplexed in the 3D display, as illustrated in Figure 1.7.

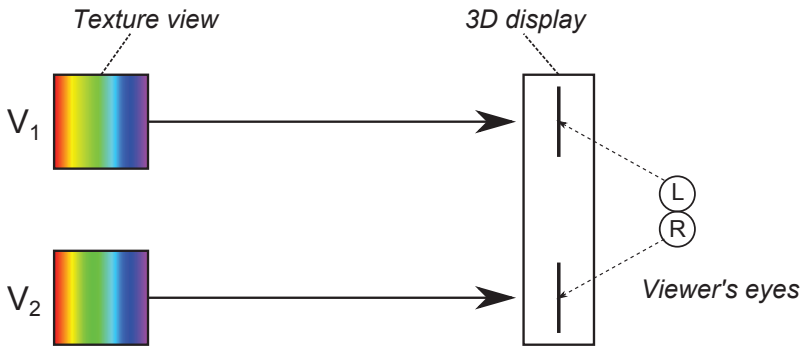


Figure 1.7: Convention Stereo Video format

The Mixed Resolution Stereo (MRS) format is an alternative to CSV [BSM+09]. It exploits the binocular suppression theory which states that if two views of different image quality are multiplexed together on a stereo display, the resulting 3D image quality will be closer to that of the higher quality view. This means that one of the two views can be represented at a lower resolution, as illustrated in Figure 1.8, without sustaining a significant loss in the resulting 3D image quality.

Multi-resolution Frame Compatible (MFC) formats multiplex both views of a stereo format onto a single support. The multiplexing can be done either spatially (in a *side-by-side* or an *over-under* configuration, also called *top-bottom*) or temporally. If temporal multiplexing is used, the views are interleaved as alternating frames or fields, whereas both views are downsampled either horizontally or vertically then

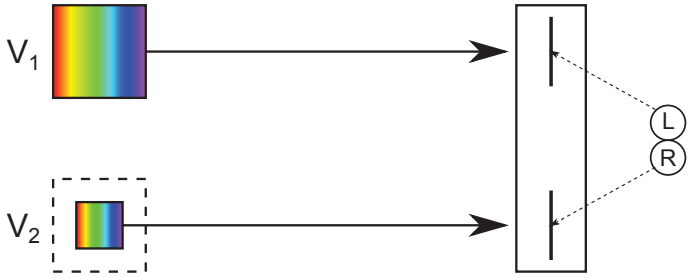


Figure 1.8: Mixed Resolution Stereo format

joined together in spatial multiplexing, hence losing spatial resolution. An example in Figure 1.9 illustrates these different types of frame packing arrangements.

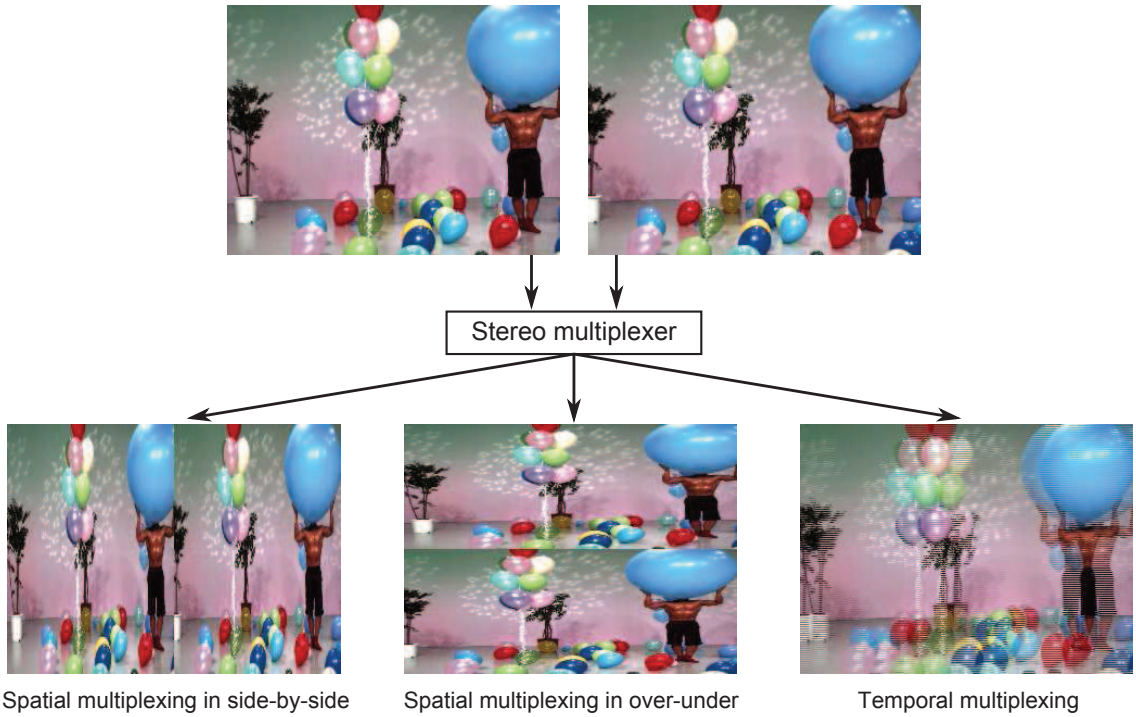


Figure 1.9: Stereo multiplexing

New emerging multimedia services require more than two views to be multiplexed on the 3D display. 3D Television (3DTV) [FCSK02] for instance, requires motion parallax to be supported, at least within practical limits: a viewer must be able to view different elements of the 3D scene when moving from left to right (looking behind objects for instance). This is impossible to perform using only two views. Furthermore, two views are not sufficient to provide fluidity in navigation, as required by applications such as Free viewpoint TeleVision (FTV) [CTMS03] for instance, in which the user is free to navigate in the 3D scene using a controller. To that end, the MultiView Video (MVV) format exists. The 3D data in a MVV format is composed of N views captured by N cameras arranged and spaced in a

specific manner.

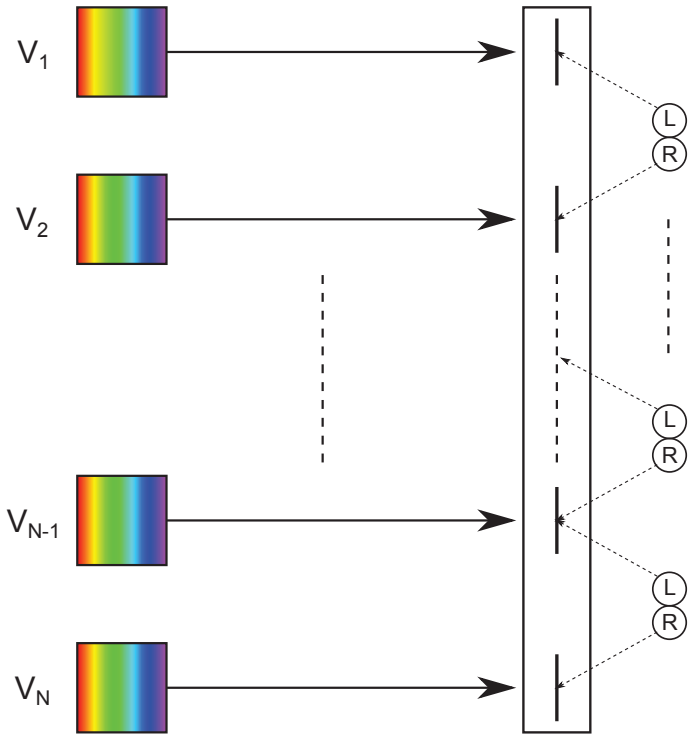


Figure 1.10: MultiView Video format

The bitrate required for transmitting the 3D data in MVV can increase rapidly as N increases. Depth based formats may be more cost-effective alternatives, hence the ever-increasing popularity of these formats in both academic and industrial research fields.

1.2.2 Depth-based 3D video

Definition and characteristics of a depth map

A depth map is a grayscale luminance only image which maps each pixel in an associated texture frame to a certain distance from the camera (a depth value). A depth map is essentially composed of large planar regions seperated by sharp edges and is invariant to illumination, patterned textures, and shadows [DTPP08, KOL+10, MdWF06, MD08, SSO09, MVJ+13]. Figure 1.11 shows an example of a depth map with its associated texture frame. The higher the luminance value (closer to white), the closer the object is to the camera.

View synthesis

Depth maps are not displayed on screen but rather used to synthesize views using a technique called Depth Image Based Rendering (DIBR). Namely, using a

Figure 1.11: A texture frame and its associated depth map from the Breakdancers video sequence

texture frame and its associated depth frame at a specific view, another view, located at a different point in space, can be extrapolated. View synthesis invokes a 3D image warping process that can be decomposed into two steps including a first back-projection of the reference image into the 3D-world, followed by projecting the back-projected 3D scene onto the targeted image plane [Dar09, McM97], as shown in Figure 1.12: a point m_1 representing a point M in real world coordinates in a reference image plane (or view) captured by a camera of center C_1 is warped to point m_2 in a targetted image plane that would be captured by another camera of center C_2.

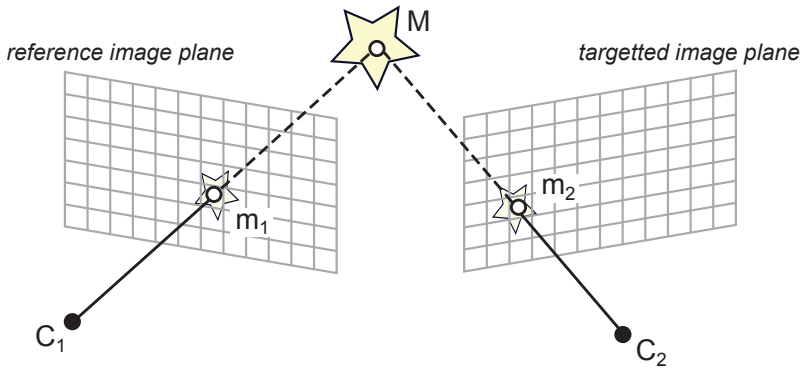


Figure 1.12: Image warping from a reference to a targetted image plane

Let us define some notations first before detailing the warping operations: let (u_1, v_1) be the coordinates of m_1 in the reference image plane, and (u_2, v_2) the coordinates of m_2 in the targetted image plane. Let $Z(u_1, v_1)$ be the depth value of m_1. The goal is to express (u_2, v_2) in terms of (u_1, v_1) and $Z(u_1, v_1)$.

First, m_1 is back-projected into point M of coordinates (x, y, z) in the 3D-world, using the following equation [Dar09]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R_1^{-1} K_1^{-1} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \lambda_1 - R_1^{-1} t_1 \tag{1.1}$$

where K_1, R_1 and t_1 are respectively the 3×3 intrinsic camera parameters matrix, the 3×3 orthogonal rotation matrix and the 3×1 translation vector of reference view 1. λ_1 is a scaling factor.

Then, point M is projected back onto the targetted image plane at point m_2 of homogeneous coordinates (u'_2, v'_2, w'_2) using the following equation:

$$\begin{pmatrix} u'_2 \\ v'_2 \\ w'_2 \end{pmatrix} = K_2 R_2 \begin{pmatrix} x \\ y \\ z \end{pmatrix} + K_2 t_2 \tag{1.2}$$

where K_2, R_2 and t_2 are respectively the 3×3 intrinsic camera parameters matrix, the 3×3 orthogonal rotation matrix and the 3×1 translation vector of reference view 2. More details on homogeneous coordinates can be found in [Dar09]. Replacing $(x, y, z)^T$ by the expression found in Equation 1.1, we obtain:

$$\begin{pmatrix} u'_2 \\ v'_2 \\ w'_2 \end{pmatrix} = K_2 R_2 R_1^{-1} K_1^{-1} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \lambda_1 - K_2 R_2 R_1^{-1} t_1 + K_2 t_2 \tag{1.3}$$

If we attach the world coordinates system to the first camera system, as it is usually done, then $R_1 = I_3$ and $t_1 = O_3$, where I_3 is the 3×3 identity matrix, and O_3 the 3×1 null vector. Equation 1.3 can thus be simplified as such:

$$\begin{pmatrix} u'_2 \\ v'_2 \\ w'_2 \end{pmatrix} = K_2 R_2 K_1^{-1} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \lambda_1 + K_2 t_2 \tag{1.4}$$

In this case, λ_1 can be written as:

$$\lambda_1 = \frac{z}{c} \text{ where } \begin{pmatrix} a \\ b \\ c \end{pmatrix} = K_1^{-1} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \tag{1.5}$$

Furthermore, if the depth map at view 1 is represented using 8 bits per pixel, z can be expressed as:

$$z = \frac{1}{\frac{Z(u_1,v_1)}{255} \cdot \left(\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}} \right) + \frac{1}{Z_{\text{far}}}} \tag{1.6}$$

where Z_{near} and Z_{far} are the extremal depth values. Finally, $(u_2, v_2) = (\frac{u'_2}{w'_2}, \frac{v'_2}{w'_2})$.

If we suppose that the cameras are identitical and that they are rectified (parallel), then $K = K_1 = K_2$, $R_2 = I_3$ (no rotation angle between cameras), and the

translation between cameras is only on one axis, meaning $t_2 = (t_x, 0, 0)^T$. In this case, $\lambda_1 = z$ and Equation 1.4 can be simplied into:

$$\begin{pmatrix} u'_2 \\ v'_2 \\ w'_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} z + K \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix} \tag{1.7}$$

In non-homogeneous coordinates, we obtain:

$$u_2 = u_1 + \frac{f \cdot t_x}{z} \text{ and } v_2 = v_1 \tag{1.8}$$

where f is the focal length of the cameras.

3D image warping often introduces occlusions and disocclusions in the synthesized view. Occlusions is when two points of different depth and clearly visible in the reference view, are mapped to the same point in the targetted view. The point with the higher depth value (the foreground) will overlap the other (background). Disocclusions are points in the targetted view that are not mapped to any point in the reference view. These correspond to areas that were not visible in the reference view (either covered by another object or simply non-existant) and which became visible in the targetted view. Filling these holes can be done in various ways. First, the synthesis can be performed at the encoder side, and the missing information sent in the bitstream (although this increases the required bitrate). Second, the depth video can be pre-processed (general smoothing [TAZ+04], assymetric filtering [ZT05], bilateral filtering [CLLY08], etc.) to reduce the number of holes. Finally, the synthesized video itself can be post-processed to fill the holes (average filtering [ZT05], multiple reference images [ZWPSxZy07], inpainting [TLD07], etc.).

Depth-based formats

First we have the Video plus Depth (V+D) format, which consists of one texture view and its associated depth view. With those two components, one or more views can be extrapolated. Figure 1.13 shows one extrapolated view using a V+D format, which along side the original texture view, allows stereoscopic viewing. Autostereoscopic viewing can be achieved if more than one view is extrapolated. However, the more distant from the original texture view the extrapolated views are, the more they contain disoccluded areas, and hence the less qualitative they will be.

The Multiview Video plus Depth (MVD) format solves this particular issue by introducing view interpolation. It consists of N texture views and their associated N depth views. With two pairs of texture and depth views, intermediate views can be interpolated, meaning they are extrapolated from each pair and the two repres-

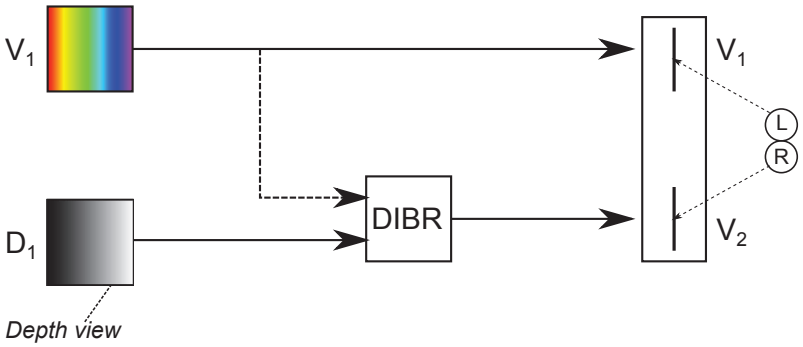


Figure 1.13: The Video+Depth format

entations are blended together to reduce the number of holes due to disocclusions. This allows synthesizing multiple intermediate views between two original texture views, hence enabling autostereoscopic viewing, without the constraint of disocclusions. The only constraint is relative to the baseline since a small baseline allows synthesizing a limited number of views. Figure 1.14 shows three intermediate views synthesized using two texture / depth pairs in an MVD format.

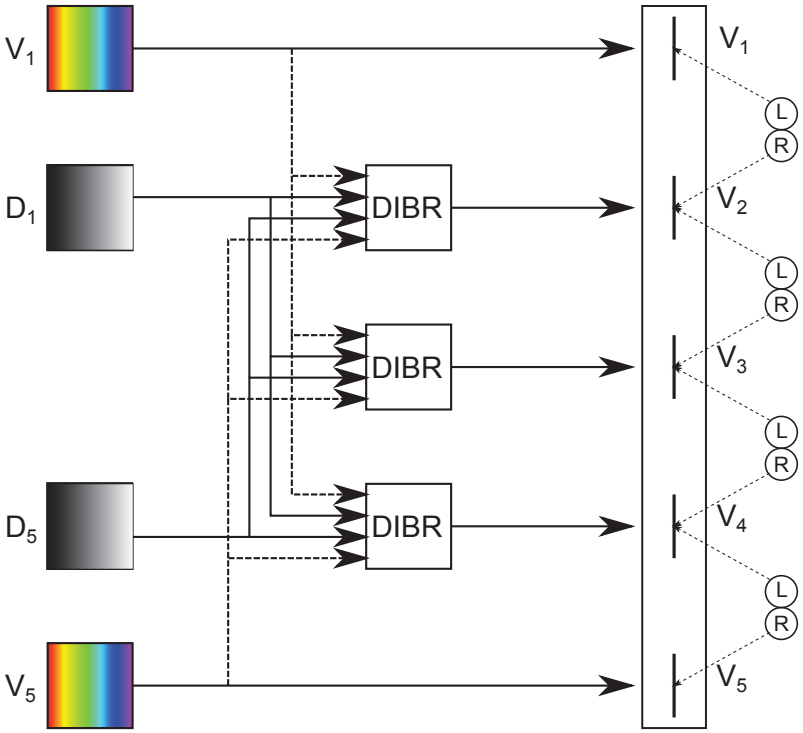


Figure 1.14: The Multiview Video plus Depth format

Layered Depth Video (LDV) is an alternative to MVD. Using a texture / depth pair (V_i, D_i) at view i, the texture / depth map pair (V^*_{i+1}, D^*_{i+1}) at view $i+1$ can be extrapolated using DIBR. A residual (RV_{i+1}, RD_{i+1}) can be computed as $(V_{i+1} - V^*_{i+1}, D_{i+1} - D^*_{i+1})$ which corresponds to the areas of view $i+1$ which could

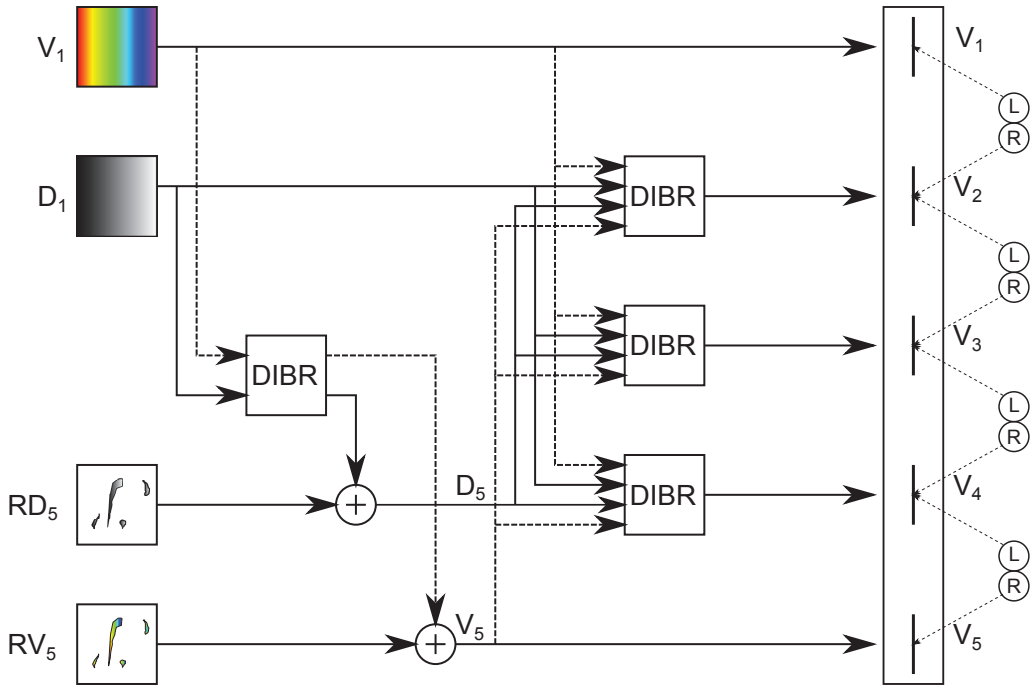


Figure 1.15: The Layered Depth Video format

not be rendered from view i (disocclusions). The residuals corresponding to the different views and at least one complete texture / depth pair (designated as *main layer*) for reference are then sent in the bitstream. Figure 1.15 shows a main layer composed of V1 and D1 and a residual layer composed of RV5 and RD5 and how the V5 / D5 pair is reconstructed using the two layers.

1.3 3D video coding

1.3.1 Overview of coding tools for MVD representations

In an MVD format, N texture views and N associated depth views are to be coded and transmitted. This represents quite a large amount of information, which increases linearly with N. A simulcast coding of each texture and depth view using a standard 2D codec such as AVC [H2605] or HEVC [BHO+13] is possible, but that would not allow to exploit the correlations between the different views. Furthermore, depth videos have different characteristics than texture videos and conventional compression algorithms might not be very efficient for this type of data. New coding tools must thus be designed specifically for depth maps. In this section, we present an overview of coding tools proposed in literature which are designed to code 3D data in an MVD representation.

Texture video coding

Standard 2D video codecs include an Inter coding mode where a block can be coded using Motion-Compensated Prediction (MCP). Coding a block with MCP involves two stages: motion estimation and motion compensation. Motion estimation is the non-normative process by which the encoder tries, for the currently coded block, to find the best matching block in an already decoded picture stored in a buffer called Decoded Picture Buffer (DPB). The best matching block is the one minimizing a certain criteria, such as the Mean of Absolute Distortion (MAD). Once found, a *motion vector* (MV) representing the displacement between the current block and the best match, and the index of the picture in which the latter has been found (called *reference index*) are sent to the decoder. Motion compensation is the normative reverse process in which the decoder builds the predictor of the block using the motion parameters (MV + reference index) associated with it (it basically recovers the best matching block previously found by the encoder). For 3D data, inter-view correlations can be exploited by extending MCP into the view direction. This basically consists in adding an already decoded picture at the same time instant but in a different view in the DPB of the current frame. The MV found in this case does not represent motion *per se*, it represents a certain disparity between views and is thus called *disparity vector* (DV). A block coded in Inter mode and which is associated with a DV is said to be coded using Disparity-Compensated Prediction (DCP) [CWU+09]. Figure 1.16 shows a current block in view V_j at time instant T_i coded in Inter mode, either with MCP (it has a MV pointing to a decoded picture at the same view V_j but at a different time instant T_{i-1}) or with DCP (it has a DV pointing to a decoded picture at the same time instant T_i but in a different view V_{j-1}).

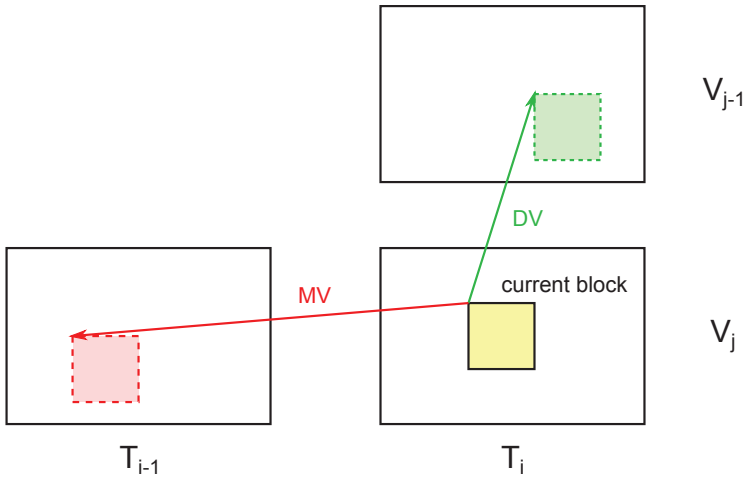


Figure 1.16: Motion and disparity-compensated predictions

Aside from DCP which exploits inter-view correlations, several coding tools exploit inter-component correlations (between texture and depth) to increase coding efficiency of texture data. In [KD10], an inter-view Direct mode is added in the MVC codec, wherein the motion parameters of each pixel in the currently coded block are inferred from a reference view at the same time instant. Basically, for each pixel (x_i, y_i) in the current block, the corresponding pixel in the reference view (x_r, y_r) is found by warping using the coded depth information $Z(x_i, y_i)$ (see Equation 1.4). The motion information contained in (x_r, y_r) is then copied to (x_i, y_i). If it does not exist (the block containing (x_r, y_r) is Intra coded or (x_r, y_r) is occluded), another reference view is tested. If the motion information of (x_i, y_i) cannot be inferred from reference views, it is inferred from local neighboring points in the current view (some sort of inpainting is applied for instance). Once the motion information of all the pixels of the current block is found, a pixel wise motion compensation is applied to form the prediction. This tool reportedly brings 6.5% bitrate reduction on top of JMVC and 10.9% on top of JMVM+Motion skip (JMVC and JMVM are respectively an older and a newer version of the MVC reference software). However, the computation of the reported gains was only performed for coded texture views (no results for synthesis) while not considering the depth rate, although depth is essential to decode the texture in this case.

View synthesis prediction (VSP) is also a popular coding tool where using a decoded texture / depth pair in a reference view $j-1$, a texture frame at view j can be synthesized and used to improve the coding efficiency of the currently coded texture frame at view j. This can be done in various ways: the synthesized frame can be added into the reference picture lists of the current frame where it will be used by MCP as in [Jag12a]. In [DGK+11], the synthesized frame is compared to the current frame in order to know which areas cannot be rendered (due to disocclusions), and only the non-renderable parts of the current frame are coded. However, VSP does not account for inter-view signal mismatches caused by camera heterogeneity and the non-Lambert reflection of objects (difference of sharpness, blurriness, color, illumination, etc.) and thus, it may sometimes be inefficient.

In [SK10], a temporal improvement of VSP is proposed. Using $T^*[i-1][j-1]$ and $T^*[i-1][j+1]$ which are the reconstructed texture frames at time instant $i-1$ at views $j-1$, and $j+1$, and their associated reconstructed depth maps $D^*[i-1][j-1]$ and $D^*[i-1][j+1]$, a texture frame $T'[i-1][j]$ can be synthesized. The same can be done at the current time instant i: using $T^*[i][j-1]$, $T^*[i][j+1]$, $D^*[i][j-1]$ and $D^*[i][j+1]$, a texture frame $T'[i][j]$ can be synthesized. A motion estimation can be performed between the two synthesized frames in order to derive a motion vector field (MVF). Since all this information is available at the decoder, the motion

estimation can also be done there, so there is no need to transmit the derived MVF. Once the MVF has been found, a motion compensation process can be performed using the MVF and $T^*[i-1][j]$ in order to form a prediction $\widehat{T}[i][j]$ that is used to code the current frame $T[i][j]$. This tool reportedly brings around 2% bitrate reduction over the standard VSP method, but it is also a lot more complex since motion estimation is done at the decoder.

Depth information can also be used to improve motion and disparity vector prediction. In [HRS+13], a default DV predictor (DV_{pred}) is computed from the depth map if none of the neighboring blocks of the current texture block has any DVs. The average of the collocated decoded depth block is computed and that depth value is transformed into a DV which is set as DV_{pred}. A second part of the tool modifies the Skip and Direct modes in H.264/AVC: the MVs of the three spatial neighbours are each used to compensate the collocated decoded depth block, and a Sum of Absolute Distortions (SAD) is computed between that depth block and its predictor. The vector yielding the lowest SAD was inherited by the texture block. This brings 8.9% bitrate reduction on coded texture views in a three view coding (left, center, right) scenario. However, this tool was judged too complex because in Skip or Direct mode, the decoder would have to perform three SAD computations for each block. The first part of the tool was adopted in the 3D extension of AVC: 3D-AVC [RCZS13]. The second was replaced in 3D-AVC with a simpler Disparity-based Skip and Direct modes where the motion information of the corresponding base view block pointed by DV_{pred} is inherited.

Rate control for texture can also take advantage of depth data. Indeed, the depth map allows to identify the areas in the texture frame that belongs to the background, and the ones corresponding to the foreground. In order to improve the perceptual quality of the coded texture frame, a higher quantization parameter (QP) (less bitrate) can be set in the background and a lower one in the foreground [DGK+11].

Depth video coding

Depth videos, by their natural composition and purpose in the video chain, have different characteristics than texture videos and thus, specific tools have been developped in order to code them. This area of research has been very active in the recent years due to the increased industrial interest in depth-based 3D video formats. Furthermore, depth coding methods can be divided into three categories: methods that exploit the inherent characteristics of the depth maps, methods that exploit the correlations with the associated texture videos, and methods that optimize depth coding for the quality of the synthesis.

1. **Methods that exploit the inherent characteristics of depth videos:**

the methods in the first category can be applied at the slice or at the block level in depth map coding, hence respectively defining two sub-categories of methods. In the first sub-category, we find the reduction of the resolution of depth videos before coding, which may be considered as a depth data compression technique [RH11]. Indeed, the artifacts associated with up-sampling the depth map after decoding are minimal, and we can thus obtain considerable compression gains by simply reducing the resolution of the depth videos to encode. The test model of 3D-AVC [RCZS13] proposes to down-sample depth videos to a quarter of their initial resolution (half in each direction) before encoding.

Another tool in the first subcategory is the Z_{near} Z_{far} compensation [DGK+11] for weighted prediction. Different frames of a same view or different views at the same instant in time, for a depth video sequence, may have different extremal depth values (noted Z_{near} and Z_{far}). As depth maps are rescaled in the $[0, 255]$ interval, different depth images may be scaled differently. In this case, the use of a depth map as the reference image for another depth map would lead to poor predictions and unefficient compression. To solve this problem, coherent scaling needs to be applied for all depth maps in question. This is easy to achieve using extremal values. For example, if a depth map with extremal values Z^s_{near} and Z^s_{far} is used as the reference image for a current depth map with extremal values Z^t_{near} and Z^t_{far}, the scaling for the current depth map is done as follows:

$$L_T = L_S \cdot \frac{Z^s_{far} - Z^s_{near}}{Z^t_{far} - Z^t_{near}} + 255 \cdot \frac{Z^s_{near} - Z^t_{near}}{Z^t_{far} - Z^t_{near}} \tag{1.9}$$

where L_S is the original depth value and L_T the rescaled value. This tool is included in 3D-AVC under the name: Depth Range based Weighted Prediction (DRWP), although implemented differently.

Non-linear Depth Representation (NDR), included in 3D-AVC for coding of the base view, allows representing closer objects more accurately than distant ones. This can be achieved by encoding non-linearly mapped disparity, normalized in the $[0, 255]$ range, instead of generic depth map values. The depth map is nonlinearly mapped through forward lookup table at the preprocessing stage of the encoder and inversely mapped back to original representation at the post-processing stage of the decoder. Only a few key values need to be mapped (and this mapping is sent in the bitstream), the others can be obtained through piece-wise linear interpolation.

Furthermore, the encoder can be configured to use only full-pel motion estima-

tion for depth coding. Indeed the eight-tap filters used for motion-compensated interpolations, as defined in the HEVC standard [HEV13], produce artifacts on sharp edges in depth. Consequently, motion (or disparity)-compensated predictions may be modified to avoid interpolations, meaning that only full-pel motion vectors and disparity vectors will be used. The reduction in precision of motion vectors also reduces the bit rate required to transmit the motion vector differences [SBB11].

In the second subcategory, *i.e.*that of block-level methods, we find the approximation of depth blocks using modeling functions, presented in [MMS+08]. As the depth map is essentially made up of smooth regions separated by contours, a depth block may be approximated by four different types of functions: a constant function, a linear function, a constant piecewise function or a linear piecewise function. If no suitable approximation can be found for the current block, then it is divided into four blocks in a quad-tree manner. The process is repeated for each block until an approximation function is found for each leaf of the quad-tree.

While the first tool concerned Intra prediction, the so-called adaptive 2D Block Matching (2D-BM) / 3D Block Matching (3D-BM) selection tool concerns Inter prediction [KFM10]. 2D-BM is the classic way in which an encoder, during a motion estimation stage, finds the temporal reference block which best corresponds to the current block. The correspondence is measured by the Sum of Squared Errors (SSE), defined as follows:

$$SSE(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(f(m,n) - g(m+i,n+j)\right)^2 \tag{1.10}$$

where f is the original current block of $M \times N$ to code, and g is the temporal reference block, also of size $M \times N$. In this case, the search is carried out in two dimensions: horizontal and vertical. However, in depth videos, we also have motion in the depth direction. The search precision may thus be increased by an extension to a third dimension. This idea is implemented in 3D-BM. Thus, the best temporal correspondence is the one minimizing the new formulation of the SSE, as follows:

$$SSE(i,j,k) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(f(m,n) - g(m+i,n+j) + k\right)^2 \tag{1.11}$$

3D-BM is more efficient than 2D-BM for high bitrates, where the gains produced by the reduction of distortion are higher than the costs associated with

the addition of a new component for motion vectors. Inversely, at low bitrates, 2D-BM is more efficient than 3D-BM. Adaptive 2D-BM / 3D-BM selection evaluates the R-D cost for the coding of each block with 2D-BM and 3D-BM and chooses the method which minimizes cost. The choice is then sent in the bitstream for decodability. This adaptive selection is more efficient than pure 2D-BM and 3D-BM for middle bitrates.

2. **Methods that exploit the correlations with the associated texture videos:** in this category, we find some mode decisions for depth that depend on the texture. the Skip mode, for instance, can be forced for a currently coded depth block each time that the co-located texture block is coded using Skip [KOL+09]. The main idea behind this tool is that the spatial uniformity of movement in texture is likely to be also present in depth. Skip mode may not necessarily be chosen under normal circumstances due to the presence of certain artifacts inherent in the original depth, which create false movements. Imposing Skip mode on a depth block if it is chosen for the co-located texture block eliminates these artifacts, thus increasing the quality of views which will be synthesized with the current depth frame. The tool also brings compression gains due to the fact that no signaling is required for Skip mode. Finally, it also reduces the complexity of the encoder as it reduces the number of motion estimations and compensations to be performed.

   Prediction information can also be inherited by depth from texture. Motion information (motion vectors + reference picture indices) is highly correlated between texture and depth, especially around object contours. In [SPW+10], a new mode, known as Motion Sharing or MS, is added to the list of existing modes (Intra, Inter, Skip). In this new mode, the motion information for a depth block is directly inherited from the corresponding texture block. No motion estimation is carried out. MS mode is not systematically imposed for depth blocks: an R-D criterion verifies the cost and compares it to those entailed by other prediction modes. MS is then selected only if it offers the lowest cost. This tool is also included in 3D-AVC under the name of Inside View Motion Prediction.

   The Intra mode of a texture block may also be inherited, as proposed in [BYN11], for the co-located depth block. Inheriting the texture Intra mode here means that the mode will be added to the list of most probable modes (MPM, will be described in the next chapter) for the current depth block, where it will act as a predictor for the depth Intra mode. While the proposal is efficient, the fact remains that one of the two already present MPM candidates is always removed in order to replace it with the texture Intra mode, otherwise there

would be an additional bit required per MPM index signaling. However, this substitution is not always the best choice if there is little dependency between the texture and depth Intra modes; *i.e.*, a potentially good MPM candidate might be replaced with a bad one.

3. **Methods that optimize depth map coding for the quality of the synthesis:** as depth maps are not shown on screen but rather used to synthesize views, the R-D model used in depth coding must consider distortion directly on the synthesized views in order to optimize coding for the truly important aspects, i.e. the intermediate synthesized views. In [LOL11], the VSD (View Synthesis Distortion) metric is calculated as a distortion in the depth block, weighted by one-pixel texture translation differences:

$$VSD = \sum_{(x,y)} \left[\frac{\alpha}{2} \cdot \left| D_{x,y} - \widetilde{D}_{x,y} \right| \cdot \left(\left| \widetilde{C}_{x,y} - \widetilde{C}_{x-1,y} \right| + \left| \widetilde{C}_{x,y} - \widetilde{C}_{x+1,y} \right| \right) \right]^2 \quad (1.12)$$

where $D_{x,y}$, $\widetilde{D}_{x,y}$, and $\widetilde{C}_{x,y}$ are respectively the original depth value, the reconstructed depth value and the reconstructed texture value at position (x, y). The value of α is defined as follows:

$$\alpha = \frac{f \cdot B}{255} \cdot \left(\frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right) \quad (1.13)$$

where f is the focal distance, B the distance between cameras for the current view and the synthesized view, and Z_{near} Z_{far} are the extremal depth values.

In Equation 1.12, we suppose that two adjacent pixels will remain adjacent after the warping operation, which is not always the case as occlusions or disocclusions may occur in synthesized views. To rectify this, $\widetilde{C}_{x-1,y}$ and $\widetilde{C}_{x+1,y}$ in Equation 1.12 may be replaced by $\widetilde{C}_{x_L,y}$ and $\widetilde{C}_{x_R,y}$ respectively, with x_L and x_R defined as follows:

$$\begin{aligned} x_L &= x + \arg\max_{l \geq 1} \lfloor \alpha \cdot (D_{x-l,y} - D_{x,y}) - l \rfloor \\ x_R &= x + \arg\min_{r \geq 1} \left[\max \left(\alpha \cdot (D_{x+r,y} - D_{x,y}) + r, 0 \right) \right] \end{aligned} \quad (1.14)$$

Equation 1.12 is then corrected to take account of regions of occlusion and disocclusion, as follows:

$$VSD = \begin{cases} \sum_{(x,y)} \left[\frac{\alpha}{2} \cdot \left| D_{x,y} - \widetilde{D}_{x,y} \right| \cdot P_C \right]^2 & \text{if } x_L < x \\ 0 & \text{if } x_L \geq x \end{cases} \quad (1.15)$$

where P_C, the weight in texture, is defined as follows:

$$P_C = \left|\widetilde{C}_{x,y} - \widetilde{C}_{x_L,y}\right| + \left|\widetilde{C}_{x,y} - \widetilde{C}_{x_R,y}\right| \tag{1.16}$$

Other depth coding tools aim to increase the sparsity of depth in the transform domain in order to increase coding efficiency. This is carried out by modifying the original values of the depth map before coding, as long as the change only affects synthesized views below a certain, predefined threshold. Depth map coding using Don't Care Regions (DCR) [VCG+12], falls into this category.

1.3.2 3D video coding standards

Different video coding standards can be used to code and transmit 3D video data in different representations. We first list exisiting standards, then introduce new test models currently in standardization ("test model" is the term used to reference a standard or an extension that has not yet reached final draft international status), and finally, we compare the performance of different 3D video coding schemes together.

Existing standards

For conventional stereo 3D formats, a multiview profile (MVP) for MPEG-2 has been defined, in which an MPEG-2 compatible base layer is coded, which corresponds to the first view. An enhancement layer corresponding to the second view is also coded using additional inter-view predictions with the first view as reference. The resulting bitstream is then decoded in order to reconstruct both views, but it remains compatible with MPEG-2 Main Profile (MP) decoders which can only decode the base layer [MPE98]. Stereoscopic video can also be coded with the multiview extension of H.264/AVC called Multiview Video Coding (MVC) [CWU+09], described in more details below.

For MFC formats, the H.264/AVC standard can be used to encode the multiplexed video if a Supplemental Enhancement Information (SEI) message is also sent in the bitstream to signal the existence and the type (side-by-side, over-under, temporal) of the frame packing arrangement [MWG05]. Once the decoder receives this message, it will know how to demultiplex the signal in order to reconstruct both views. The disadvantage of this technique is that the bitstream is not AVC-compatible. An H.264/AVC decoder will not be able to extract a 2D video out of the 3D data stream.

For video+depth formats, MPEG-C Part 3 [BF06] can be used to code the two components. MPEG-C Part 3 was standardized in 2007 as a response to a strong

industrial demand. An MPEG-C Part 3 encoder basically consists of two MPEG-2 or H.264/AVC encoders used to independently encode both components, and the two resulting bitstreams (plus other auxiliary data if available) are multiplexed into an MPEG-2 compatible bitstream. Due to the inherent characteristics of depth maps, the bitrate required to code the depth component is in the 10-20% range of the total bitrate required to code both components. The advantage of MPEG-C Part 3 is that the produced bitstream is compatible with legacy 2D decoders.

MPEG-4 Part 2 Multiple Auxiliary Components (MAC) is an alternative to MPEG-C Part 3 for coding video+depth data. In MPEG-4 Part 2 MAC, the depth video is inputted as an auxiliary channel. Traditionally, auxiliary channels describe the transparency of objects (*alpha* channel), objects shape, secondary textures... The encoding of the auxiliary components re-uses the same motion vectors as the texture component for motion compensation. Furthemore, this scheme can also be applied for conventional stereo formats [CYBH03], where a disparity vector field and residual luma and chroma coefficients after disparity compensation are all input as auxiliary components.

For MVV formats, MVC can be used. MVC uses DCP to exploit the correlations between views. In MVC, there is always a view which is coded independently of the other views, using only temporal predictions. This view, which corresponds to S0 in Figure 1.17 [CWU+09], is denoted as base view, and it provides backward compatibility for legacy 2D H.264/AVC decoders. The base view provides random access functionnalities using anchor pictures, coded in Intra mode. All other pictures are called non-anchor pictures. In non-base views, random access can be provided if the prediction structure involves V-pictures, which are pictures that are not temporally dependent on any other picture in the same view, but which can be dependent on pictures in other views at the same time instant. In Figure 1.17, the P and B pictures at T0 and T8 in non-base views are all V-pictures. Instanteneous Decoder Refresh (IDR) pictures are natural random access points which correspond to anchor pictures in the base view and to V-pictures in non-base views (hence the commonly found denomination: V-IDR). Furthermore, MVC extends Skip mode to the view direction by introducing Inter-view Skip mode, and implements Illumination Change-Adaptive Motion Compensation (ICA MC) to compensate the natural illumination change between views that otherwise weakens the inter-view prediction.

For MVD formats, there is currently no existing standard that can code into a single bitstream all the texture and depth views of the 3D signal. The abovementionned standards can however be used to produce multiple bitstreams: the texture views and the depth views can be encoded independently using MVC and

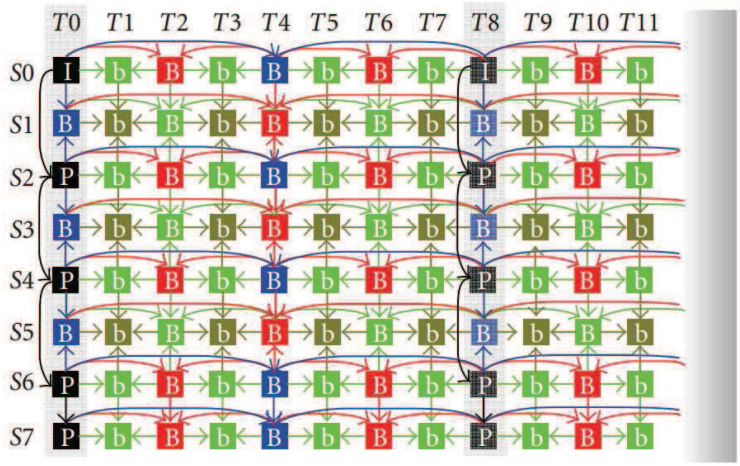


Figure 1.17: Typical MVC prediction structure

two bitstreams will thus be produced. An MPEG-C Part 3 or MPEG-4 Part 2 MAC encoding can otherwise be performed on every texture/depth pair hence producing as many bitstreams as the total number of views. Coding efficiency would not however be maximal since in each case, there would be unexploited signal correlations. A standard capable of jointly coding all the texture and depth views is thus needed.

Recent standardization activities

To answer this need, a Call for Proposal (CfP) for 3D video coding technologies was issued by MPEG in March 2011 [CFP11]. The CfP was answered in November 2011 with multiple contributions. Two standardization tracks, now directed by a joint collaborative team between ISO and ITU called JCT-3V, were later created: one for AVC-based solutions and another for HEVC-based ones. In the first track, the goal was to standardize an MVC-compatible extension including depth (called MVC+D), where the main target is to enable 3D enhancements while maintaining MVC stereo compatibility for the texture videos [VM13]. This involves only high level syntax change for the decoder to recognize and extract depth data. Macroblock-level changes to the AVC or MVC syntax, semantics and decoding processes are not considered to guarantee backward compatibility. MVC+D has reached Final Draft Amendment (FDAM) status following the 3rd JCT-3V meeting in January 2013. Another goal of the first track is to standardize an AVC-compatible extension (3D-AVC) that includes depth. In this approach, further coding efficiency gains could be obtained by improving the compression efficiency of non-base texture views and the depth data itself. However, in contrast to the MVC-compatible approach, this method requires changes to the syntax and decoding process for non-base (dependent) texture views and depth information at the block level [VM13]. 3D-AVC has

reached the Final Draft Amendment (FDAM) status in November 2013.

In the second track, a multiview extension of HEVC (MV-HEVC) including depth is being standardized. MV-HEVC provides backward compatibility with HEVC (the base view is encoded in HEVC without any additional tools). Only high level syntax is employed to code dependent texture and depth views by introducing inter-view pictures into the DPB to allow disparity compensated prediction (no block level coding tools will be considered). MV-HEVC produces two seperately decodable bitstreams, one for texture and the other for depth. It is expected to reach FDAM status in early 2014. Furthermore, a multiview extension of HEVC including depth and introducing block level changes is also being standardized under the name: 3D-HEVC. The coding efficiency of dependent texture views and depth data is increased using additional coding tools and syntax. Since this implies a redesign of the decoders, 3D-HEVC is not expected to be finalized before 2015 [Ohm13].

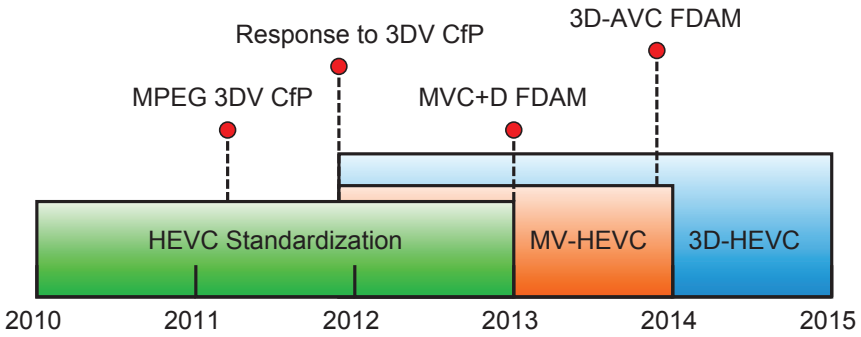


Figure 1.18: 3DV standardization timeline

Performance comparison between different 3D video coding schemes[1]

We consider two 3D data representations in the following tests: a first scenario where 9 views of 3D video sequences are coded and transmitted, and a second one where only 3 out of those 9 views (the leftmost, the center and the rightmost views), with their associated depth videos, are coded and transmitted, and the remaining 6 views synthesized at the receiver side using DIBR. We first compare the coding performance of HEVC simulcast, MVC, MV-HEVC and 3D-HEVC together in each scenario, then compare the two scenarios together per used codec.

For the first scenario, results are evaluated on each coded view using the Bjontegaard-Delta rate (BD-Rate) metric [Bjo01]. The rate and PSNR involved in each computation correspond to the bitrate needed to code the view and its PSNR compared to an uncompressed original view. For the second scenario, results are evaluated on the

[1]I would like to thank Antoine Dricot for performing the following tests during his internship at Orange Labs

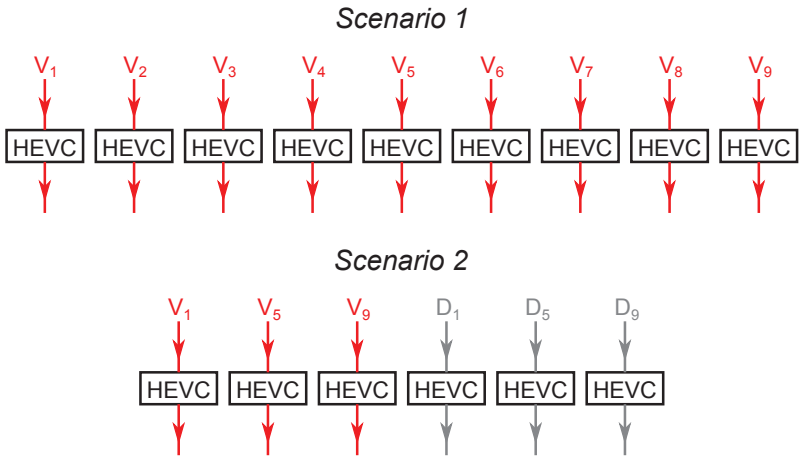


Figure 1.19: Coding structure of HEVC

coded texture views and on the synthesized views using BD-Rate as well. Three results columns thus appear in the results table: "Video PSNR / Video bitrate" where the bitrate considered is the sum of the 3 coded texture views bitrate, and the PSNR the average of the 3 coded texture views PSNR, "Video PSNR / Total bitrate" which differs from the previous column in the fact that the bitrate considered is the sum of the 3 texture and the 3 depth bitrates, and finally "Total PSNR / Total bitrate", where the PSNR considered is the average between the 9 views (3 coded + 6 synthesized views). In the results tables, a negative value represents a gain over the considered reference.

The reference softwares of the different standards used are the following: HM-10.0 for HEVC, JMVC-8.5 for MVC, and HTM-6.0 for 3D-HEVC and MV-HEVC. QPs were aligned in order to obtain good BD-Rate measures. For 3D-HEVC, the following texture/depth (respectively) QP combinations were used: (20/30, 25/34, 30/39, 35/42). For MVC and HEVC: (22/31, 27/36, 32/39, 37/42), and for MV-HEVC: (17/28, 22/31, 27/36, 32/39). The encoding structure for the different codecs is the following: for HEVC, each texture and depth view is coded independently in a Simulcast (SC) fashion as shown in Figure 1.19. For MVC and MV-HEVC, an hierarchical B-picture structure is used to code the views, as shown in Figure 1.20. In scenario 2, this coding structure is applied independently for the texture views and for the depth views hence producing two different bitstreams. Finally, for 3D-HEVC, the central view (base view) is coded first, and is used as reference to code the side views (dependent views). Inter-component (between texture and depth) dependencies also exist as shown in Figure 1.21. Note that in Figures 1.20 and 1.21, black and blue dashed lines represent respectively inter-view and inter-component dependencies. The coding order is indicated in yellow circles.

The coding configuration was also aligned among the different codecs. The GOP

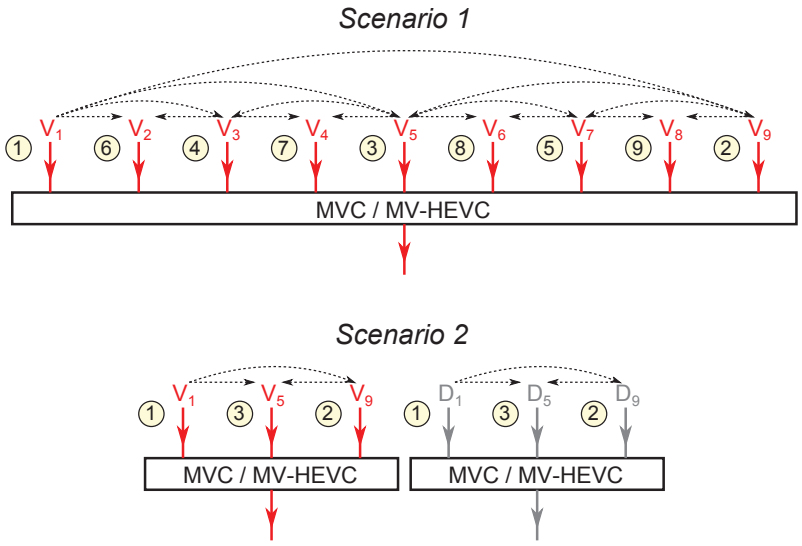


Figure 1.20: Coding structure of MVC and MV-HEVC

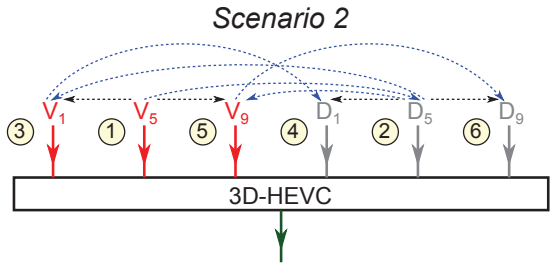


Figure 1.21: Coding structure of 3D-HEVC

size was set to 8, the Intra period was set to 24, and finally, the search range for motion estimation was set to 96. The sequences considered are defined in the Common Test Conditions for 3D-AVC and 3D-HEVC experiments, defined by the JCT-3V group [RMV13]. All frames of all sequences were coded. The renderer used to synthesize views is the one included in the HTM-6.0 package (VSRS-1DFast).

First, we compare the different codecs together per scenario. For scenario 1, the reference was an HEVC simulcast coding of the different views. Table 1.1 summarizes the results of MVC and MV-HEVC. 3D-HEVC cannot be used for scenario 1 because no depth videos are considered. MVC codes V1 in the main profile of H.264/AVC to provide backward compatibility, which explains the large loss compared to HEVC on this particular view. As the number of coded views increases inter-view correlations become more and more exploited in MVC which becomes eventually more performant than HEVC, starting from the 4^{th} coded view (V3 in display order), as shown in Table 1.1. On average, a 7.5% bitrate reduction is reported for this codec over HEVC. MV-HEVC codes V1 with HEVC for backward compatibility as well. Hence there is practically no difference with HEVC on this

Codec	Sequences	V1	V2	V3	V4	V5	V6	V7	V8	V9	Avg
MVC	Balloons	88.1	-31.9	-15.7	-34.2	21.3	-36.5	-13.9	-27.0	54.7	-1.4
	Kendo	82.8	-36.3	-19.2	-43.5	19.6	-43.9	-14.3	-28.5	56.3	-4.8
	Newspaper	75.0	-37.8	-24.3	-46.7	-5.6	-50.2	-29.8	-43.7	32.8	-16.3
	Average	**82.0**	**-35.3**	**-19.8**	**-41.5**	**11.8**	**-43.5**	**-19.3**	**-33.1**	**47.9**	**-7.5**
MV-HEVC	Balloons	-0.9	-71.0	-57.3	-47.2	-32.7	-43.5	-35.4	-38.7	-21.0	-37.9
	Kendo	-0.6	-70.7	-57.5	-48.4	-31.8	-42.7	-33.2	-33.1	-16.8	-36.6
	Newspaper	0.0	-66.7	-51.2	-46.4	-39.2	-45.4	-33.4	-36.0	-21.4	-37.0
	Average	**-0.5**	**-69.5**	**-55.3**	**-47.3**	**-34.6**	**-43.8**	**-34.0**	**-35.9**	**-19.7**	**-37.1**

Table 1.1: BD-Rate coding results of MVC and MV-HEVC vs. HEVC in scenario 1

Codec	Video PSNR / Video bitrate	Video PSNR / Total bitrate	Total PSNR / Total bitrate
MVC	55.6	62.2	73.2
MV-HEVC	-30.7	-29.0	-28.8
3D-HEVC	-35.3	-35.3	-45.6

Table 1.2: BD-Rate coding results of MVC, MV-HEVC and 3D-HEVC vs. HEVC in scenario 2

particular view (small differences come from syntax changes but the coding / decoding process is the same). Similarily to MVC, MV-HEVC exploits more inter-view correlations as the number of coded views increases, and eventually outperforms HEVC with a significant average gain of -37.1%. While MVC achieves some gains, its performance is always hindered by the more efficient coding tools implemented in HEVC. This problem does not exist in MV-HEVC; the gains result from solely exploiting inter-view correlations and are not countered by any loss.

Table 1.2 summarizes the coding results of MVC, MV-HEVC and 3D-HEVC in scenario 2. In this scenario, only 3 views are coded. However, as seen previously, MVC starts to outperform HEVC from the 4th coded view. Hence, in this second scenario, MVC brings losses compared to HEVC. MV-HEVC, on the other hand, outperforms HEVC and brings -28.8% gain. 3D-HEVC brings additional gains (-45.6%) because it introduces new coding tools to code depth data, and allows exploiting inter-component correlations. These functionnalities bring -23.6% gain over MV-HEVC as shown in Table 1.3.

Next, we compare, for a fixed codec, the coding results of using scenario 2 (MVD) instead of scenario 1 (MVV) to represent the 3D data. These results are summarized in Table 1.4. We can see that regardless of the codec used, representing the 3D data in an MVD format provides important coding gains with respect to an MVV representation. This is due to the fact that the amount of data to be encoded is far less important in MVD than in MVV (lesser texture views are coded and depth maps do not cost as much to code as texture views), and because the quality of

Sequence	**Total PSNR / Total bitrate**
Balloons	-22.7
Kendo	-22.0
Newspaper	-29.7
GT Fly	-4.1
Poznan Hall 2	-33.8
Poznan Street	-18.8
Dancer	-34.3
Average	**-23.6**

Table 1.3: BD-Rate coding results of 3D-HEVC vs. MV-HEVC

Codec	**Video PSNR / Video bitrate**	**Video PSNR / Total bitrate**	**Total PSNR / Total bitrate**
MVC	-46.1	-30.2	-23.7
MV-HEVC	-40.0	-32.0	-31.2
HEVC	-61.3	-56.6	-55.5

Table 1.4: BD-Rate coding results of scenario 2 vs. scenario 1, per used codec

the synthesis is maintained (there is no large PSNR drops, but this highly depends on the renderer used). The performance gap is smaller if MV-HEVC or MVC is used due inter-view coding tools that reduce the rate needed to code the views that otherwise in MVD would not need to be sent.

1.4 Conclusion

In this chapter, we have reviewed different types of 3D displays in Section 1.1. While it is expected that holoscopic and holographic displays will dominate the 3D market in the future, these technologies are, to this day, quite expensive for consumer commercialization and far from production maturity. Stereoscopic imagery still has some years ahead of it, and a number of problems with auto-stereoscopic displays can be tackled with an increased number of views projected on the display.

To this purpose, the MVD format (studied amongst other formats in Section 1.2 can be quite useful, as it allows synthesizing a large number of views at the receiver, given a few texture views and associated depth videos as input. As shown in Section 1.3, an MVD representation can be a better alternative than MVV as it allows to reduce the cost of coding the 3D data.

However, to this day, there are no standards capable of coding all texture and depth views of an MVD format, while exploiting in the process inter-view and inter-component dependencies for maximal coding efficiency, and producing a single bitstream for the data. 3D-HEVC is currently being standardized to answer this need.

As seen from the results of Section 1.3.2, it is currently the extension that offers the most coding gains for MVD representations (-41% gain over HEVC simulcast). The next chapter will detail 3D-HEVC for a better understanding of the coding tools used for dependent view and depth map coding.

Chapter 2

Coding tools in 3D-HEVC

Contents

In this chapter, we first describe the context which led to the creation of the current 3D-HEVC test model, then we present the coding structure 3D-HEVC utilizes. The improvement of video compression techniques within 3D-HEVC requires a fine understanding of the coding tools already present in the extension: hence, in a third section, we first describe the classic HEVC coding of the base view (which provides monoscopic backward compatibility). Dependent views and depth videos are coded

with additional coding tools that will be presented respectively in a fourth and fifth section. Non-normative encoder controls are further explained in section 6. Section 7 finally concludes this state-of-the-art.

2.1 Context

With HEVC reaching Final Draft International Standard (FDIS) status in January 2013, standardization efforts have now shifted towards a 3D extension of HEVC. Following an MPEG CfP on 3D video coding technologies in March 2011, a total of 11 HEVC-compatible proposals have been submitted and compared during the 98th MPEG meeting in Geneva (November 2011). Specifically, proponents had to submit results in a two-view and a three-view scenario. In the first scenario, two views of each of the eight sequences considered in the CfP had to be coded and one intermediate view had to be synthesized between the two, whereas in the second scenario, three views had to be coded and fifteen (twenty-three for some sequences) had to be synthesized between each pair. Results were compared against an HEVC simulcast coding of each texture and depth views. Four rate points were specified in the CfP for each scenario. All proposals and anchors had the same bitrate, such that a Mean Opinion Score (MOS) subjective quality comparison could be performed as well. The viewing results of the two-view scenario on a stereoscopic display (SD) with polarized glasses and the results of the three-view scenario on an auto-stereoscopic 28-view display (ASD) are given for each rate point as the average MOS value over all eight sequences considered. For the two-view scenario, the stereo pair viewed was composed of the synthesized view and one of the two (left and right) reference views, whereas the 28 views shown on the ASD were selected, amongst the reference and synthesized views, in a way to guarantee visual comfort and provide sufficient depth range (a sufficiently wide baseline distance was imposed between the 1st and the 28th view). Figure 2.1 [VM13] shows the MOS values of the best performing proposal and the HEVC anchor.

The best performing proposal achieved 57% and 61% bitrate reduction (measured in BD-Rate) in the two-view and three-view scenarios respectively. A corresponding test model, which would later be called 3D-HEVC Test Model, was then drafted, and the software of the proposal became the reference software on top of which further tool experiments were to be performed. The standardization process of 3D-HEVC is now directed by JCT-3V which holds trimestrial meetings, during which new coding tools are evaluated and adopted. At each meeting, a new version of the software, which includes all meeting adoptions, and following the naming convention "HTM-X.0", is maintained.

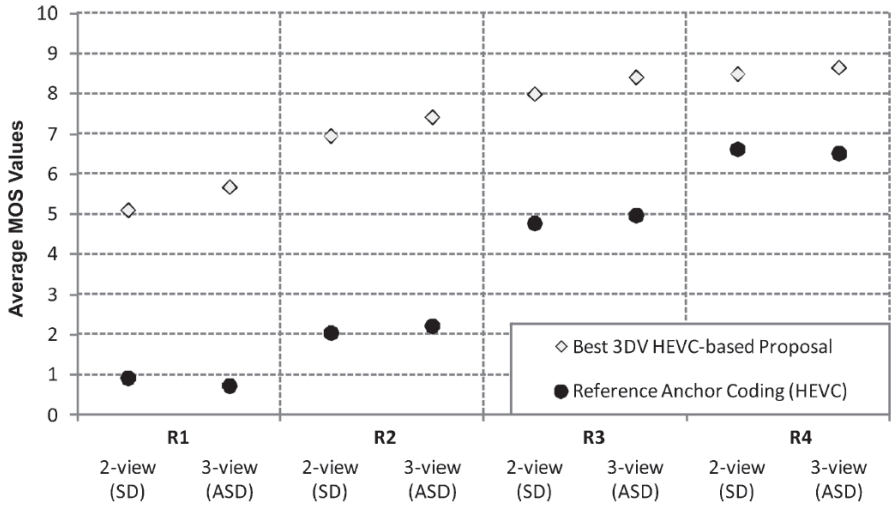


Figure 2.1: Average MOS accross 8 sequences of the best performing HEVC-compatible proposal and of the HEVC anchor

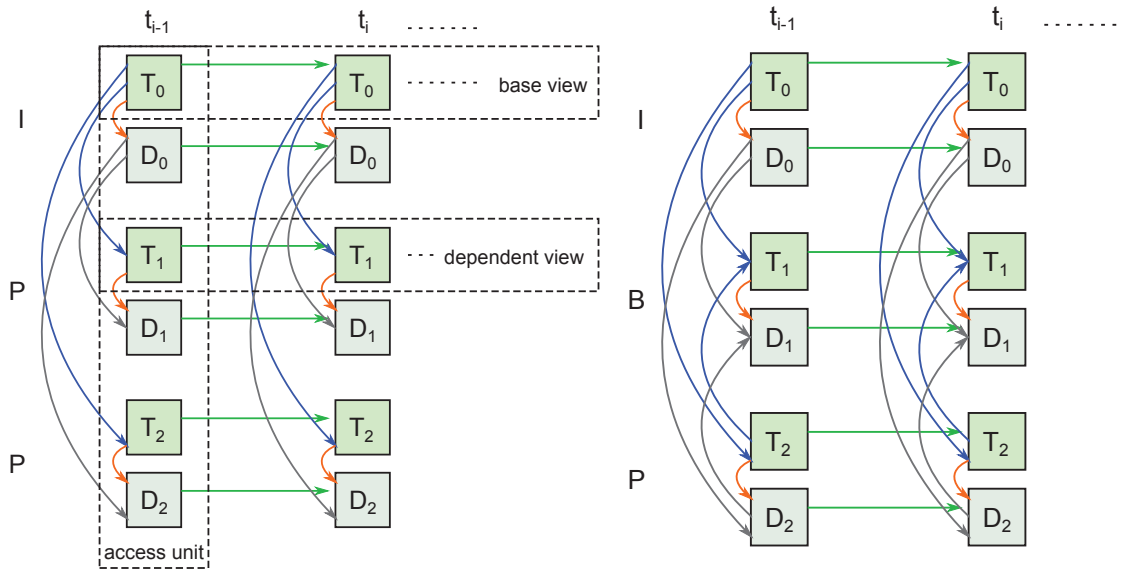
2.2 Coding structure

Data coded in 3D-HEVC is processed by access units. An access unit is composed of a base texture view and other dependent texture views, along with their associated depth videos, at one time instant, as shown in Figure 2.2. All the data inside one access unit is coded, then another access unit at the next time instant is processed and so on.

Inside an access unit, the base view is always coded first, followed by dependent views where inter-view redundancies are exploited. In each view, the texture component is coded before the depth component. Indeed, depth videos are coded by exploiting redundancies with the associated texture (also called inter-component dependencies). A non-normative tool, called Flexible Coding Order (FCO), allows to change the coding order for dependent views so that depth is coded before texture, although this implies disabling specific depth-using-texture coding tools which may reduce the coding efficiency of depth data.

Furthermore, a dependent view in an access unit is coded using another view as reference view. The CTCs defined by the JCT-3V group impose the base view as reference for all dependent views. However, an hierarchical IBP configuration is possible using 3D-HEVC, as shown in Figure 2.2. In this Figure, green, blue / black and orange lines represent respectively temporal, inter-view and inter-component dependencies exploited in 3D-HEVC.

The two following sections list the different coding tools which were adopted in 3D-HEVC at the time of developping the coding methods proposed in the following chapters. Since 3D-HEVC is not yet finalized at the time of writing this thesis,



(a) IPP configuration (CTC) (b) IBP configuration

Figure 2.2: Different coding structures in 3D-HEVC

some of these tools might have been modified, others added and others completely removed. Hence, it is important to note that the following sections do not describe the final version of the 3D-HEVC draft.

2.3 Coding of the base view

This section will basically detail the HEVC standard since it is used to code the base view. The HEVC structure and tools are also used to code the dependent views and the depth videos, along side additional coding tools designed to further increase their coding efficiency.

2.3.1 Coding structure

HEVC uses an advanced quadtree-based coding approach [HMK+10], wherein a picture is divided into coding tree units (CTUs). Those are the equivalent of macroblocks in previous video coding standards. Each CTU can then be split into four coding units (CU), and each CU can be further split into four CUs, and so on. A maximum CTU size and a maximum depth level are set to limit the CU split recursion. This quadtree approach allows having different block sizes inside the same image, making the encoding more adapted to the image contents. Note that it is at the CU level where a specific coding mode (Intra, Inter, or Merge) is chosen: the Intra coding mode predicts a CU using samples from previously coded CUs in the same image, whereas the Inter coding mode predicts the current CU with samples

located in a previously coded frame. Merge mode allows to make an Inter prediction of a CU using the motion parameters of other neighboring CUs. Skip mode is a particular Merge mode where no CU residual is sent. These modes are further detailed in Sections 2.3.2 and 2.3.3.

A CU can further be partitioned into prediction units (PUs) where all PUs are coded in the same coding mode, but where each has its own prediction information (Intra mode, motion vectors, etc.). The PUs however cannot be further partitioned. Different coding modes imply different possible PU partitions:

- Merge: 2N×2N
- Intra: 2N×2N, N×N (only possible at the maximum depth level)
- Inter: 2N×2N, 2N×N, N×2N, 2N×nU, 2N×nD, nL×2N, nR×2N and N×N (only possible at the maximum depth level)

These partition sizes are shown in Figure 2.3. A 2N×2N partitionning gives a single PU which is the same size as the CU, whereas a 2N×N partitionning for instance gives two rectangular PUs, where each is half the size (same width, halved height) of the CU.

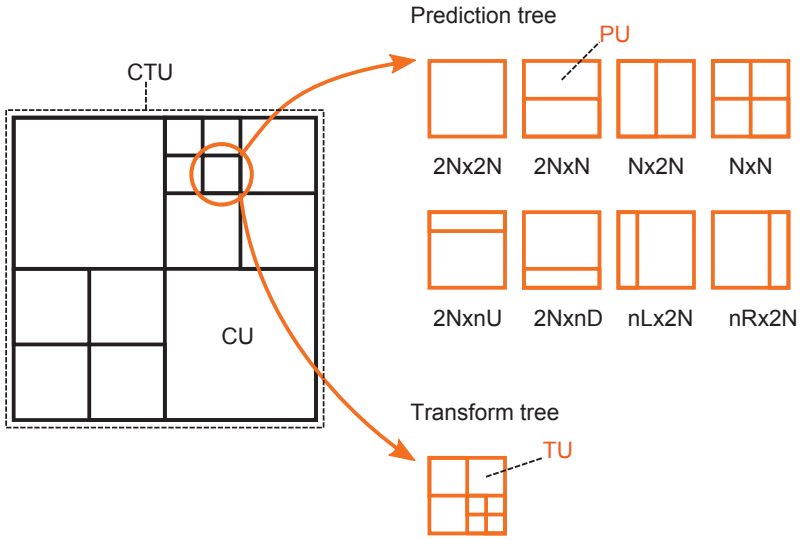


Figure 2.3: Quadtree structure of a CTU and possible partition shapes

Each CU is also the root of a transform tree, known as residual quadtree (RQT). A CU can be partitionned into multiple transform units (TUs). A TU defines the size of the transform and quantization to be applied to the corresponding area of the CU. TU sizes vary between 4×4 and 32×32. TUs in Intra mode are exclusively square partitions. In Inter, rectangular TUs are possible (of size 4×16, 16×4, 8×32 and 32×8) in order to avoid applying the transform accross block edges which will

produce high frequency coefficients in the residual. Note that the RQT is completely independent from the prediction tree: a TU can cover one or more PUs.

In HTM, for each CU, coding mode and partition size, a non-normative Rate Distortion Optimization (RDO) process tests all possible TU partitionnings. A Lagrangian cost is computed for each coding mode / partition size / RQT combination. The cost of splitting the CU is also computed and the configuration yielding the lowest cost is selected for that CU.

2.3.2 Intra coding mode

HEVC introduces a total of 34 Intra modes, 32 of which are directional. The two non-directional modes are DC and Planar. Each mode tries to form a prediction of the current PU using reference pixels belonging to the neighboring row and column, respectively above and left of the current PU. The Intra modes are ordered according to the direction angle. Vertical and horizontal directions are associated with low Intra indices (1 and 2 respectively), while finer angles have higher Intra indices, as shown in Figure 2.4.

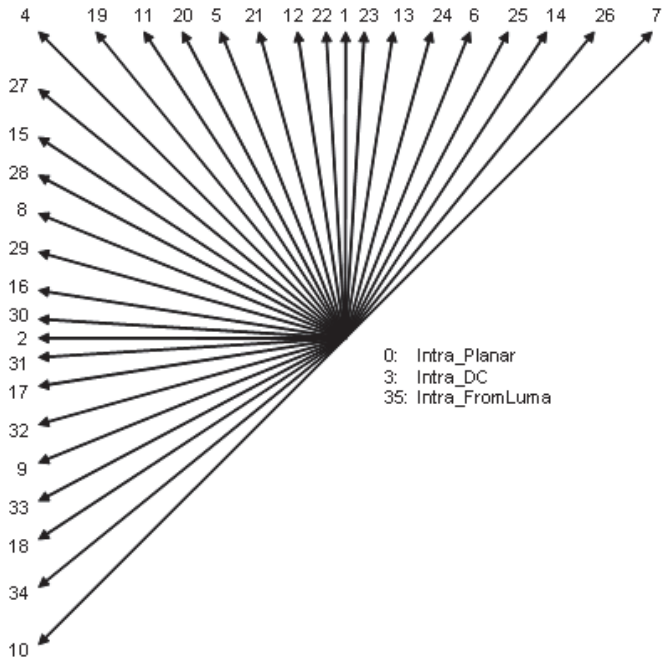


Figure 2.4: List of Intra modes in HEVC

The DC mode forms a prediction where each sample has the same value, which is the mean of the reference pixels. It is efficient to code uniform areas of the image. The Planar mode involves bi-linear interpolation: each pixel is interpolated using the bottom row and the rightmost column of the current PU which are respectively

substituted with the bottom-left and above-right causal reference samples. This prediction mode, which is particularily adapted to smooth textures in the image, solves a problem with classical Intra prediction which imposes a non-negligible distance between the reference row and column and the pixels at the bottom right of the PU.

Coding the Intra mode of a PU requires 6 bits at most. In order to reduce this cost, the Most Probable Mode (MPM) technique has been introduced. Each Intra coded PU has at most two MPM candidates, which are the Intra modes of the PU above and the PU to the left of the coded PU if they are available. If the best Intra mode of the coded PU matches one of the MPM candidates, only a flag signaling the use of an MPM candidate and another flag signaling which MPM candidate matches the best mode are transmitted, making it a total of 2 bits used for signalling instead of 6. This technique thus effectively avoids coding the Intra mode itself and reduces the Intra mode signaling bitrate.

2.3.3 Inter coding mode

Inter prediction involves two aspects: motion compensation and motion vector coding. For the first part, HEVC supports motion vectors with units of one quarter and one eighth of the distance between luma and chroma samples respectively. Hence, an 8-tap and a 7-tap DCT-based interpolation filter is used for luma samples for respectively the half-pel and the quarter-pel positions, whereas a 4-tap filter is used for chroma samples for all eighth-pel positions [SOHW12].

For the second part, two tools exist. The first tool is called Advanced Motion Vector Prediction (AMVP) [LJPP08, MHK+10]. In AMVP, a list of 2 motion vector predictor (MVP) candidates is established. The 2 candidates are selected amongst the MVs of the PUs covered by different neighboring positions: there are 5 spatial positions and 2 temporal ones, as depicted in Figure 2.5. Note that the temporal candidates are derived from a collocated picture in the temporal reference pictures of the current slice. A first spatial candidate is derived from A_0 and A_1 (the two PUs covered by these positions are checked in this order, and the first MV found is selected). A second one is derived from B_0, B_1 and B_2. If neither or only one of the two spatial candidates is unavailable, or if the two candidates are identical, then a temporal candidate is derived from C_0 and C_1 and added to the list. If the list still contains less than 2 candidates, it is filled with zero MVs. Only a residual between the MV and the MVP is sent in the bitstream, along with an index signalling which MVP candidate out of the two was selected. If the two candidates are identical after the list construction, no index is sent.

The second tool is called Merge mode [HOB+12]. Merge mode allows a PU to inherit the motion information (motion vectors + reference indices, in both reference

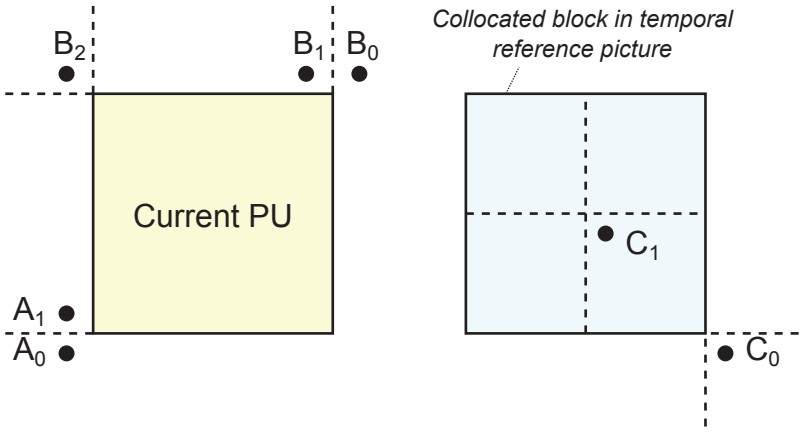


Figure 2.5: AMVP candidates in HEVC

lists L0 and L1) of neighboring spatial or temporal PUs. A list of candidates is established here as well, composed of the motion information of the PUs covered by the same positions as in AMVP. Only the index of the selected candidate is sent in the bitsream. The candidate list in Merge mode is composed, in order, of four spatial candidates: A_1, B_1, B_0, A_0 (and B_2 if one of these four is unavailable) and one temporal candidate. A pruning process is performed within the 4 spatial candidates to remove redundant vectors [BLU11]. Finally, if some of these 5 candidates are unavailable (the PU corresponding to the position falls outside the slice, or is Intra-coded, or the candidate is redundant), a secondary list of candidates is computed. These candidates are then appended to the list so that the total number of candidates is always 5. The candidates in that secondary list are, in order, combined candidates from mixed primary vectors of both reference lists, and zero-vector candidates, each having a different reference index.

Note that a CU is said to be coded in Skip mode when it is coded in Merge mode and when no pixel residual for that CU is sent. The reconstructed signal in this case equals the prediction signal.

2.4 Coding of the dependent views

Aside from Disparity-Compensated Prediction (DCP) used by the dependent views to exploit inter-view correlations, and which does not require any block level changes, there are a couple of tools that are implemented in 3D-HEVC to increase the coding efficiency of these views. Note that these tools are not implemented in MV-HEVC which only provides DCP.

2.4.1 Inter-View Motion Prediction

Inter-View Motion Prediction (IVMP) improves the coding of the motion vectors of an Inter coded PU by exploiting inter-view motion vector correlations. It basically consists of inserting a multiview candidate in the Merge and AMVP candidate lists, making it a total of 6 and 3 candidates respectively in each list [TWCY13]. First a disparity vector, called DV_{IVMP} in this section, pointing to a reference view is derived for the current PU. The derivation of this DV is performed using a neighboring search process called NBDV which will be described next. Once the DV is found, it is used to find a corresponding PU, PU_{ref} in the reference view, as shown in Figure 2.6.

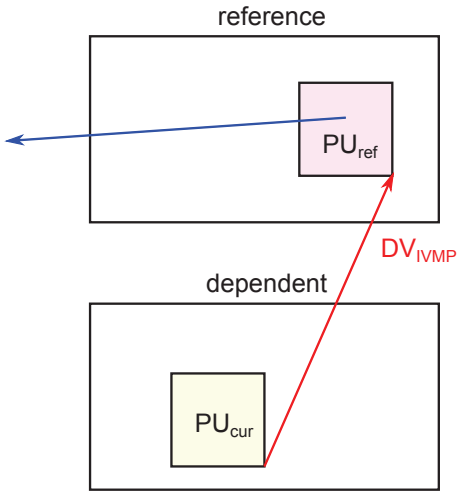


Figure 2.6: Inter-view motion prediction

DV derivation process

Obtaining DV_{IVMP} involves a derivation process called Neighboring Disparity Vector (NBDV) [ZCK12]. NBDV is a simple search process across neighboring positions. The PUs covered by these positions are checked if they are coded using disparity-compensated prediction (DCP), in which case they have a DV, and the first DV found is selected as the final DV used for IVMP. The positions are the same as the Merge / AMVP positions depicted in Figure 2.5.

Spatial positions are checked first, in the following order: A_1, B_1, B_0, A_0, B_2. A PU covered by one of the 5 spatial positions can have up to two vectors, one from each reference list (list 0 and list 1) and they are both checked if they are DVs. Temporal positions are checked next in the following order: C_0, then C_1. Those positions are checked in a maximum of two temporal reference pictures. Furthermore, if a neighboring spatial PU was coded in MCP (*i.e*, the PU has a MV in a specific reference list, not a DV), its MV could have been computed using IVMP which

necessarily involves the derivation of a DV. Indeed, the MV could have been inherited from the multi-view candidate in Merge mode or predicted using the multi-view MV predictor in AMVP. Constructing the multi-view candidate in both lists (Merge & AMVP) requires a prior derivation of a DV, which would then be linked to the current MV. These special DVs, called DDVs [SKY12], are also checked after the temporal neighbors in the following order A_0, A_1, B_0, B_1, B_2. If no DV can be found after this entire search process, DV_{IVMP} is set as (0,0).

IVMP in AMVP

In AMVP, if the motion vector of PU_{ref} points to a picture in the same access unit as the one pointed to by the current motion vector, it is added in the first position in the AMVP candidate list. If the currently coded vector is a disparity vector, then DV_{IVMP} is added instead. In both cases, no redundancy check with other candidates is performed.

IVMP in Merge

The multiview candidate in the Merge list is the motion vector of PU_{ref}, if it exists (PU_{ref} is Inter coded and falls inside the base slice). If not, the multiview candidate is DV_{IVMP} (it will be a disparity candidate). Indeed, there is always a preference for this candidate to be a motion vector rather than a disparity vector.

2.4.2 Inter-view Residual Prediction

Inter-view Residual Prediction (IVRP) also involves the derivation of a DV, called here DV_{IVRP}. The same derivation process as in IVMP, NBDV, is invoked to obtain this DV, which will thus equal DV_{IVMP}. DV_{IVRP} will point to a corresponding PU in the base view: PU_{ref}. The residual of PU_{ref} is used to predict the residual of the current PU: it will be substracted from the current residual, and only the remaining residual is transformed and quantized. IVRP is only applied, systematically, for 2N×2N PUs coded in Merge mode with the multiview Merge candidate, or with Skip mode. Hence no explicit block level signalling is used. A flag signalling the use of IVRP at the slice level is sent nevertheless for dependent views, depending on the decoded information in the base view.

2.4.3 Illumination Compensation

A discrepancy between different views in a multiview setting may exist due to illumination changes (for instance, one camera is directed more towards the sunlight and hence, it captures more light than another adjacent camera). This discrepancy

often leads to bad inter-view prediction signals. To correct this, Illumination Compensation (IC) has been introduced in [LJS+12] and later adopted in the 3D-HEVC working draft.

IC uses a linear model with two parameters a and b to correct the inter-view prediction signal. If the current PU, $\mathrm{PU}_{\mathrm{cur}}$, is DCP coded with a DV pointing to $\mathrm{PU}_{\mathrm{ref}}$ in a reference view, the residual r, typically computed as $PU_{\mathrm{cur}} - PU_{\mathrm{ref}}$ would be computed, using IC, as $PU_{\mathrm{cur}} - (a \cdot PU_{\mathrm{ref}} + b)$. The model parameters, a and b, are computed using reconstructed neighboring samples. Hence they are not transmitted in the bitstream. A flag is sent at CU level indicating whether IC is applied or not. However, a simplified encoder can just check IC for Merge mode in 2N×2N partition, and if IC is not selectionned for this mode, IC is disabled for all other Inter modes. IC brings 0.6% bitrate reduction for synthesized views. Details about the test conditions used and the way these gains are evaluated can be found in Section 4.2.

2.4.4 View Synthesis Prediction

View Synthesis Prediction (VSP) [TZV13] uses the reference decoded depth to increase the coding efficiency of a dependent view. By imposing this dependency, significant coding gains can be achieved at the price of losing stereo compatibility: decoding only a stereo representation of the video, composed of a reference view and a dependent view will not be possible if VSP is enabled, because decoding the dependent view requires the prior decoding of the reference depth. VSP involves a DV, $\mathrm{DV}_{\mathrm{VSP}}$, obtained through a DV derivation process such as NBDV. But unlike IVMP or IVRP, $\mathrm{DV}_{\mathrm{VSP}}$ is used to find the **depth** PU in the reference view, $\mathrm{PU}^{\mathrm{d}}_{\mathrm{ref}}$ corresponding to the current texture PU in the dependent view, $\mathrm{PU}_{\mathrm{cur}}$. $\mathrm{PU}^{\mathrm{d}}_{\mathrm{ref}}$ is then processed by sub-blocks of size 4×4. For each sub-block, a depth value is determined as the maximum value of the four corners of the sub-block. This value is transformed into a disparity vector, which will then be used to compensate the corresponding 4×4 sub-block in $\mathrm{PU}_{\mathrm{cur}}$, as shown in Figure 2.7. Although it might increase coding efficiency, disparity compensation is not performed on a pixel basis because it would require, in hardware, developping a new module to compensate each pixel. 4×4 is the smallest block size capable of being compensated using the already existing modules in HEVC.

This mode is introduced as a new VSP candidate in the Merge candidate list. The advantage of this approach is that the current PU is compensated using a disparity vector field composed of a DV for each 4×4 sub-block. This semi-dense approach (a true dense approach would associate a DV with each pixel) accounts for having objects of different depth (and hence different disparities) inside the same PU and

thus improves the inter-view prediction signal. Significant gains of more than -3.0% on coded views and -0.8% on synthesized views with only a 1% and 3% encoder and decoder runtime increases (respectively) are reported for this tool.

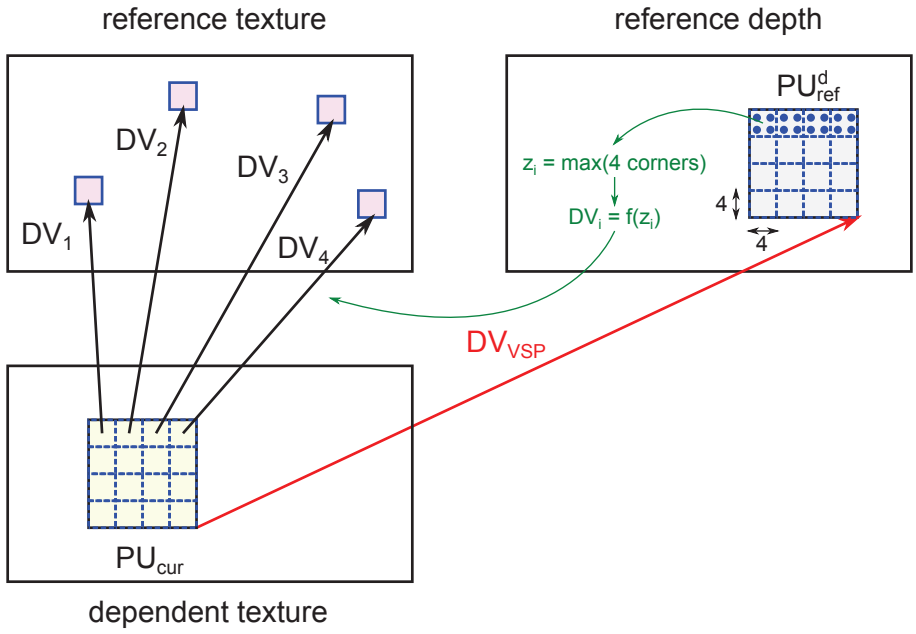


Figure 2.7: View synthesis prediction

2.5 Coding of the depth videos

Additional tools have been introduced in 3D-HEVC to increase coding efficiency of depth data. In Intra, several new modes have been added: Depth Modeling Modes (DMM), Region Boundary Chain coding (RBC), and Simplified Depth Coding (SDC). The signaling of the depth Intra modes is different than the one of the texture Intra modes, which still follows the HEVC recommendation. Indeed, when a texture CU is Intra coded, an Intra direction (from 0 to 33) is read from the bitstream for each PU, whereas when a depth CU is Intra coded, a new parameter called **depth_intra_mode** is read first for each PU. Then additional parameters are optionally read according to the depth Intra mode. Table 2.1 summarizes the possible depth Intra modes. The rest of the section describes these additional tools.

2.5.1 Depth Modeling Modes

Four new Intra modes, called Depth Modeling Modes (DMM) may be used to code depth data. In DMM 1, the depth PU is approximated by two constant regions, R_1 and R_2, separated by a straight line (wedgelet). For each region R_i, a parameter

Depth Intra Mode	Description
INTRA_DEP_SDC_PLANAR	Signals SDC mode with Planar
INTRA_DEP_NONE	Signals the use of a normal HEVC Intra direction, which is read next
INTRA_DEP_SDC_DMM_WFULL	Signals SDC mode with DMM 1, whose partition information are read next
INTRA_DEP_DMM_WFULL	Signals DMM 1 whose partition information are read next
INTRA_DEP_DMM_CPREDTEX	Signals DMM 3
INTRA_DEP_DMM_WPREDTEX	Signals DMM 4
INTRA_DEP_CHAIN	Signals RBC mode whose chain codes are read next

Table 2.1: Depth Intra modes

called region constant (P_i), is defined as the average of the values of the pixels covered by R_i and is sent in the bitstream to the decoder. Partition information is also sent, consisting in the start and end points of the straight line separating the two regions.

In DMM 2, P_1 and P_2 are sent but the partition information is not. Instead, it is inferred from neighboring PUs. However, the resulting partitioning may not be adequate for the current PU. Hence, an offset E_{off} is introduced that corrects the end point of the straight line, as shown in Figure 2.8. E_{off} is sent in the bitstream as well.

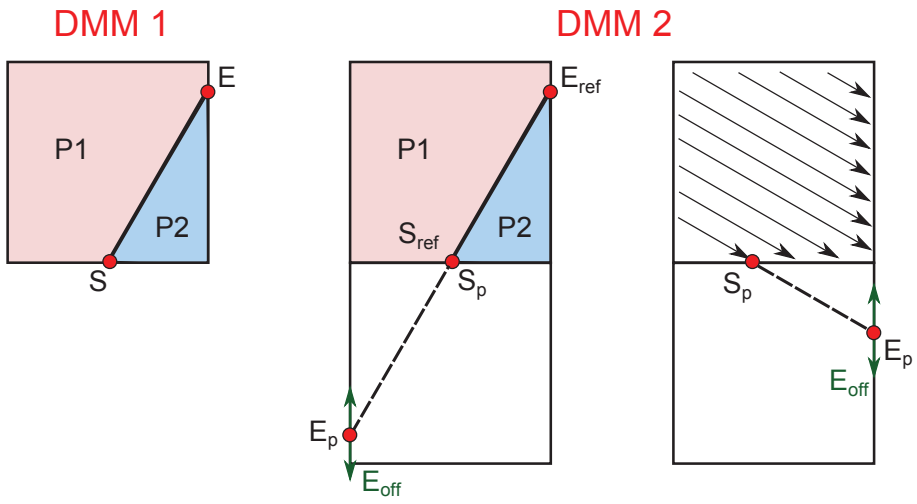


Figure 2.8: Depth modeling modes 1 & 2

DMM 3 also approximates the current PU by two constant regions, although

here, the partition information is inherited from the texture itself. Indeed, a simple thresholding of the texture Luma samples allows to divide the texture block into two separate regions. The resulting partitioning is used as partition information for depth. No start and end points are thus computed in this mode. Since we are dealing with reconstructed and not original textures, the process is decodable and the partition information does not need to be signaled. The constants of the two regions are sent though.

DMM 4 is the same as DMM 3, except that the PU is divided into three constant regions, as shown in Figure 2.9 (this is called Contour partitioning as opposed to Wedgelet partitioning where the block is divided into two constant regions as in DMM 1, 2 and 3).

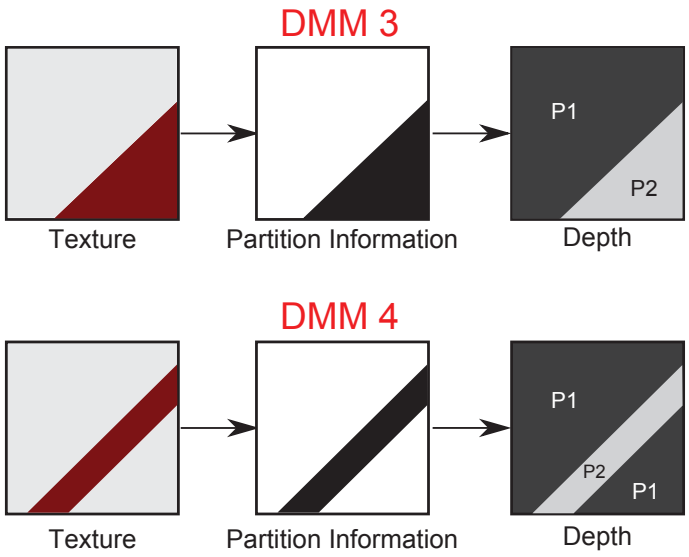


Figure 2.9: Depth modeling modes 3 & 4

2.5.2 Region boundary chain coding

The Region Boundary Chain (RBC) [HSYSg12] coding mode is an Intra mode that, just like DMM 1, tries to segment a depth block into two regions. The only difference is in the way the partition information is signalled. No start and end points are transmitted, but rather chain codes that, read one after the other, describe an arbitrary edge (not necessarily a straight line).

Basically, edges are detected at the encoder by comparing horizontally and vertically adjacent samples. If the difference is higher than a specified threshold, an edge is detected. Then, the coordinates of the starting point of the edge are sent in the bitstream, followed by a series of angles (chain codes) to direct the edge from the starting point to the end point. Region constants P_1 and P_2 are also sent as in

DMM 1.

RBC's distinctive feature is that it can well describe even a curved depth pattern (unlike DMM 1) that has weak or little correlation with the corresponding texture pattern (unlike DMM 3 or 4). However, this comes at the cost of signalling chain codes. RBC brings -0.3% gain on synthesized views with 2% encoder runtime increase.

2.5.3 Simplified Depth Coding

Simplified Depth Coding (SDC) [Jag12b] is another Intra mode, wherein a depth PU can only be predicted by DMM 1 or the Planar mode (a flag signalling which configuration was selected is sent in the bitstream). In SDC, a single residual index for each segment (in Planar there is only one segment whereas in DMM 1, there are two), $\mathrm{i_{resi}}$, is sent instead of a residual signal for the whole PU (a residue for each sample of the PU). It is computed a such:

$$i_{resi} = I(d_{orig}) - I(d_{pred}) \tag{2.1}$$

where I(·) indicates a depth lookup table (DLT) mapping of each depth value. $\mathrm{d_{pred}}$ is, for the Planar mode, the average value of the four corners of the predicted PU, and for DMM 1, the average value of the sub-set of the four corners that are available in each segment of the predicted PU, as shown in Figure 2.10 [ZTWY13]. In Figure 2.10(a):

$$d_{pred} = \frac{p(0,0) + p(7,0) + p(0,7) + p(7,7)}{4} \tag{2.2}$$

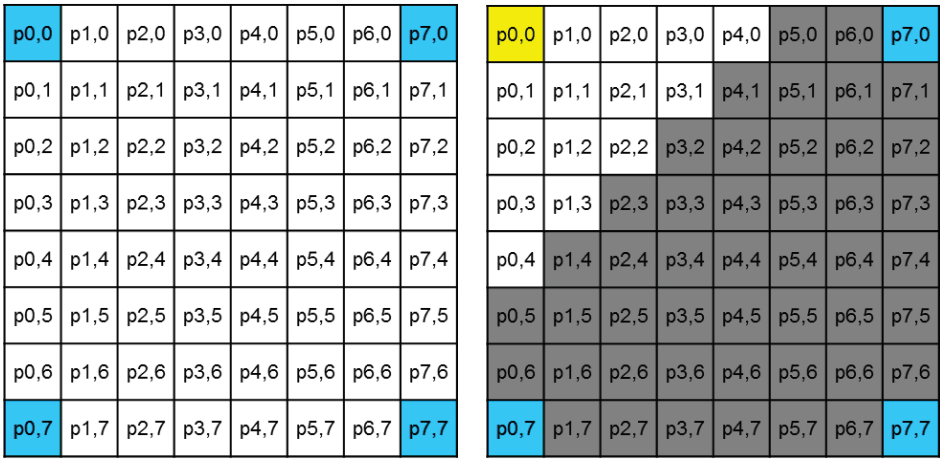
and in Figure 2.10(b):

$$d_{pred}[0] = p(0,0) \tag{2.3}$$

$$d_{pred}[1] = \frac{p(7,0) + p(0,7) + p(7,7)}{3} \tag{2.4}$$

The computation of $\mathrm{d_{orig}}$ is non-normative. In HTM-8.0, it is identical to the computation of $\mathrm{d_{pred}}$ but with the original version of the PU used instead of the predictor. The advantage of the DLT used is the reduced bit depth of the residual index for sequences with reduced depth value range (e.g. they have estimated depth maps where not all depth values are present). However, it still must be transmitted to the decoder to perform the inverse mapping. The decoder receives $\mathrm{i_{resi}}$ and the DLT, and reconstructs $\widehat{\mathrm{d}}_{\mathrm{orig}}$

$$\widehat{d}_{orig} = I^{-1}(i_{resi} + I(d_{pred})) \tag{2.5}$$



(a) SDC Planar (b) SDC DMM 1

Figure 2.10: Computing d_{pred} in SDC

A residual value is then computed as:

$$\widehat{d}_{resi} = \widehat{d}_{orig} - d_{pred} \tag{2.6}$$

And finally, the PU is reconstructed using the computed residual value, and the predictor signal $P_{x,y}$.

$$\widehat{P}_{x,y} = P_{x,y} + \widehat{d}_{resi} \tag{2.7}$$

SDC brings -0.5% gain on synthesized views in a random access configuration, and -2.0% in an all-intra configuration (all frames coded in Intra), with respectively 6% and 22% encoder runtime increase and 2% and 29% decoder runtime decrease.

2.5.4 Motion parameter inheritance

The motion vector correlations between the texture and the depth component are exploited with the Motion Parameter Inheritance (MPI) tool for depth coding. In MPI, the motion vectors and, when the texture CU is smaller than the depth CU, the quadtree coding structure of the colocated texture CU, are considered for inheritance in the depth CU as an additional Merge candidate. This introduces a dependency when parsing the depth, because of the need to check the sizes of both the texture and depth CUs before deciding to parse the split flag.

Note that MPI, DMM 3 and DMM 4 are all inter-component coding tools for depth data, which bring altogether -1.8% gain on synthesized views with respectively 6% and 7% encoder and decoder runtime increases [JL12].

2.6 Encoder control

Since depth videos are not displayed on screen but rather used to synthesize intermediate views after decoding, the RDO process should consider distortion directly on synthesized views in order to optimize depth coding for the quality of what actually matters: the synthesized views. In HTM, when coding depth, a non-normative method called View Sytnhesis Optimization (VSO) computes the Synthesized View Distortion Change (SVDC) [TSMW12] caused by a change at the level of the depth block being coded. The SVDC metric is computed by actually synthesizing in-loop the part of the view affected by the depth block change. It is later used by the RDO process in the Lagrangian cost computation instead of the usual depth block distortion as such: J = SVDC + λR to make decisions for depth, except in motion estimation where conventional RDO is still used to reduce the computational complexity.

Figure 2.11 shows how the SVDC metric is computed. A depth frame s_D is

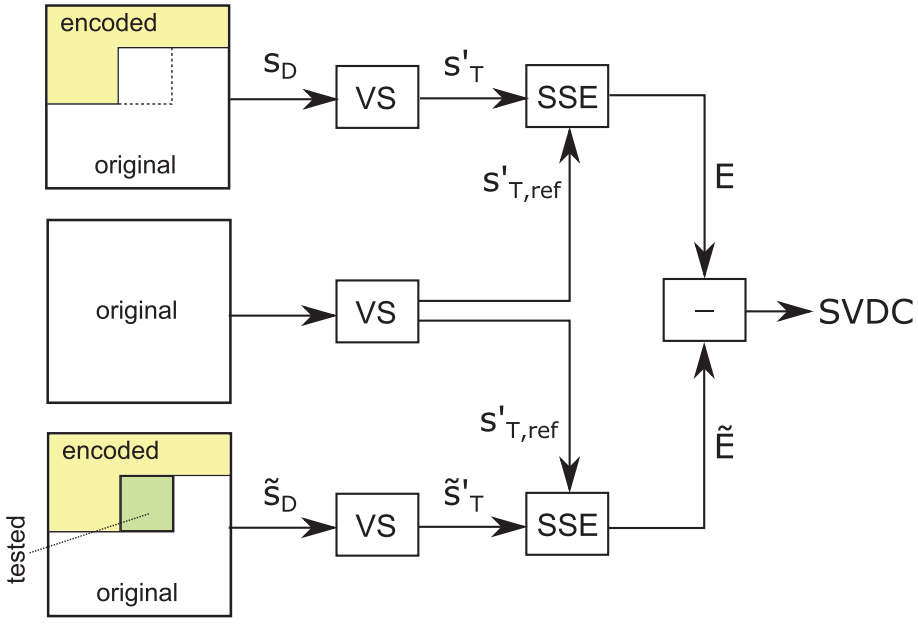


Figure 2.11: Computing the SVDC

considered, where the depth block being tested and all subsequent blocks are in their original form, and all previous blocks are already coded and reconstructed blocks. Another depth frame $\widetilde{s}_D$ is formed just like s_D, only the depth block being tested is replaced with its reconstructed form according to the test being performed. s_D and $\widetilde{s}_D$ are used to synthesize two texture views s'_T and $\widetilde{s}'_T$. Another view $s'_{T,ref}$ can be synthesized using the original depth and texture frames. Computing the SSE between s'_T and $s'_{T,ref}$ gives E, and between $\widetilde{s}'_T$ and $s'_{T,ref}$ gives $\widetilde{E}$. The SVDC metric is the difference $\widetilde{E} - E$.

The method is characterized by the fact that it considers occlusions and disocclusions in the synthesized views. In addition, it is relative to a block, and it allows partial distortions to be additive: if the block is divided into four sub-blocks, which is a common operation in quadtree codecs such as HEVC, the sum of the distortions caused by a change in each sub block equals the distortion caused by the corresponding change in the whole block. This assumption holds for SVDC because only the change in the synthesized view distortion caused by a change in the depth block is considered, and not the total synthesized view distortion itself.

2.7 Conclusion

In this chapter, we have presented an overview of all coding tools used to code the base view, the dependent views, and the depth data of a 3D-HEVC input video signal. Intensive research efforts are being put to further improve the coding efficiency of 3D-HEVC data, concretized every three months in a JCT-3V standardization meeting. In this thesis, we developped new coding tools on top of 3D-HEVC designed to increase the coding efficiency of dependent views, synthesized views, and depth videos in 3D-HEVC. The next three chapters will detail original yet conventional coding approaches, in the sense that they strictly follow standardization constraints in terms of complexity, runtime, memory usage, and practicality. In the final two chapters, we describe unconstrained approaches where the goal is to achieve significant coding efficiency regardless of the additional complexity the tools imply. These approaches would serve as a stepping stone to future research projects in this field.

Chapter 3

Texture Intra mode inheritance for depth coding in 3D-HEVC

Contents

In this chapter, we present an inter-component coding method aimed at increasing the coding efficiency of depth data in 3D-HEVC, by exploiting correlations with the associated texture. Namely, the Intra direction or mode of the collocated texture block is conditionally inherited and used as a predictor for the current depth block's Intra mode. The inheritance is driven by a specific criterion, which estimates how much the two Intra modes are supposed to match. Inheritance is only performed when a metric, quantifying the selected criterion, is higher than a certain threshold.

In the first section of this chapter, we motivate our method with a preliminary study in which we analyze the texture and depth Intra modes in 3D-HEVC. Then, we present the core algorithm of our method, and a first criterion which we use to drive the inheritance: *GradientMax*, along with the bitrate reductions it brings. A second criterion, *DominantAngle*, is then introduced and detailed with its respective coding results. These results are interpreted, and a comparison with the first criterion is then made, concluding the study.

3.1 Motivation

When analyzing a depth video bitstream coded in an Intra configuration using HTM-0.3 at four texture / depth QPs: 25/34, 30/39, 35/42, 40/45 (please refer to Section 3.3.1 for more details about the coding configuration used), we find that the Intra mode signaling (including DMMs) represents roughly 25% of the total depth bitrate. It is the element that has the largest coding cost in depth videos. Hence, there is much to gain if the bitrate needed to code the depth Intra modes is reduced. The MPM tool, described in Section 2.3.2, reduces this cost. In an MVD system, a depth video is always associated with a texture video, and we can further improve the efficiency of the MPM tool if we exploit the dependency between the Intra modes of texture and depth. In practice, our idea consists in adding a new MPM candidate or predictor, which is the Intra mode of the corresponding texture PU, to the MPM candidate list of a depth PU. However, texture and depth Intra modes do not always match.

Further experiments, under the same configuration and testing conditions, have shown that when comparing a coded depth map to its associated coded texture frame, the Intra mode of a depth PU, chosen by the Rate Distortion Optimization (RDO) process, matches the Intra mode of its co-located texture PU in average 18% of the time, as shown in Table 3.1. These experiments also show that the matching

Sequence	**Percentage of Intra mode matchings (in %)**
Kendo	8.68
Newspaper	12.39
Balloons	6.88
Dancer	26.74
GT Fly	19.29
PoznanHall2	37.63
PoznanStreet	13.78
Average	**17.91**

Table 3.1: Percentage of PUs where the texture Intra mode matches the one in depth for various tested sequences

occurs mostly in areas where there are sharp edges in texture. Indeed, a sharp edge in texture defines a dominant geometric structure which is likely to exist in depth as well. Since Intra modes are highly directional, they closely follow this structure, and hence, the texture and depth Intra modes are likely to be the same, as shown in Figure 3.1. This figure shows a texture PU and its co-located depth PU. The texture PU is composed of parts of a red object and a background. Since the object and the background have different depth levels, they will also be clearly represented in the depth PU as two different regions seperated by a clear edge. A specific Intra direction (represented by arrows), roughly parallel to this edge, will be chosen for the texture PU by RDO, as it is the direction that most accurately describes the geometric structure of the PU. Since this structure also exists in depth, the same Intra direction will be chosen by the depth RDO.

Consequently, if the texture Intra mode inheritance is only done for depth PUs whose co-located texture PU contains sharp edges, the Intra mode signaling bitrate for these PUs will be reduced. The remaining depth PUs, to which the texture Intra mode is irrelevant, will not be impacted.

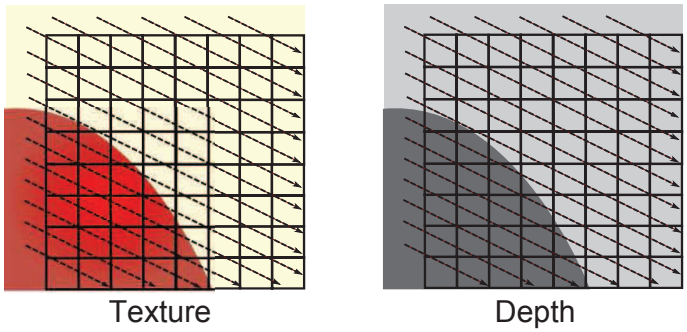


Figure 3.1: Geometric partitionning and resulting Intra direction of a texture and depth PU caused by the presence of an edge

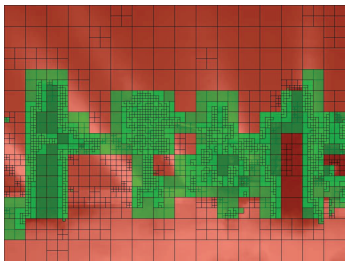
Figure 3.2 shows the depth map of a sequence in our test set: Kendo, with its associated texture frame. This frame is coded using the same Intra configuration and testing conditions as in the previous experiments. In Figures 3.2(c), 3.2(d) and 3.2(e), all PUs are coded in Intra. PUs marked in green in Figure 3.2(c) are PUs that contain sharp edges detected by gradient estimation in texture. PUs marked in yellow in Figure 3.2(d) are PUs where the depth Intra mode matches the texture Intra mode. For the maximal coding efficiency, the texture Intra mode inheritance should only be performed on the yellow PUs, but those are not known *a priori* by the decoder. The green PUs are known, and we can see that the green and yellow PUs approximately superpose, hence validating our assumption.

Note that some Intra mode matchings can be exploited in smooth areas, but it is not very beneficial to do so. Indeed, Figure 3.2(e) shows the coding cost of Intra

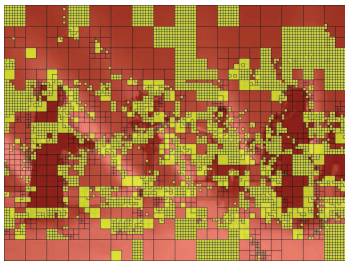
CUs in the Kendo depth map. The R-D cost computed at the encoder for each CU is mapped to an intensity value. Hence, a lighter red color indicates a CU that costs more to code than a darker colored CU. We can see that the smooth areas are cheap to code so we can not expect much gains if we reduce the Intra mode signaling at this level.

(a) Kendo texture frame

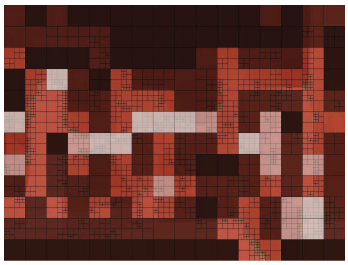
(b) Kendo depth map



(c) In green: PUs containing edges detected in the Kendo texture frame



(d) In yellow: PUs where a texture-depth Intra mode match occurs



(e) Coding cost of depth CUs: lighter colors cost more to code

Figure 3.2: Kendo texture frame and associated depth map analysis

3.2 Proposed method and GradientMax criterion

For the current depth PU, the corresponding texture PU, T_{ref}, needs to be found. It is generally the co-located PU except when the texture CU is more finely partitionned than the depth CU: T_{ref} is set as the top left PU, and when the depth CU is more finely partitionned than the texture CU: the latter is set as T_{ref} for all depth partitions (PUs). These three cases are shown in Figure 3.3.

Then, a metric is computed on T_{ref} and compared to a fixed constant threshold. If the computed metric is higher than that threshold, the texture Intra mode is inherited and added to the MPM candidate list of the depth PU. Otherwise, no inheritance is performed. In Section 3.1, we have shown that the texture and depth Intra modes match in PUs where we have sharp edges in texture. Hence the metric has to quantify the presence of an edge in T_{ref}.

To compute the metric, we perform a Sobel filtering on T_{ref}. We choose the Sobel filter because it is a simple and effective edge detector. Other edge detection filters

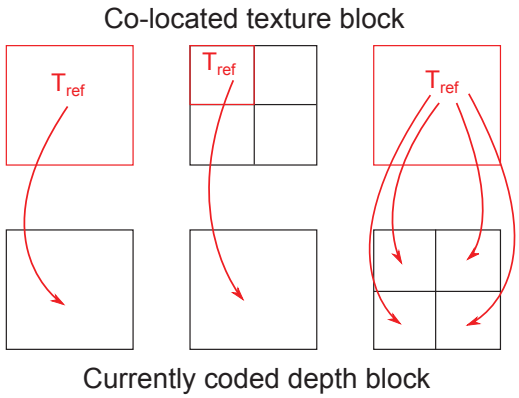


Figure 3.3: Definition of the corresponding texture PU, T_{ref}, for a currently coded depth PU in our algorithm

such as the Prewitt or Canny filters can also be used. The Sobel filter output is composed of two gradient matrices G_x and G_y. From these matrices, we compute the gradient module matrix M as follows:

$$M = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

The metric is set as the maximum value in the PU of this gradient module matrix (GradientMax). It is a measure of the sharpness of the edges present in the texture PU, and can therefore be used to drive the inheritance.

If the computed metric is larger than the threshold, the texture Intra mode is inherited and inserted into the MPM candidate list. However, that list contains at most two candidates. If there is only one spatial candidate in the list (the modes of the two spatial neighbours are identical, or one of the two neighbours is not Intra coded or falls outside the slice), the texture Intra mode is added, and the resulting list is sorted in ascending order of Intra indices. If there are two spatial candidates in the list, the texture Intra mode always replaces the second, and the resulting list is sorted in ascending order of Intra indices. The different steps of the algorithm are depicted in Figure 3.4.

Note that the proposed method is intended to be integrated into a full 3D-HEVC codec. The resulting bitsream is decodable and so, the edge detection in step 2 is performed on a reconstructed T_{ref}, because the depth decoder does not have access to original texture samples. This is possible, since we assume that the texture frame is coded before the depth frame, meaning the depth encoder also has access to reconstructed texture samples.

Furthermore, we observe that the proposed method is somewhat similar to the MPI method described in Section 2.5.4 which can be used in the case of Inter coded PUs. However, MPI is not driven by a criterion as in our algorithm (the motion

vector is always inherited), and as we show in the following, imposing conditions on the inheritance is the most relevant part of the proposed method. Note that MPI and our algorithm can be used together, as also shown in the experimental section.

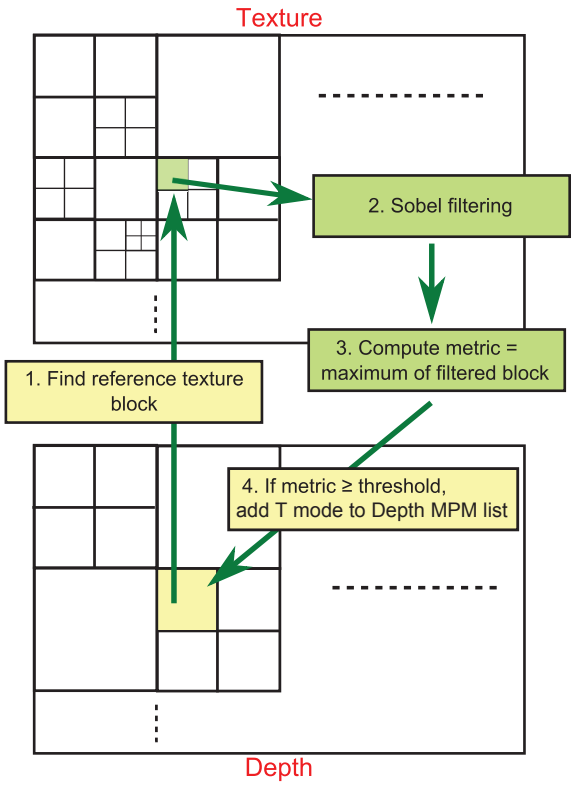


Figure 3.4: Algorithm of our proposed tool

3.3 Experimental results with the GradientMax criterion

3.3.1 Experimental setting

We implement our tool in HTM-0.3 [Teca]. The software is based on the HM-3.0 reference software for HEVC. The coding configuration used is the one defined by the JCT-3V community in the Common Test Conditions (CTC) [RMV13], which are used to evaluate new coding tools for potential adoption in 3D-HEVC. It consists of the following: 8-bit internal processing, CABAC entropy coding, disabling the loop filter for depth coding, DMM, VSO and MPI tools enabled for depth coding, a GOP size of 8 and an Intra period of 24. 35 Intra modes are considered for both texture and depth components. PU sizes range from 64×64 to 4×4. Four QPs for texture: 25, 30, 35 and 40 and their respective QPs for depth: 34, 39, 42, 45 are used to code the sequences, as indicated in the CTCs.

We respect all of these conditions, except that we evaluate our tool in an Intra

configuration (by setting the GOP size and the Intra period to 1), where all the frames of all the views (texture and depth) are coded in Intra. This should not be surprising, since our method is intended to improve Intra coding. Gains on Inter images are still possible because, resulting from the use of our tool, better Intra reference frames are used for Inter prediction, and because PUs can be coded in Intra mode in these frames as well. However, gains in the Inter configuration are smaller since our method has less chances to be selected.

Note that HEVC-specific tools such as the Residual QuadTree (RQT), the Mode-Dependant Directional Transform (MDDT) and the Intra Smoothing (IS) tools are enabled as well, which make up an efficient Intra configuration, on top of which our tool is added.

We test our tool on the seven sequences defined in the CTCs, which consist of four 1920×1088 and three 1024×768 resolution sequences, as shown in Table 3.2. The length of these sequences is 10 seconds each, but we choose to only code half a second of video to speed up the simulations. We believe that this is acceptable since multiple sequences are considered, with different types of content, and because an all-Intra configuration is used anyway, which means the coding of each frame is independant of the coding choices made in the previous frame. To conform to CTCs, a 3-view testing scenario is considered where 3 texture views with their 3 associated depth views are encoded for each sequence. After encoding, 3 views are synthesized between the center and the left view, and another 3 between the center and the right view, making a total of 6 synthesized views per sequence. The Bjontegaard delta rate (BD-Rate) metric [Bjo01] is used to evaluate gains on the depth and synthesized view components. The reference consists of HTM-0.3 with the same coding configuration.

Class	**Sequence**	Frames per second	Total number of frames	Number of coded frames
class A (1920 × 1088)	Dancer	25	250	12
	GT Fly	25	250	12
	PoznanHall2	25	200	10
	PoznanStreet	25	250	12
class C (1024 × 768)	Balloons	30	300	15
	Kendo	30	300	15
	Newspaper	30	300	15

Table 3.2: Sequences in test set

To obtain the average gain for depth, the BD-Rate for each of the 3 coded depth views is computed using the rate and PSNR values associated with the coding of that depth view. Then the three gains are averaged. To obtain the average gain for synthesized views, the BD-Rate for each of the 6 synthesized views is computed

using the PSNR value of the synthesized view and the combined rate of all 3 coded depth views, since all of them are involved in the synthesis of each intermediate view. To compute the PSNRs of the synthesized views, uncompressed texture and depth views are used to synthesize 6 intermediate uncompressed views based on which the PSNR is computed, since original intermediate views to compare to are not available.

In Section 3.2, we introduce a threshold to decide whether to inherit the texture Intra mode for the currently coded depth PU or not. This threshold is empirically fixed and is known by both encoder and decoder, hence, it does not need to be transmitted. We perform the optimization of the threshold on only the first frame of the seven sequences considered in Table 3.2. The optimization consists in testing a large set of thresholds ranging from 0 to 4000 for all sequences, hence it is too computationally intensive to perform on all the frames especially since the encoding runtime of HTM-0.3 is high. Once the threshold that gives the largest gains on average for the first frame is found, we use it as is for the other frames. In this work, we thus present coding results on the first frame and on the entire set of frames, knowing that in the latter case, the optimization is done only on the first frame. The optimal threshold found after exhaustive searches equals 50.

3.3.2 Coding results

Table 3.3 shows the coding gains on the first frame of each sequence in the test set, evaluated on synthesized views and on coded depth views (negative values are gains, positive ones are losses). Average bitrate reductions of 1.3% are reported on synthesized views, and minor losses (0.2%) on depth videos. Gains on the entire set of frames are shown in Table 3.4. In this case, average bitrate reductions of 0.9% and 0.7% on synthesized views and depth videos respectively are reported.

Sequence	**Gains on synthesized views (in %)**	**Gains on depth (in %)**
GT Fly	−3.0	+6.1
Newspaper	−2.0	−0.7
PoznanHall2	−1.4	−2.7
Kendo	−1.0	+0.3
Balloons	−0.7	−0.1
Dancer	−0.6	−1.4
PoznanStreet	−0.2	−0.4
Average	**−1.3**	**+0.2**

Table 3.3: BD-Rate coding results for the first frame in synthesized views and depth with GradientMax

Sequence	Gains on synthesized views (in %)	Gains on depth (in %)
GT Fly	−0.6	0.0
Newspaper	−0.9	−0.8
PoznanHall2	−1.3	−0.5
Kendo	−1.2	−1.2
Balloons	−1.1	−0.4
Dancer	−0.5	−0.6
PoznanStreet	−0.7	−1.3
Average	**−0.9**	**−0.7**

Table 3.4: BD-Rate coding results for the entire set of frames in synthesized views and depth with GradientMax

3.3.3 Results analysis and conclusion

Our tool with the GradientMax criterion gives overall −1.3% gain on synthesized views, and minor losses (0.2%) on depth videos. The loss on depth videos can be explained by the fact that the RDO process in HTM-0.3 optimizes depth map coding for the quality of the synthesized views (due to VSO). This means that a coding mode that is optimal for a currently coded depth PU may not be selected for this PU if it is not optimal for synthesis. Since our tool introduces a new predictor for depth Intra modes, the R-D choices are altered and this may lead in some cases, as in the GT Fly sequence, to a selection of Intra modes that improves significantly the quality of the synthesis (−3.0% gain) at the expense of an even bigger loss on depth (6.1%). Consequently, the gains on depth are not very relevant here.

The gains on synthesized views for the entire set of frames are lower than if they are evaluated only the first frame. This is due to the fact that the threshold is not optimized per frame, but only on the first frame of the sequences. Our optimization process to obtain the threshold can however be done online, using a multi-pass encoder which codes the frame N times until it finds the threshold giving the largest gains, and then finally codes the frame using the threshold obtained. This also means that the encoder should transmit the threshold to the decoder for each frame. Nevertheless, the results given would still hold because the extra signaling is minimal. If this has to be done for each frame, the whole encoding process would turn out to be very complex. Optimizing on the first frame, as we have presented it, can therefore be seen as a compromise between coding efficiency and complexity.

The inheritance and selection percentages, and the inheritance efficiency of our method for both test cases and for each sequence are given in Table 3.5. The inheritance percentage is the ratio between the number of PUs where a texture Intra mode inheritance occurs and the total number of coded PUs. The selection percentage is the ratio between the number of PUs where a texture Intra mode

inheritance occurs and where this inherited mode turns out to be the best (R-D wise) mode for the PU (so in other words, the number of PUs where we "correctly" inherit) and the total number of coded PUs. The inheritance efficiency is the ratio between the selection and the inheritance percentages. We can see that the depth and synthesized view gains are correlated with the selection percentages shown in Table 3.5. Of course, this is not an exact measure, since inheriting the texture Intra mode and selecting it as the best R-D mode for a PU does not necessarily imply gains (the mode in question might already be in the MPM list as a spatial candidate). Likewise, inheriting the texture Intra mode and not selecting it as best R-D mode does not necessarily imply losses (the texture Intra mode would have to replace a predictor which would have been selected as best R-D mode, or get inserted next to it hence increasing its signaling bitrate in case it was alone in the list). Nevertheless, the selection percentage still gives us an idea on the performance of the tool in the various tested sequences. In general, as it increases, the gains (considered on both the depth and synthesized views) increase also.

Sequence	**First frame**			**Entire set of frames**		
	Inh. Perc.	**Selection Perc.**	**Inh. Efficiency**	**Inh. Perc.**	**Selection Perc.**	**Inh. Efficiency**
GT Fly	60.80	15.52	25.52	60.76	15.32	25.22
Newspaper	70.78	16.56	23.39	71.08	15.76	22.18
PoznanHall2	45.95	20.35	44.28	45.89	20.18	43.97
Kendo	49.23	9.86	20.03	49.92	10.90	21.83
Balloons	65.65	10.43	15.89	65.46	10.89	16.64
Dancer	70.73	17.57	24.84	70.88	17.50	24.69
PoznanStreet	79.16	19.88	25.11	78.51	19.22	24.48

Table 3.5: Inheritance and selection percentages, and inheritance efficiency of our method with GradientMax

We believe that the coding gains could be higher, if a better texture mode inheritance criterion is considered, even at the cost of a slightly higher computational cost. The work devoted to finding and testing this criterion is presented in the next section.

3.4 The DominantAngle criterion

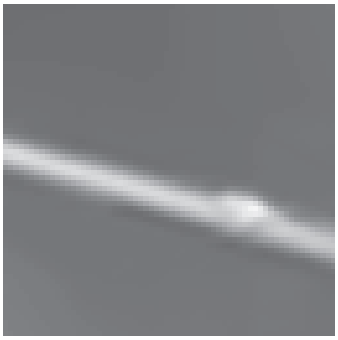
3.4.1 Preliminary Study

An analysis of the texture-depth Intra modes matchings shows that they mostly occur in PUs where there is only one clear sharp directional edge in texture rather than in PUs that have several edges in texture, or one edge which does not have a dominant direction.

Figure 3.5 shows two texture CUs, one having a sharp edge which does not have a single dominant direction, and another having a single sharp directional edge. The first CU (Figure 3.5(a)) is likely to be partitionned in the texture encoding pass because it is difficult to find an Intra mode which gives an accurate prediction of the texture signal. Hence, there will not only be one Intra mode for this CU, but rather one for each partition. Inheriting one of these Intra modes (in practice, it is the one of the top left PU which will be inherited, as shown in Figure 3.3) for the co-located depth PU would not be efficient because the inherited Intra mode is not pertinent in depth. In these situations, it is better not to inherit at all. In the second PU however (Figure 3.5(b)), a specific Intra mode will be able to describe perfectly the dynamics of the PU (hence giving a good prediction signal) and will thus be retained for coding the texture PU. This mode would also be able to describe accurately the dynamics of the co-located depth PU and hence, in this case, it is efficient to inherit it.

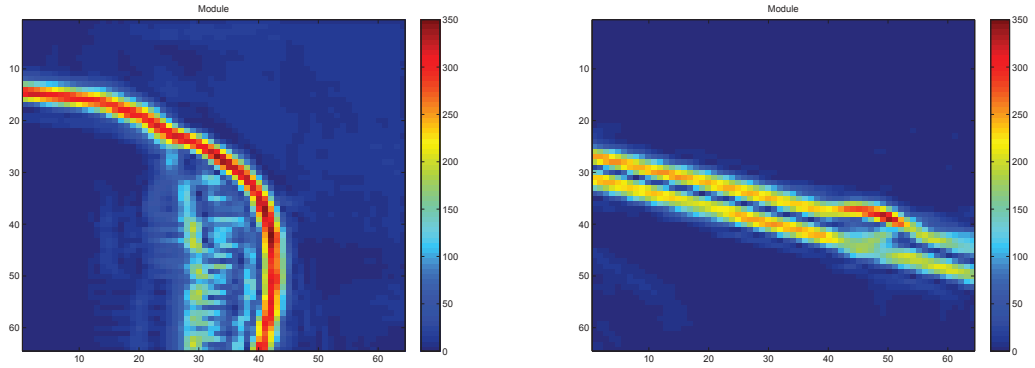
Figure 3.6 shows the module of the gradient computed as in Equation 3.1 for the two PUs of Figure 3.5. The red pixels in the module matrices correspond to sharp edges. The maximum value of the module matrix in the first case equals 346 and in the second case, it equals 337. These are the values of the GradientMax criterion if used. Both are relatively high values. Indeed, if the threshold was set to 50 as in the previous case, the texture Intra mode would be inherited in both cases since both metrics are larger than 50. This inheritance is fine for the second case, but not for the first case. Consequently, there is a need to develop a new criterion which accounts for the direction of the edge and not its sharpness alone.

(a) Texture PU containing a sharp non directional edge



(b) Texture PU containing only one sharp directional edge

Figure 3.5: Two types of texture PUs containing sharp edges



(a) Module matrix obtained after Sobel filtering of the texture PU containing a sharp non directional edge

(b) Module matrix obtained after Sobel filtering of the texture PU containing only one sharp directional edge

Figure 3.6: Module matrices obtained after Sobel filtering of two types of texture PUs containing sharp edges

3.4.2 Proposed criterion

We have developped a new criterion, DominantAngle, that takes into account the direction of the edges present in the texture PUs.

To compute the metric associated with the criterion, we first perform a gradient calculation on the texture PU. Besides the gradient module, we also compute its angle A as follows:

$$A = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.2}$$

The module matrix shows the magnitude of the edges in the texture PU. The angle matrix gives the direction of these edges. For the computation of the metric, we establish the histogram of these angles, but only the angles corresponding to edges with a relatively high module value (50 in our method, as in GradientMax) are considered. The histogram will thus list the number of occurences of the angles corresponding only to sharp edges. To establish the histogram, a number of bins has to be set. We choose $\beta = 33$ bins which correspond to the number of directional Intra modes in HEVC.

The aim is to find PUs containing a single sharp directional edge. Thus, we have to detect a single peak (local maximum) in the histogram, corresponding to that edge. Having other distant peaks in the histogram discredits the initial peak and decreases the pertinence of the texture mode to be inherited. Consequently, we propose to compute the metric as follows:

1. First, initialize the metric c to the maximum histogram value (highest peak). Let x_c denote the bin index of that value in the histogram.

2. If $c = 0$, stop the algorithm.

Else, find the next highest value in the histogram, denoted as p, with a bin index of x_p

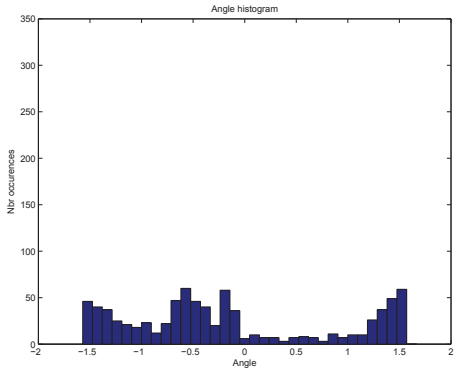
3. If $p \geq \alpha \cdot c$, reduce c such that:

$$c \leftarrow c \cdot \left(1 - \frac{|x_c - x_p|}{\beta}\right) \tag{3.3}$$

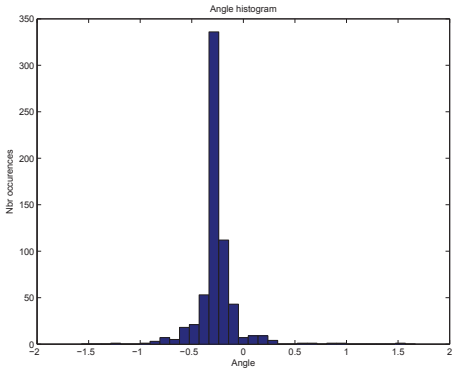
and return to step 2.
Else, stop the algorithm.

This criterion penalizes the maximum histogram value if there are other peaks in the histogram and this penalty is proportional to the distance separating the two peaks, which corresponds to the angle difference. The binning operation, which can be seen as a type of quantization, may lead to the insertion of two close angles into seperate bins. Thus, we can find in some cases two high histogram values that are next to each other. These are not two different peaks. The criterion initialized to the highest value should not be penalized by the presence of the other, because it is practically the same angle, the difference being only due to the binning operation. Eq. 3.3 accounts for this situation. Furthermore, the α parameter has been optimized and empirically set to 0.75 for maximum coding gains.

Figure 3.7 shows the angle histograms of the two PUs in Figure 3.5. For the PU which contains a sharp non-directional edge, the histogram presents many peaks while the histogram of the PU containing only one sharp directional edge contains only one peak that corresponds to that direction. The above-mentionned metric computation gives the values of 4 and 344 respectively. If the threshold was set to 50, this means we will inherit only in the second case and not in the first case, and that is exactly what is required.



(a) Angles histogram of the texture PU containing a sharp non directional edge



(b) Angles histogram of the texture PU containing only one sharp directional edge

Figure 3.7: Angles histograms of two types of texture PUs containing sharp edges

As opposed to the GradientMax criterion case, the threshold used for the DominantAngle criterion is PU size dependent since the maximum number of occurences of an angle in the histogram of a $N \times N$ PU equals N^2. Each PU size has a different criterion dynamic. If the computed metric in a PU of a given size is larger than the threshold corresponding to that PU size, the texture Intra mode is inherited and is inserted into the MPM candidate list. Experiments show that normalizing the computed metric by N^2 to obtain a single threshold for all PU sizes is not the most efficient solution, as bigger PU sizes have a higher weight than smaller ones, and thus require a higher threshold.

3.5 Experimental results with the DominantAngle criterion

3.5.1 Experimental setting

We implement DominantAngle in the same experimental framework and testing conditions as the GradientMax criterion (see Section 3.3.1) to allow for a fair comparison between the two.

The threshold for each PU size is empirically determined. These thresholds are fixed and are known by both the encoder and decoder so they do not need to be transmitted. In HTM-0.3, PU sizes vary from 64×64 to 4×4. Hence there are 5 different PU sizes, and 5 different thresholds to determine. Here also, the optimization is done only on the first frame and the best thresholds obtained are used to code the rest of the frames. Indeed, the optimization process here is even more complex than in GradientMax since it has to be done independently for each PU size. Thus, in this section, we also present coding gains on the first frame, and on the entire set of frames.

The optimization process consists of an exhaustive search for the best threshold for each PU size, starting with 64×64. All the other thresholds are set to 0. Once the threshold that maximizes the average view synthesis gain for all sequences is found, another exhaustive search is performed for 32×32 PUs. The threshold for the 64×64 PU size is set to the previously found threshold and all the others are set to 0. This process is repeated until all five thresholds are found. The optimal thresholds found after exhaustive searches are 15 for the 64×64 PUs and 0 for smaller PU sizes.

3.5.2 Coding results

Table 3.6 shows the coding gains, evaluated on synthesized views ("SV") and on coded depth views, for the first frame of each sequence in the test set, on which the threshold optimization is performed. Average bitrate reductions of 1.6% and

2.3% are reported for synthesized views and depth videos respectively. Gains for the entire set of frames are given in Table 3.7. In this scenario, average bitrate reductions equal 1.0% and 0.7% respectively. These two tables also recall the gains of GradientMax for comparison.

Sequence	**DominantAngle**		**GradientMax**	
	Gains on SV (in %)	**Gains on depth (in %)**	**Gains on SV (in %)**	**Gains on depth (in %)**
GT Fly	−1.4	−1.4	−3.0	+6.1
Newspaper	−1.4	−2.9	−2.0	−0.7
PoznanHall2	−3.1	−7.8	−1.4	−2.7
Kendo	−0.6	+1.1	−1.0	+0.3
Balloons	−0.9	−1.9	−0.7	−0.1
Dancer	−2.4	−2.4	−0.6	−1.4
PoznanStreet	−1.2	−1.2	−0.2	−0.4
Average	**−1.6**	**−2.3**	**−1.3**	**+0.2**

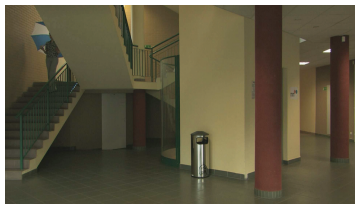
Table 3.6: BD-Rate coding results for the first frame on synthesized views and depth with DominantAngle and GradientMax (for comparison)

Sequence	**DominantAngle**		**GradientMax**	
	Gains on SV (in %)	**Gains on depth (in %)**	**Gains on SV (in %)**	**Gains on depth (in %)**
GT Fly	−0.5	+1.6	−0.6	0.0
Newspaper	−0.9	−1.3	−0.9	−0.8
PoznanHall2	−2.1	−0.6	−1.3	−0.5
Kendo	−0.9	−0.8	−1.2	−1.2
Balloons	−0.6	−0.6	−1.1	−0.4
Dancer	−1.2	−1.2	−0.5	−0.6
PoznanStreet	−0.7	−2.0	−0.7	−1.3
Average	**−1.0**	**−0.7**	**−0.9**	**−0.7**

Table 3.7: BD-Rate coding results for the entire set of frames on synthesized views and depth with DominantAngle and GradientMax (for comparison)

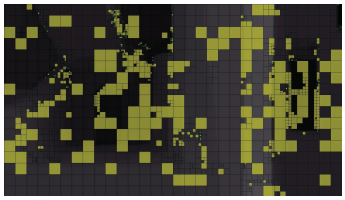
3.5.3 Results interpretation

The optimal thresholds obtained imply that in PUs smaller than 64×64, we always inherit the texture Intra mode. In these PUs, the gains obtained by exploiting all possible matchings overcome the loss induced by the occasional replacement of a good spatial predictor with the inherited texture Intra mode. For 64×64 PUs, this is not the case. A threshold must be established to avoid inheriting in patterned PUs in texture or at edges intersections, since those PUs contain many directional contours. Furthermore, the weight of a 64×64 PU is significant. A bad prediction

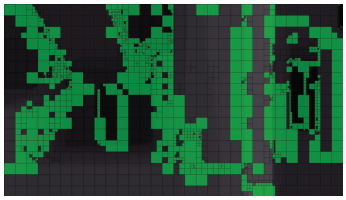


(a) PoznanHall2 central view texture frame

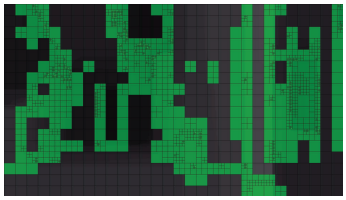
(b) PoznanHall2 central view depth map



(c) Texture-depth Intra mode matchings in depth map



(d) Inheritance with GradientMax



(e) Inheritance with DominantAngle

Figure 3.8: Texture-depth Intra mode matchings PUs and inheritance PUs with the GradientMax and DominantAngle criteria, in the first frame of the central view of PoznanHall2

in this PU affects the prediction and coding of many subsequent PUs. Hence, for this PU size, the optimal threshold is 15.

Furthermore, the inheritance and selection percentages, and the inheritance efficiency of our method for both test cases and for each sequence are given in Table 3.8. The gains shown in Table 3.6 and Table 3.7 are correlated with these selection percentages. In most cases, when the selection percentage increases, the gains increase as well. Compared to GradientMax (see Table 3.5), DominantAngle yields more frequent selections, but more frequent inheritances as well. This is due to the fact that the threshold for PU sizes different than 64×64 is 0, meaning the texture Intra mode is always inherited in those PUs.

Sequence	**First frame**			**Entire set of frames**		
	Inh. Perc.	**Selection Perc.**	**Inh. Efficiency**	**Inh. Perc.**	**Selection Perc.**	**Inh. Efficiency**
GT Fly	75.20	20.48	27.24	75.20	19.26	25.61
Newspaper	96.83	22.46	23.20	96.87	22.15	22.87
PoznanHall2	68.89	30.07	43.64	69.64	32.13	46.14
Kendo	88.32	18.23	20.64	88.04	16.71	18.98
Balloons	92.91	14.43	15.54	93.01	14.83	15.95
Dancer	89.01	26.90	30.22	89.74	27.15	30.26
PoznanStreet	91.69	21.15	23.07	91.68	21.86	23.84

Table 3.8: Inheritance and selection percentages and inheritance efficiency of our method with DominantAngle

Figure 3.8(c) shows, in yellow, the PUs in which a texture-depth Intra mode

matching can be exploited in the first frame of the central view of the PoznanHall2 sequence. Figure 3.8(d) shows, in green, the PUs where we inherit the texture Intra mode using GradientMax, and Figure 3.8(e) shows the same with DominantAngle. We can see that there are indeed more green PUs in DominantAngle than in GradientMax. DominantAngle covers more yellow PUs than GradientMax (hence the observed increase in the selection percentages), but also more PUs where there is not a matching to exploit, and those are mostly smaller size PUs due to the threshold set to 0 for these PU sizes.

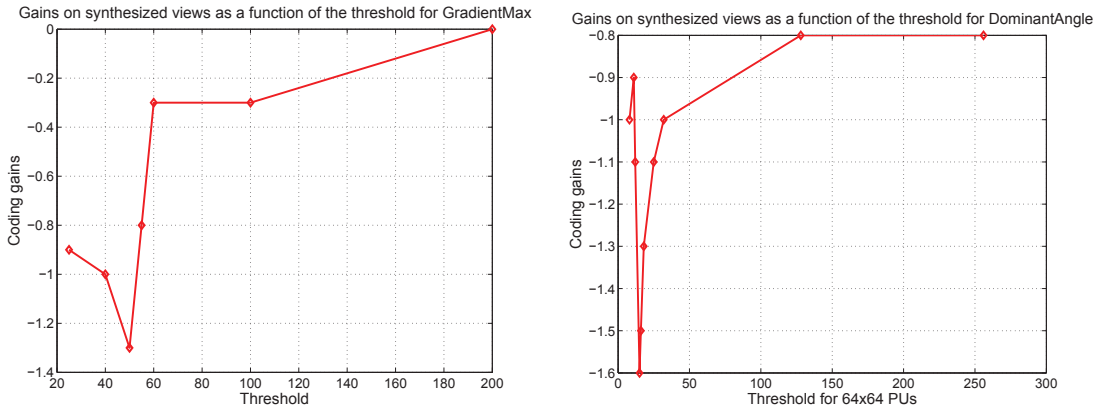
However, that increase in the inheritance percentage in DominantAngle, and consequently, in the percentage of PUs where the texture Intra mode was inherited but not selected (which can be seen as the difference between the inheritance and the selection percentage), is not problematic. As previously said, the fact that the texture Intra mode was inherited for a PU and not selected does not always imply losses. But even if it does, the losses would be minimal because they occur in smaller PU sizes. Also, the smaller the PU size, the more planar the depth PU and the corresponding texture PU would be, and thus, in these cases, the best mode is most probably either the non-directional Planar or DC mode. Inheriting one instead of the other is certainly not ideal, but is not catastrophic either as both succeed in representing the dynamics of a planar PU.

There are however some PUs where an inheritance can happen in GradientMax, and not in DominantAngle, and those are PUs where there is more than one clear directional edge in texture. Avoiding to inherit in these PUs will reduce losses in DominantAngle compared to GradientMax. Figure 3.8(c) shows for instance that there is no Intra mode matching in the first 64×64 top-left PU in the depth map of the central view of PoznanHall2. This is expected, because the corresponding texture PU is actually patterned, as shown in Figure 3.9, so it has no pertinent Intra mode to offer for its corresponding depth PU. In GradientMax, an inheritance is made in this case, as shown in Figure 3.8(d), but that is successfully avoided in DominantAngle, as shown in Figure 3.8(e).

Figure 3.9: First top-left 64×64 PU (with adjusted contrast for visibility) of the first texture frame in the central view of PoznanHall2

When comparing the stability of the optimal threshold obtained for each criterion (for DominantAngle it is the 64×64 threshold) as shown in Figure 3.10, we find that in the case of DominantAngle, when the threshold approaches the optimal value (15), the coding gain varies very quickly. However, we also observe that, even if we make a wrong guess about the best value (for example we double the value and use 30), the gain are still high (-1%). Even if we use an overestimated value (e.g. 120, 8 times the best value), we still keep half of the gains. In conclusion, it is important to use the best threshold value, but globally this technique is robust with respect to the selection of this parameter.

As far as the GradientMax method is concerned, we observe a similar behavior. The best value of the threshold is 50, which gives 1.3% rate reduction. If this value is halved, we still have 0.9% rate reduction. On the other hand, when the threshold is doubled we only gain -0.3%, and when it is quadrupled, all the gains are lost. We conclude that the GradientMax method, not only has a smaller best gain, but is also a bit less robust with respect to the threshold selection.



(a) GradientMax (b) DominantAngle

Figure 3.10: Average gains on synthesized views as a function of the threshold

When further analyzing the gains we obtain on synthesized views for each proposed criterion, considering the more realistic scenario where we code all the frames, we can see (from Table 3.7) that the results are somewhat coherent across sequences. The standard deviation equals 0.50 for DominantAngle and 0.29 for GradientMax which are relatively small values. For DominantAngle, the gains vary between 50% and 210% of the average gain value while in GradientMax the gains vary between 56% to 144%. We believe that these ranges are acceptable. Both methods are statiscally stable in that sense, and even though GradientMax outperforms DominantAngle in some sequences, on average, DominantAngle remains better. Consequently, we have succeeded in finding a better criterion than GradientMax.

3.6 Conclusion

In this chapter, we have presented a novel depth video coding tool that exploits the statistical dependency between the texture and depth Intra modes in order to increase the coding efficiency and achieve gains on synthesized views. The proposed method first finds the corresponding texture PU for a currently coded depth PU. Then it computes a metric on that texture PU and if it is larger than a specified threshold, adds the texture Intra mode to the MPM candidate list where it may replace another spatial candidate.

Two criteria were studied in this work: GradientMax and DominantAngle. The rationale behind the GradientMax criterion is that the texture-depth Intra mode matchings occur only in areas where there are sharp edges in texture. Based on that assumption, the metric was set as the sharpness of the edges present in the texture PU. Our tool associated with the GradientMax criterion gave -1.3% gain on average for synthesized sequences and a small loss on depth, when the corresponding threshold was optimized. A further study showed that our initial assumption was not completely accurate. Texture-depth Intra mode matchings actually occured in areas where there is one sharp edge in texture. This meant that the initial inheritance set was actually a superset of the appropriate inheritance set. Based on this remark, we developped a new criterion, DominantAngle, which accounted for the direction of the edge in a texture PU. This new criterion gave -2.3% gain on depth sequences and -1.6% gain on synthesized sequences with optimized thresholds. A journal article detailing our method and the two developped criterions was accepted for publication in the APSIPA Transactions on Signal and Information Processing in October 2013 [MJCPP13a].

In the future, we will implement a more intelligent content-adaptive and systematic way to drive the inheritance without relying on complex threshold optimizations. Furthermore, the direct inheritance of the texture Intra mode can also be considered (wherein the currently coded depth PU is forced to be coded with the inherited mode, without having to signal this mode in the bitstream since the same process can be repeated at the decoder) if the statistical dependency is expected to be exceptionally high. The resulting progressive inheritance scheme would therefore be able to adapt itself to the degree of dependency between texture and depth Intra modes in order to increase coding efficiency.

Chapter 4

Improvement of the inter-view motion prediction in 3D-HEVC

Contents

As presented in Chapter 2, inter-view motion prediction (IVMP) is a tool in 3D-HEVC that increases the coding efficiency of dependent views by introducing new MV predictors from the base view. Specifically, IVMP adds a multiview MV candidate in the Merge and AMVP candidate lists of a dependent view PU. This

MV is the one of the base view PU which corresponds to the current PU in the dependent view. The correspondence is made using a disparity vector (DV_{IVMP}) derived in the NBDV process.

In this chapter, we propose to improve IVMP in two ways: first, an additional interview candidate, which is always a DV, is added in the Merge candidate list to achieve a better equilibrium between MVs and DVs in this list, which will in turn increase disparity-compensated prediction (DCP) selection and achieve coding gains. Second, the NBDV process is modified to avoid selecting as DV_{IVMP} the first neighboring DV found, which is not necessarily the best from a coding efficiency point of view.

The first section presents method 1, the results of which are detailed in a second section. The second method is detailed in section 3 along with its different variants, and the corresponding coding results are detailed in a fourth section. Results are analyzed and interpreted in both cases. Section 5 concludes this chapter and underlines possibilities for future work.

4.1 Additional Inter-view Merge Candidate

In 3D-HEVC, inter-view redundancies are exploited using DCP. DCP enables having, for the current frame, reference frames from different views at the same time instant. DVs are estimated in the same way as MVs. However, when analyzing videos coded using HTM-4.1, we can see that DCP-coded PUs are not numerous. This is, in part, due to the fact that there are not enough DV candidates in the Merge candidate list. Indeed, even though IVMP introduces a multiview candidate in the list, this candidate is preferred to be a MV rather than a DV. In this first section, we modify the Merge candidate list to achieve a better equilibrium between MV and DV candidates, with the aim of increasing DCP selection and coding gains.

4.1.1 State-of-the-art

Several tools that modify the Merge candidate list construction in 3D-HEVC were proposed to either achieve coding gains, or reduce complexity / memory consumption. In [GGG12a], the primary candidate list is checked and the first DV candidate found is used to compute, by adding a positive and a negative offset, two more DV candidates which will then be added to the list. However this requires having a DV in the primary list to begin with, which is not a frequent case. Consequently, the coding gains are limited. In [GGG12b], a dynamic re-ordering of the Merge index is performed to reduce its coding cost, based on a conversion table and a Merge index histogram regularly updated. The coding gains however are not high enough to

justify the usage and the frequent updates of the histogram and conversion tables, which increase complexity. The Merge pruning process can also be changed, like in [LCHL12], where a comparison between the multiview candidate and the first two spatial candidates is added. This method achieves 0.3% bitrate reduction on dependent views with no runtime increase and was adopted in 3D-HEVC.

Tools that affect the Merge candidate list construction were also proposed in HEVC. In [LCT+11], the temporal candidate (TMVP) position is changed from the center of the co-located PU to the bottom-right position. Significant bitrate reductions of 0.9% were reported and thus the method was adopted in HEVC. In [YI12], two refined candidates are computed from the first Merge candidate and added to the secondary list of candidates, to replace the combined ones for uni-predicted PUs. Coding gains were not significant enough however to favor adoption.

These methods try to improve the candidate list construction but with no particular intention to balance the DCP selection against the MCP selection in the process. We propose in this section a novel method to reach a better DCP / MCP equilibrium by inserting a DV candidate in the Merge list.

4.1.2 Motivation

Figure 4.1 shows parts of a B-frame of the Kendo sequence coded with HTM-4.1. The PUs coded using MCP are shown in grey (Merge-SKIP) and green (Inter). PUs coded using DCP are shown in light pink (Merge-SKIP) and dark pink (Inter). Blue PUs are coded in Intra. We can clearly see that Merge mode is selected often, and that DCP coded PUs are not numerous.

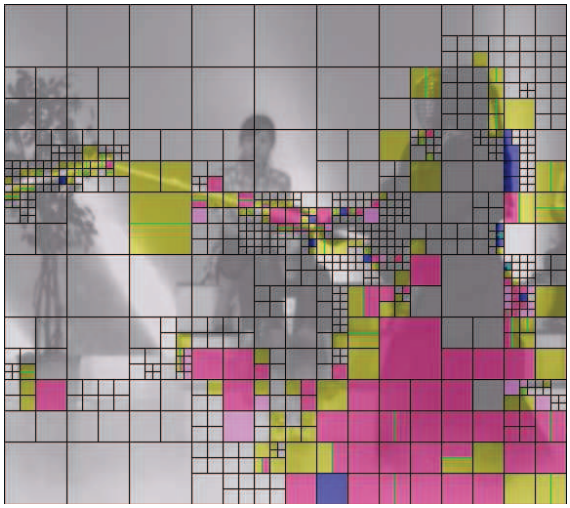


Figure 4.1: CU coding modes in parts of a Kendo B-frame coded with HTM-4.1

Furthermore, Table 4.1 gives the percentages of Merge coded PUs, DCP coded

PUs, and DCP coded PUs in Merge mode, averaged across four QPs, in the dependent texture and depth views of the seven JCT-3V sequences. These results confirm

Sequence	**Merge**	**DCP**	**DCP-Merge**
Kendo	92	17	14
Newspaper	88	15	12
Balloons	93	13	11
Dancer	90	26	21
GT Fly	96	18	15
Poznan Hall2	95	9	7
Poznan Street	94	15	12
Average	**92**	**16**	**13**

Table 4.1: Percentage of Merge coded PUs, DCP coded PUs, and DCP coded PUs in Merge mode in HTM-4.1

the assertion that Merge mode is selected often, actually for 92% of PUs. This is due to the fact that Merge mode is very efficient at reducing the cost of motion / disparity parameters as only an index is encoded. Table 4.1 also shows that only 16% of PUs use DCP, and they are also most often coded in Merge mode (13%).

While it is true that there are often more temporal correlations than inter-view, as shown in [ZJYH09], the main issue behind the unfrequent DCP selection remains the lack of DV candidates in the Merge candidate list. Indeed, DCP can yield a better prediction for a given PU than MCP, in case there is little disparity between views or if there is fast motion in the video. However, not having a DV candidate in the Merge list increases the rate needed to code the PU with DCP since the only option left is to send a motion vector residual. MCP, while maybe not yielding a lower distortion value, requires a lower rate due to the fact that there are numerous MV candidates in the Merge list and signaling the motion parameters only costs an index. Consequently MCP is chosen more often since its Lagrangian cost is smaller, but if a DV candidate was added in the Merge list, as proposed in this section, the required rate for DCP coding would be decreased, hence increasing the selection of DCP and achieving coding gains.

4.1.3 Description of the proposed method

When computing the multiview candidate in the Merge list, a DV (referred to as $\mathrm{DV_{IVMP}}$) pointing to a reference block in the base view is derived using NBDV, as explained in Section 2.4.1. The multiview candidate is set as the MV of that reference block, and only if that MV does not exist, it is set as $\mathrm{DV_{IVMP}}$. We propose to insert $\mathrm{DV_{IVMP}}$ as a new interview candidate in the Merge list along side the multiview candidate if the latter turned out to be a MV. In the rest of this chapter, this

proposed method is referred to as Additional Inter-view Merge Candidate (AIMC).

Two insertion methods are proposed. In method 1 (AIMC-1), the candidate is inserted in the secondary list along with the combined and the zero vector candidates. If any of the first five candidates in the primary list is unavailable, the interview candidate is inserted after the final spatial candidate (before the temporal) to complete the list. If more primary candidates are unavailable, the combined and zero vector candidates are then appended to the list, as it is normally done. In method 2 (AIMC-2), the candidate is inserted in the primary list, in the 5^{th} position, shifting the final spatial candidate to the 6^{th} position. The temporal candidate is hence pushed out of the primary list and into the secondary list. It is the first candidate in the secondary list to be appended back in the primary list if some candidates are unavailable. Figure 4.2 illustrates these two methods. In both methods, before inserting the interview candidate, a redundancy check (*i.e.*a vector equality check in both horizontal and vertical components) with all candidates preceding it in the list is performed for better coding efficiency. Note that the insertion positions in both methods have been empirically set as those positions gave out the most coding gains on average.

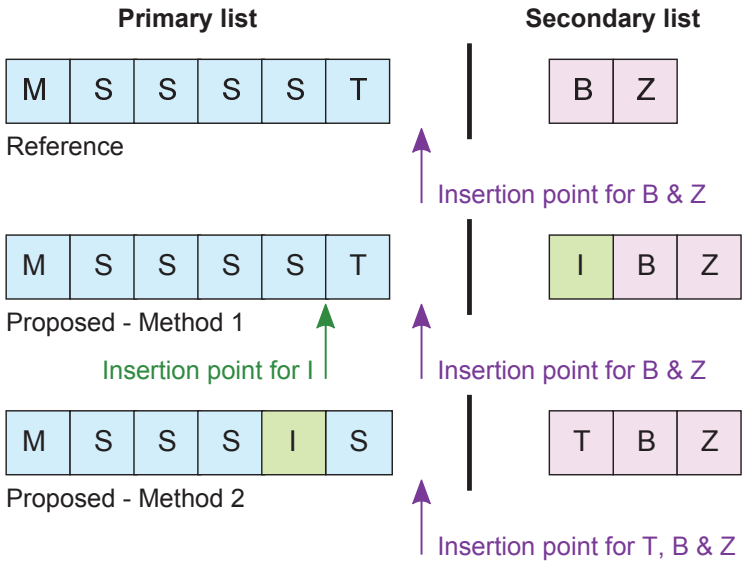


Figure 4.2: Proposed insertion methods (M: multiview, S: spatial, T: temporal, B: combined, Z: zero, I: interview candidate)

4.2 Experimental results of AIMC

4.2.1 Experimental setting

We implement AIMC-1 & AIMC-2 in HTM-4.1 [Tecd]. We strictly follow the coding configuration defined in the CTCs [RMV13] and which is detailed in Section 3.3.1.

Full-frame experiments are done on 10 seconds of video length (cf. Table 3.2). The same 3-view testing scenario as described in Section 3.3.1 is used.

4.2.2 Coding results

Tables 4.2 and 4.3 give the coding gains, measured in BD-Rate (negative values are gains), and runtimes obtained with AIMC-1 and AIMC-2 respectively. These results are summarized in Table 4.4 which also gives the average results if the redundancy check preceding the insertion of the interview candidate in the list is removed. In these tables, the "Video" column shows the gains on the central (0) and on the two side views (1 and 2) and averages these results. The "Synt." column gives results on the 6 synthesized views (the bitrate considered is the sum of the 3 texture and depth bitrates, and the PSNR is the average PSNR of all 6 synthesized views). The "Coded+Synt." result is the same as in the previous column except that the PSNR considered is the average PSNR of the 6 synthesized views and the 3 coded texture views.

Sequence	**Video**				**Synt.**	**Coded**	**Runtimes**	
	0	**1**	**2**	**Avg**		**+Synt**	**Enc**	**Dec**
Balloons	0.0	-0.6	-0.6	-0.3	-0.2	-0.2	96	102
Kendo	0.0	-0.6	-0.4	-0.2	-0.1	-0.2	100	101
Newspaper	0.0	-0.3	-0.3	-0.1	-0.1	-0.1	100	101
GT Fly	0.0	-1.2	-1.0	-0.3	-0.2	-0.3	97	100
Poznan Hall2	0.0	0.2	-0.5	-0.1	-0.1	-0.1	97	94
Poznan Street	0.0	-0.6	-0.6	-0.2	-0.2	-0.2	90	100
Dancer	0.0	-0.6	-0.6	-0.2	-0.2	-0.2	97	102
Average	**0.0**	**-0.5**	**-0.6**	**-0.2**	**-0.2**	**-0.2**	**97**	**100**

Table 4.2: BD-Rate coding results with AIMC-1

Sequence	**Video**				**Synt.**	**Coded**	**Runtimes**	
	0	**1**	**2**	**Avg**		**+Synt**	**Enc**	**Dec**
Balloons	0.0	-0.6	-0.6	-0.3	-0.2	-0.2	97	95
Kendo	0.0	-0.6	-0.4	-0.2	-0.1	-0.1	98	101
Newspaper	0.0	-0.3	-0.2	-0.1	-0.1	-0.1	100	90
GT Fly	0.0	-1.2	-1.2	-0.4	-0.2	-0.3	101	100
Poznan Hall2	0.0	-0.1	-0.3	-0.1	-0.1	-0.1	93	107
Poznan Street	0.0	-0.7	-0.6	-0.2	-0.2	-0.2	92	100
Dancer	0.0	-0.6	-0.6	-0.2	-0.2	-0.2	93	101
Average	**0.0**	**-0.6**	**-0.6**	**-0.2**	**-0.2**	**-0.2**	**96**	**99**

Table 4.3: BD-Rate coding results with AIMC-2

Method	Video				Synt.	Coded +Synt	Runtimes	
	0	1	2	Avg			Enc	Dec
AIMC-1	0.0	-0.5	-0.6	-0.2	-0.2	-0.2	97	100
AIMC-1-NORC	0.0	-0.4	-0.4	-0.1	-0.1	-0.1	98	101
AIMC-2	0.0	-0.6	-0.6	-0.2	-0.2	-0.2	96	99
AIMC-2-NORC	0.0	-0.5	-0.4	-0.2	-0.1	-0.1	98	100

Table 4.4: BD-Rate coding results when the redundancy check is removed (NORC) in AIMC-1 and AIMC-2

Tables 4.2 and 4.3 show bitrate reductions of 0.5% (resp. 0.6%) and 0.6% for side views, 0.2% for synthesized and 0.2% for coded and synthesized views. This is accompanied by a 3% (resp. 4%) encoder runtime reduction. No gains are achieved on the central view since AIMC is not applied there. Table 4.4 shows that coding efficiency is reduced if the redundancy check is removed, with no decrease in encoder and decoder runtimes compared to the original version.

4.2.3 Results interpretations

The gains obtained result from an increase in DCP selection. Inserting a DV into the Merge candidate list reduces the rate needed for DCP coding and favors its selection, especially if there is small disparity between views (interview redundancies are much higher, and DVs can point to a better hypothesis) or if there is fast motion in the video (MVs are not able to correctly predict PUs). Figure 4.3 indeed shows an increase in DCP coded PUs compared to Figure 4.1. This is confirmed in the numerical results of Table 4.5, which shows, for AIMC-1 and AIMC-2, an increase of 8% and 11% on average in the percentage of DCP-coded PUs and DCP-coded PUs using Merge mode.

Sequence	DCP increase		DCP-Merge increase	
	AIMC-1	AIMC-2	AIMC-1	AIMC-2
Kendo	6.2	6.4	9.1	8.9
Newspaper	5.0	4.9	7.8	8.0
Balloons	8.2	7.8	12.2	11.3
Dancer	7.7	7.3	10.8	10.6
GT Fly	17.6	17.0	21.0	20.3
Poznan Hall2	6.3	6.8	8.4	8.9
Poznan Street	6.9	8.2	9.3	10.8
Average	**8.3**	**8.3**	**11.2**	**11.3**

Table 4.5: Percentage increase of DCP-coded PUs and DCP-coded PUs using Merge mode in AIMC-1 and AIMC-2

The complexity resulting from the redundancy check used in AIMC-1 and AIMC-

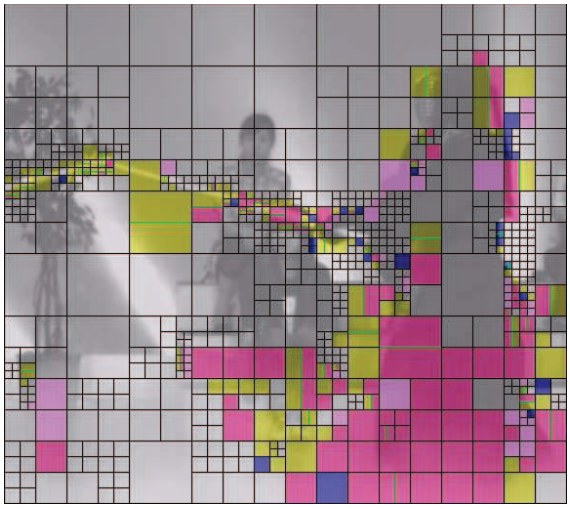


Figure 4.3: CU coding modes in parts of a Kendo B-frame coded with HTM-4.1 and AIMC

Sequence	**Combined**	**Zero**	**Total**
Kendo	6	12	9
Newspaper	6	13	9
Balloons	7	12	9
Dancer	8	10	9
GT Fly	6	10	8
Poznan Hall2	8	11	9
Poznan Street	6	13	9
Average	**7**	**12**	**9**

Table 4.6: Reduction percentage of the number of constructed combined and zero candidates in AIMC-1 or AIMC-2 (same percentages in both)

2 is debatable. The purpose of this redundancy check is to avoid having a redundant DV candidate in the list which will either push potentially better primary candidates further down the list, while increasing their indices, and hence their signaling cost, in the process, or take the place of other, potentially better, secondary candidates which will not even be evaluated. The maximum number of checks equals 4 and 5 in AIMC-1 and AIMC-2 respectively. These would be quite complex to perform for each PU. However, we show in Table 4.4 that removing the redundancy check decreases coding efficiency, as expected, while not reducing neither encoder or decoder runtime. Indeed, the worst case rarely occurs. Consequently, keeping the redundancy check is a better choice.

AIMC-1 and AIMC-2 also bring small encoder runtime reductions of 3 and 4%. This is because inserting a DV candidate in the Merge list means constructing one less secondary candidate, which is a complex process since it involves mixing different vectors to construct combined candidates or looping around all reference indices

to construct zero-vector candidates. Additional experiments have shown that the number of constructed secondary candidates has decreased by 9% on average in AIMC-1 and AIMC-2 as shown in Table 4.6.

AIMC-1 and AIMC-2 were presented at the 2nd JCT-3V meeting in October 2012 [MJPPC12]. AIMC-2 was adopted in the 3D-HEVC draft and software (HTM-5.0). The next section details a second method aimed at improving IVMP based on a HTM-5.0.1 reference which thus includes AIMC-2.

4.3 MedianNBDV

In order to construct the multiview and inter-view candidates in the Merge and AMVP candidate lists, DV_{IVMP} needs to be derived using the NBDV process. However, in NBDV, the first DV / DDV found in a neighboring PU is selected as the final DV with no guarantee of optimality. The method proposed in this section modifies the NBDV process to obtain more accurate DVs which will, in turn, increase coding gains.

4.3.1 State-of-the-art

Several DV derivation techniques have been proposed in 3D-HEVC. The first one involves a depth map estimate which is computed and maintained for each view using already coded texture information. The maximum depth value contained in the collocated PU in the depth map estimate is transformed into the required DV. This derivation process is called Depth Map Disparity Vector (DMDV) [TWCY12]. To obtain the depth map estimates, the coded disparity vector field between the first dependent view and the base view is transformed into a depth map which is then warped to the base view and to other dependent views. Over time, the estimated depth maps are motion-compensated using the same motion vector field as in texture and corrected with coded disparity vector fields. The complex warpings and successive motion compensations that DMDV involves led to its later replacement with NBDV.

Depth-oriented NBDV (DoNBDV) [CWTL12] is an interesting refinement of the classic NBDV process. It uses the coded depth map of the base view to refine the DV obtained after the standard NBDV process. Basicaly, the DV obtained is used to point to the corresponding PU in the base depth view. The maximum depth value inside that PU is converted into another DV which will be used for IVRP and IVMP. DoNBDV achieves significant bitrate reductions compared to NBDV (0.4% on coded views and 0.3% on coded+synthesized views) but adds a non-negligible decoding dependency between the base depth view and the dependent texture view

(indeed, if the base depth view is corrupted, the dependent texture view cannot be decoded).

A final DV derivation process can be conceived if the depth is coded before the texture component. This is possible using the flexible coding order (FCO) tool which allows to change the coding order in 3D-HEVC. In this case, the DV can simply be computed from the coded depth component (taking the maximum depth value in the collocated depth PU and transforming it into a disparity) without the need of transmitting it since the process can be repeated at the decoder.

The DV derivation process in 3D-HEVC is subject to intensive research and is expected to change over the course of the standardization phase. At the time of developping the method proposed in this section (following the 2nd JCT-3V meeting) the derivation process used in 3D-HEVC was NBDV since it was coding efficient, not complex, and did not introduce new decoding dependencies. However, NBDV is sub-optimal. Indeed, the first DV / DDV found in a neighboring PU during the NBDV search process is selected as the final DV and the search process stops. The remaining neighbors are not checked even if some have a DV / DDV which is better, rate-distortion (R-D) wise, than the selected one hence the sub-optimality of the process. The proposed method answers and solves this specific issue.

4.3.2 Motivation

Table 4.7 shows the percentage of PUs coded in Merge mode using either the multiview or the inter-view candidate in HTM-5.0.1, averaged across four QPs, for the seven JCT-3V sequences. The test conditions used are the same as the ones used to evaluate our method, which are described in Section 4.4.1. We can see that the multiview Merge candidate is largely selected in HTM-5.0.1 (by 57% of PUs coded in Merge mode, on average) since it is inserted at the first position in the list (the rate needed to code a merge index of 0 is small, hence the R-D cost of this candidate is small as well). The inter-view candidate is inserted further down the list and is thus selected less often (only 1%).

The efficiency of the multi-view and the inter-view candidate directly depends on the DV derived using NBDV (DV_{IVMP}), which is used to construct both candidates. If the accuracy of DV_{IVMP} is improved through the modification of the NBDV process, as proposed in this section, the distortion associated with these two candidates will decrease, along with their R-D cost, hence increasing their selection and achieving coding gains. These gains will be significant since in general, the Merge coding mode which is already widely selected (92% of PUs as shown in Table 4.1) will be improved, and more specifically, the first candidate in the Merge list which is also largely selected as shown in Table 4.7 will be improved as well. It is important to

Sequence	Multi-view	Inter-view
Kendo	51.6	2.1
Newspaper	53.4	1.4
Balloons	59.9	1.6
Dancer	45.9	1.7
GT Fly	65.6	0.9
Poznan Hall2	61.5	0.9
Poznan Street	60.6	1.2
Average	**56.9**	**1.4**

Table 4.7: Percentage of PUs coded in Merge mode using the multi-view or the inter-view candidates in HTM-5.0.1

note that some of the resulting gains will also come from improving these candidates in the AMVP list but those gains are small compared to the ones resulting from the improvement in the Merge list.

4.3.3 Description of the proposed method

In the proposed method, referred to in the rest of this chapter by MedianNBDV, the search process for a DV in NBDV is never stopped. All the spatial and temporal neighbors are checked, in the usual order, and all found DVs and DDVs are stored together in a single list. A redundancy check is applied to remove redundant vectors in this list. Then, the median of all remaining vectors is computed and is set as the final DV used for IVMP. Applying MedianNBDV for IVRP as well will be tested seperately in a variant (*i.e.* MedianNBDV is used to derive DV_{IVMP} and NBDV is used for DV_{IVRP}). Figure 4.4 illustrates the different steps of our algorithm.

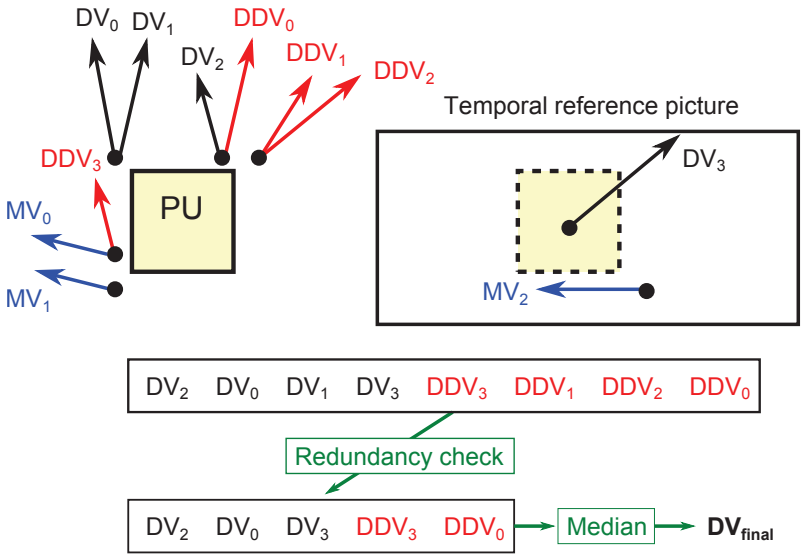


Figure 4.4: Proposed DV derivation method

The advantage of MedianNBDV is that it groups different types of DVs, namely DVs obtained from DCP-coded PUs and DDVs obtained from MCP-coded PUs, in a single list. This heterogeneity in lists is usually coding efficient. For instance, in the Merge candidate list, secondary candidates are constructed to fill the list if primary candidates are unavailable. Hence, two types of candidates can potentially be in the same list, namely primary and secondary candidates. This configuration has been proven to bring coding gains compared to one which does not involve secondary candidates. MedianNBDV is thus set in the same mind frame.

The disadvantage of MedianNBDV lies in the worst case scenario for median computation. Indeed, each spatial neighbor can have up to two vectors, one from each reference list, and there are 5 neighbors. In addition, there are two temporal neighbors in two temporal reference frames, each one having at most one vector. In case all spatial neighbors in both reference lists and all temporal neighbors have DVs or DDVs, and there is no redundancy between these vectors, the median has thus to be computed on 14 vectors. This is quite complex to perform in hardware. Consequently, in order to avoid this worst-case scenario, different variants of the method have been implemented and tested.

4.3.4 Variants

The 1REF variant consists of storing in the list a maximum of one vector per spatial neighbor. In case the spatial neighbor has two DVs or two DDVs, only the one from reference list 0 is stored. In case it has one DV and one DDV, only the DV is stored. In this configuration, the maximum number of spatial candidates is 5, making the worst-case maximum number of vectors on which the median is computed (*MaxCand*) equal to 9. Another variant, RMPOS, consists in simply removing one or more spatial positions (for example A_0) from the check, hence decreasing *MaxCand* by 2 (or by 1 if associated with 1REF) for each spatial position removed. In our experiments, RMPOS consisted in removing the A_0 and the B_2 spatial positions from the check. A final variant aimed at reducing *MaxCand*, called LIMIT-X, consists in storing only the first X found DVs / DDVs in the list. In this case, $MaxCand = X$. Note that the LIMIT-1 variant is equivalent to the standard NBDV process.

The following variants are not aimed at reducing *MaxCand*, but rather implemented and tested to make interesting interpretations: NODDV does not store any DDVs in the list, ALLOWRED removes the redundancy check before median computation, NOAMVP does not apply our method for AMVP while APPLYRES applies it for IVRP as well, and finally, MEAN replaces the median computation with the computation of the vectors' average.

4.4 Experimental results of MedianNBDV and its variants

4.4.1 Experimental setting

We implement MedianNBDV and its variants in HTM-5.0.1 [Tece]. We strictly follow the CTCs [RMV13]. Full-frame experiments on 10 seconds of video are performed, and a 3-view test scenario is considered.

4.4.2 Coding results

Objective results

Table 4.8 gives the coding gains, measured in BD-Rate (negative values are gains), achieved with MedianNBDV. These results are summarized also in Table 4.9 along side all the studied variants (only the average results accross all sequences are shown). In this table, the column "MaxCand" is added to show the maximum number of vectors on which the median can be computed in a worst-case scenario, per variant. Note that there is a ±3% error margin on the encoder and decoder runtimes, because even if launched back to back on the same machine, the runtime of an encoding or decoding process varies slightly every time.

Sequence	**Video**				**Synt.**	**Coded**	**Runtimes**	
	0	**1**	**2**	**Avg**		**+Synt**	**Enc**	**Dec**
Balloons	0.0	-0.7	-0.7	-0.3	-0.2	-0.2	100	99
Kendo	0.0	-1.2	-1.3	-0.5	-0.4	-0.4	98	97
Newspaper	0.0	-0.7	-0.7	-0.3	-0.2	-0.2	99	98
GT Fly	0.0	-0.6	-0.9	-0.2	-0.2	-0.2	89	98
Poznan Hall2	0.0	0.0	-0.5	-0.1	-0.2	-0.2	102	101
Poznan Street	0.0	-0.4	-0.4	-0.1	-0.1	-0.1	104	95
Dancer	0.0	-0.9	-1.0	-0.3	-0.4	-0.4	108	99
Average	**0.0**	**-0.6**	**-0.8**	**-0.3**	**-0.2**	**-0.2**	**100**	**98**

Table 4.8: BD-Rate coding results per sequence, in %, with MedianNBDV

Table 4.8 shows 0.6% and 0.8% average bitrate reductions on the dependent views, and 0.2% on the synthesized views, with a *MaxCand* of 14, as explained in Section 4.3.3. These gains were achieved with no increase on encoder and decoder runtimes. Note that no gains are reported on the central view because no DV derivation is done on the base view.

Table 4.9 shows the average coding results of three variants (1REF, 1REF+RMPOS, LIMIT-4) aimed at reducing *MaxCand*. These variants slightly reduce the gains obtained in MedianNBDV but alleviate the median computation in hardware in the worst-case scenario. The NODDV, ALLOWRED and MEAN variants however

Variant	Max Cand	Video				Synt.	Coded +Synt	Runtimes	
		0	1	2	Avg			Enc	Dec
MedianNBDV	14	0.0	-0.6	-0.8	-0.3	-0.2	-0.2	100	99
1REF	9	0.0	-0.6	-0.7	-0.2	-0.2	-0.2	97	98
1REF+RMPOS	7	0.0	-0.5	-0.6	-0.2	-0.2	-0.2	99	99
LIMIT-4	4	0.0	-0.5	-0.7	-0.2	-0.2	-0.2	103	98
NODDV	14	0.0	-0.3	-0.3	-0.1	-0.1	-0.1	97	99
ALLOWRED	14	0.0	-0.2	-0.3	-0.1	-0.1	-0.1	98	98
MEAN	14	0.0	-0.3	+0.1	0.0	0.0	0.0	100	98
NOAMVP	14	0.0	-0.6	-0.7	-0.3	-0.2	-0.2	101	98
APPLYRES	14	0.0	-0.6	-0.8	-0.3	-0.2	-0.2	108	98

Table 4.9: Average BD-Rate coding results for the different variants of MedianNBDV

keep the same *MaxCand* as MedianNBDV but significantly reduce gains (losses are even reported for the MEAN variant on the second dependent view). Finally, the NOAMVP variant slightly reduces gains while not affecting runtime, while the APPLYRES variant, on the contrary, achieves the same coding performance as MedianNBDV with the same *MaxCand* but with an increase in encoder runtime (108%).

Visual results

The significant gains on the dependent views for the Kendo and Dancer sequences in MedianNBDV are visible in Figure 4.5. Parts of the left view (view 1) and the right view (view 2) at a QP of 40 and 35 for the Kendo and Dancer sequences respectively, coded using the HTM-5.0.1 reference software and with MedianNBDV are shown in this figure. For the Kendo sequence, we can see that our method avoids having the sword broken in two as in the reference. For the Dancer sequence, the back of the dancer's head is more sharply represented using MedianNBDV.

4.4.3 Results interpretation

Origin of the gains

MedianNBDV improves the quality of the DV used in IVMP. Consequently, the multi-view and the inter-view candidates in the Merge list, which depend on that DV, are also improved and more often selected. Table 4.10 shows the increase in the number of PUs coded in Merge mode using the multi-view or the inter-view candidates, in MedianNBDV, for each tested sequence, averaged across four QPs. A significant increase is noted for the inter-view candidate (31% on average) since it directly corresponds to the improved DV. For the multi-view candidate, the improved DV is only used to find a PU in the base view from which to extract a MV. Consequently, the improved DV may point to a PU that has the same MV as the one

(a) Kendo V1 QP40 reference (b) Kendo V1 QP40 proposed

(c) Dancer V2 QP35 reference (d) Dancer V2 QP35 proposed

Figure 4.5: Parts of dependent views coded with the reference software and with the proposed method

of the PU pointed to by the original DV. In this case, our DV improvement has no effect, and this explains why on average the selection of the multi-view candidate has only slightly increased (2%) compared to an HTM-5.0.1 reference. In any case, these increases are directly correlated with the coding gains achieved in MedianNBDV.

Sequence	**Multi-view increase**	**Inter-view increase**
Kendo	0.6	23.3
Newspaper	1.4	21.5
Balloons	3.6	18.4
Dancer	3.5	54.8
GT Fly	0.5	62.9
Poznan Hall2	2.2	16.7
Poznan Street	1.4	20.1
Average	**1.9**	**31.1**

Table 4.10: Increase in the percentage of PUs coded in Merge mode using the multi-view or the inter-view candidates

Runtime results analysis

Furthermore, Table 4.11 shows the average, minimum and maximum number of vectors on which the median is computed for each tested sequence in the encoder

and the decoder, in MedianNBDV. We can see that the worst case scenario in which the median is computed on 14 values never occurs for any sequence (maximum is 12). On average, the median is computed on 1.9 vectors at the encoder and 2.2 vectors at the decoder, the difference being due to the fact that the encoder tests all possible CU sizes and partitions and hence performs the median computation much more often than the decoder. Some of these CUs tested by the encoder only have a few neighboring DVs. These CUs contribute in decreasing the average compared to the one evaluated at the decoder. In any case, most of the time, the median computation is simple and is performed quickly. This explains why the runtime increase at both coder sides was imperceptible. Indeed, this increase definetely exists since MedianNBDV necessarily adds some operations to the encoder and decoder without removing others, but it is not visible in Table 4.8 because it is really small.

Sequence	**Encoder**			**Decoder**		
	Avg	**Min**	**Max**	**Avg**	**Min**	**Max**
Kendo	1.9	1	11	2.2	1	9
Newspaper	1.9	1	11	2.1	1	10
Balloons	1.9	1	10	2.2	1	10
Dancer	1.9	1	10	2.2	1	10
GT Fly	2.3	1	12	2.4	1	12
Poznan Hall2	1.7	1	11	1.9	1	10
Poznan Street	2.0	1	11	2.2	1	11
Overall	**1.9**	**1**	**12**	**2.2**	**1**	**12**

Table 4.11: Average, minimum and maximum number of vectors for median computation at the encoder and decoder side

Variants results interpretation

The 1REF, 1REF+RMPOS and the LIMIT-4 variants all succeed in reducing *MaxCand* with a small penalty on coding gains (0.1% on coded texture videos). The performance of these three variants is roughly equivalent, but LIMIT-4 reduces *MaxCand* the most (to 4 instead of 9 or 7), making it clearly the best variant in this category. Note that Table 4.9 shows that LIMIT-4 increases the encoder runtime (103%) but as previously said, there is a ±3% error margin on this runtime so any increase below 103% or any decrease above 97% is not considered valid.

The gains are more significantly reduced in the ALLOWRED variant, in which the redundancy check on the vectors before median computation is not performed. This can be explained by the fact that the redundancy check allows to diversify the input vectors for the median computation, hence avoiding having the same DV

chosen over a contiguous region with different disparity values. In addition, the redundancy check reduces the average and maximum number of vectors (considered on all sequences) on which the median is computed. Indeed, our experiments show that these values would have increased to 4.0 and 14 at the encoder, and 4.8 and 14 at the decoder, respectively, if the check was not performed.

Storing only the DVs in the list while discarding DDVs also reduces the gains of our method. Indeed, not considering DDVs in the list penalizes MedianNBDV in case there are no DVs to insert because in that case, DV_{IVMP} is set as a zero vector, while in NBDV, a DDV may be chosen, which is almost always more accurate than a zero vector. This result also validates our intuition discussed in Section 4.3.3 about the fact that the heterogeneity in lists in a video coder is more efficient than homogeneity.

If we do not apply MedianNBDV for AMVP, the multi-view and the inter-view candidates in the AMVP list are not improved. Consequently, a slight reduction of the gains on the dependent views (0.1% loss) is noted with practically no influence on the encoder runtime. This validates our assumption that the contribution of improving the multi-view and inter-view AMVP candidates in MedianNBDV is small.

If MedianNBDV is applied for IVRP as well as IVMP, as in the APPLYRES variant, the average coding gains remain the same as in the original method. This is because improving the multi-view and the inter-view Merge candidates has a much higher impact than improving IVRP. However, the slight increase in encoder runtime in our method becomes multiplied by around 2.5 since IVRP is applied for all PUs, including PUs coded in Intra, as opposed to IVMP. As a consequence, it becomes visible as seen in Table 4.9. Hence, for a better coding efficiency / complexity tradeoff, MedianNBDV should not be applied for IVRP.

Finally, we have tested replacing the median computation with a simpler average computation (the MEAN variant). However, the coding gains obtained are small. Some losses are even reported for the second dependent view. This can be explained by the fact that the median allows to select a DV out of accurate, previously estimated DVs, whereas the average creates a new DV which might not truly describe the disparity at the level of the current PU.

4.5 Conclusion

In this chapter, we have presented two methods to improve IVMP in 3D-HEVC. The first method, AIMC, introduces an inter-view DV candidate in the Merge candidate list to increase DCP selection. Two insertion methods for AIMC have been proposed,

one where the DV is inserted in the secondary candidate list (AIMC-1) and another where the DV is inserted in the primary list (AIMC-2). Bitrate reductions of 0.5% (resp. 0.6%) and 0.6% for the two side views, along with 0.2% for synthesized and for coded+synthesized views are reported. These are accompanied by a 3% (resp. 4%) encoder runtime reduction since secondary candidates are less required to be constructed. AIMC-1 and AIMC-2 were presented at the 2nd JCT-3V meeting and AIMC-2 was adopted in 3D-HEVC.

The second method, MedianNBDV, tackles the sub-optimality problem in NBDV resulting from selecting the first DV or DDV found in the search. In MedianNBDV, all found DVs and DDVs in spatial and temporal neighboring PUs are stored together in a single list, and the search process is never stopped. Redundant vectors in the list are removed, and the median of the remaining vectors is computed and set as the final DV used for IVMP. Average bitrate reductions of 0.6% and 0.8% on the two dependent views, along with 0.2% on synthesized views are achieved with no increase in encoder and decoder runtimes. Additional results also show that the maximum number of vectors on which the median is computed can be reduced from 14 to 4, with only a limited impact on the coding results. Two conference papers were published for AIMC and MedianNBDV respectively, the first being for the IEEE International Conference on Image Processing (ICIP) 2013 [MJCPP13c], and the second for the IEEE workshop on Multimedia Signal Processing (MMSP) 2013 [MJPPC13].

In the future, AIMC can be improved by driving the insertion of the inter-view candidate with a criterion which estimates, using decoded information (to avoid additional signaling), the benefits of having the candidate in the list. In MedianNBDV, the selection of the final DV could be based on an R-D check applied on the candidates stored in the list. The DV selected would be the one yielding the lowest R-D cost. This requires sending the index of the DV in the list to the decoder, but the method might still bring significant gains. Hence, it is an interesting idea to consider for future work.

Chapter 5

Initialization, limitation and predictive coding of the texture and depth quadtrees in 3D-HEVC

Contents

3D-HEVC utilizes the quadtree-based coding structure described in Section 2.3.1 for both the texture and depth components. In HTM, the RDO process is performed

at each level of the quadtree to determine the best coding mode and partition size for a CU at that level. The best depth level for the quadtree of a CU is also determined using an R-D check. Tools aimed at speeding up the quadtree construction by skipping some of these R-D checks are included in HTM, but these encoder shortcuts are always accompanied by coding losses.

Furthermore, existing inter-component tools in 3D-HEVC are not designed to directly handle this quadtree coding structure. They can only influence its construction indirectly by favoring a specific inherited coding mode or prediction parameter amongst others, which can lead to a different partitioning of a CTU. In this chapter, we propose a novel inter-component tool that utilizes either the coded texture or depth quadtree to control the construction and coding of the other component's quadtree. The aim is to reduce encoder runtime while simultaneously improving coding efficiency. The main idea is to force a depth CU to be less or equally partitioned than its co-located texture CU. This means that if the depth is coded first, the texture quadtree will be initialized from the depth quadtree. Otherwise, the depth quadtree will be limited to the texture quadtree. Additionally, a predictive coding of the quadtree of a component using the other component's quadtree is performed to increase coding efficiency.

Section 1 presents tools designed to speed-up the construction of the texture and depth quadtrees. Section 2 analyzes the texture-depth quadtree relationship in order to prove the potential of the proposed texture quadtree initialization (QTI) and the depth quadtree limitation (QTL). Section 3 presents QTI, QTL, and their associated predictive coding part. The results for QTI and QTL are given in a fourth section, then analyzed in a fifth. Section 6 concludes this chapter while underlining possibilities for future work.

5.1 State-of-the-art

HTM includes three non-normative tools that directly impact the splitting and partitioning of CTUs. First, an Early Skip tool checks whether the R-D cost of the Skip mode in a 2N×2N partition is lower than a certain threshold. If that is the case, no other prediction modes are tested and the CU is no longer split (the recursion is stopped at this level). Second, the Coding Block Flag (CBF)-based early termination [GLL11], where after each Inter partition check, if there is no residual to code (i.e. if the CBF equals 0) in the CU, all subsequent Inter checks (except Inter mode in N×N) and Intra checks are no longer tested. Finally, the Early CU Termination tool (ECU) stops the CU split at a certain depth level if the SKIP mode in 2N×2N turned out to be the best mode at that level [CJ12].

In JMVM (reference software of the MVC standard), these three tools have been implemented and tested [SLA+11]. A technique that categorizes macroblocks (MBs) into either simple or complex mode regions MBs, and in which Inter mode checks (including time-consuming motion and disparity estimations) for the first category are skipped, is proposed as well in [SZL12]. Also in JMVM, the Previous Disparity Vector Disparity Estimation (PDV-DE) and Stereo-Motion Consistency Constraint Motion and Disparity Estimation (SMCC-MDE) shortcuts are proposed in [KHAF12] to reduce the search range for disparity estimation.

More recently, the Enhanced Depth CU (EDCU) [LK13] shortcut is proposed for the depth component only, where the recursive split of the depth CU is stopped if the best mode of the current depth CU is Skip and if the co-located texture CU is encoded in Skip mode as well. Other tools [SLZ+12, TLTY12] attempt to reach a good complexity-performance trade-off by implementing algorithms to make more intelligent early CU termination decisions.

However, all these tools are aimed at reducing encoder runtime at the expense of a decrease in coding efficiency as they do not allow any efficient coding of the quadtree syntax elements. Furthermore, these tools would not be able to exploit the relationship between the texture and the depth quadtrees as they are purely 2D coding tools. The proposed methods exploit this relationship to both reduce the encoder runtime and achieve coding gains.

5.2 Motivation

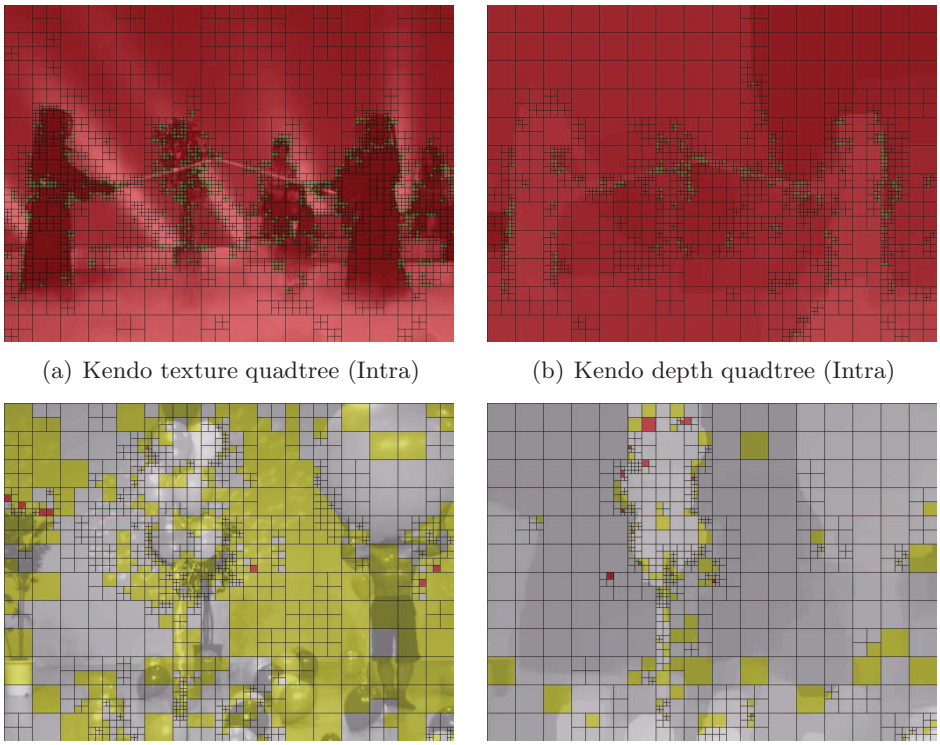
5.2.1 Comparison of the texture and depth quadtree

Our work in this chapter is based on the following assumption:

Assumption 1 *A texture CU is at least as partitioned as its co-located depth CU.*

Indeed, fine partitioning is usually performed along edges, to account for the lack of a correct prediction for a CU. Since texture has more edges than depth due to illumination changes, patterned textures and shadows (to which a depth map is invariant, as mentionned in Section 1.2.2), it is thus in general more partitioned. This can be seen in Figure 5.1 which presents the texture and depth coding and prediction quadtrees at QP 25 in an Intra and an Inter frame of the Kendo and Balloons sequences respectively.

Table 5.1 gives the percentage of CUs where Assumption 1 fails for four tested QPs and for one GOP of the sequences considered (assumptions 2 and 3 will be discussed in Section 5.3). The GOP consists of an 8-frame pyramid, with I, P and hierarchical B pictures. The test was done under HTM-4.0 in the same coding con-



(a) Kendo texture quadtree (Intra) (b) Kendo depth quadtree (Intra)

(c) Balloons texture quadtree (Inter) (d) Balloons depth quadtree (Inter)

Figure 5.1: Texture and depth quadtree partitions for a Kendo Intra frame and a Balloons Inter frame at QP 25

figuration and test conditions as described later in Section 5.4.1. We can see from

Sequence	**Assumption 1**				**Assumption 2**				**Assumption 3**			
	25	**30**	**35**	**40**	**25**	**30**	**35**	**40**	**25**	**30**	**35**	**40**
Kendo	38	15	6	2	15	4	1	0	43	18	8	3
Newspaper	56	31	16	6	27	13	6	1	60	34	17	7
Balloons	40	19	7	4	17	7	1	0	44	21	8	5
Dancer	15	16	10	4	3	5	3	0	21	20	12	5
GT Fly	21	11	4	2	5	4	1	0	25	12	4	2
Poznan Hall2	32	11	4	2	12	3	1	0	34	13	4	2
Poznan Street	31	13	6	2	9	3	1	0	36	15	7	2

Table 5.1: Percentage of CUs per sequence and per QP where Assumptions 1, 2 and 3 fail

this table that the assumption seems reasonable. It holds particularly well for the Dancer and the GT Fly sequences because they are computer-generated sequences with very clean depth maps. Indeed, the depth maps of the other sequences considered are estimated using stereo matching algorithms and thus, they inherently contain artefacts. These artefacts sometimes come out as false edges in the depth map which cause a fine partitioning of an area that is supposed to be flat. This area would however be coarsly partitioned in texture, hence breaking the initial as-

sumption. These false edges are smoothed out at lower bitrates, which is why the assumption failure percentage decreases from QP 25 to QP 40.

Furthermore, the assumption does not hold if we have two adjacent objects in texture with similar luminance and chrominance but with different depths. In that case, the CUs containing both objects will be split in depth and not in texture. Particular motions (zooming for instance) also break the assumption. However these cases are rare, and this is confirmed by the results of Table 5.1.

5.2.2 Analysis of the quadtree coding cost

Three syntax elements, sent for each CU, control the quadtree coding structure for texture and depth. First, the **split_flag** syntax element is sent to signal whether a CU has been split into four sub-CUs or not. If the CU is at the highest depth level, it cannot be further split, and hence the split flag is not sent. If the CU is not split, a **skip_flag** syntax element is sent to signal whether the CU has been coded in SKIP mode or not. If not, the **partition_size** syntax element signals the partition shape which has been selected for coding that CU. Table 5.2 shows the percentage of bits used to code the split flag and the partition size per slice type and for one entire GOP (in the texture and depth bitstreams). The same software basis, test conditions and coding configuration as the ones used to get the results of Table 5.1 were used in order to get these percentages. Results shown here are averaged across four QPs (25, 30, 35 and 40).

Sequence	**Texture**				**Depth**			
	I	**P**	**B**	**GOP**	**I**	**P**	**B**	**GOP**
Kendo	3.2	6.1	16.6	7.5	8.2	12.1	19.4	13.3
Newspaper	3.1	7.4	22.0	5.7	9.4	14.3	20.3	13.6
Balloons	3.2	8.0	20.8	6.5	9.3	14.5	18.6	13.4
Dancer	2.9	9.1	17.9	7.0	14.6	22.8	24.2	21.9
GT Fly	4.2	11.1	21.5	8.8	10.6	17.3	18.9	16.1
Poznan Hall2	4.7	8.6	16.2	8.3	13.3	18.2	18.4	17.2
Poznan Street	3.7	7.7	14.8	7.0	10.4	15.8	20.4	15.5
Average	**3.6**	**8.3**	**18.5**	**7.2**	**10.8**	**16.4**	**20.0**	**15.8**

Table 5.2: Percentage of bits per slice type for the "split_flag" plus the "part_size" elements in the texture and depth bitstreams

This table shows that there is a significant amount of bits used to code the split flag and the partition size. The bitrate percentage for these two elements is especially high on P and B slices for depth and on B slices for texture. And although on average, the percentage in the GOP is higher for depth, our experiments show that the texture as a whole represents around 90% of the entire texture+depth bitstream in HTM-4.0.

Consequently the quadtree information of the texture represents 6.5% (7.2% × 0.9) of the entire bitstream, while the quadtree information for depth represents only 1.6% (15.8% × 0.1).

5.2.3 Conclusion

Since the assumption seems reasonable judging from Table 5.1, the encoder runtime is expected to be reduced if the quadtree structure of a texture CTU is directly initialized from that of the depth or if the quadtree structure of a depth CTU is limited to that of the texture. The encoder runtime reduction can be evaluated fairly in both components since the texture and depth runtimes are roughly the same in HTM-4.0 (coding all texture views takes, on average, 55% of the total encoding time of the sequences, while coding the depth views takes the remaining 45%).

Furthermore, Table 5.2 shows that the cost of the quadtree syntax element is significant, especially in texture. There is much to be gained from an efficient coding of these elements.

Consequently, a texture quadtree initialization using depth and a depth quadtree limitation using texture, with a quadtree predictive coding scheme in both methods can achieve both encoder runtime reduction and coding gains.

5.3 Proposed methods

5.3.1 Texture quadtree initialization

Proposed scheme

When the depth is coded before the texture, the texture encoder has access to the quadtree information (split flag and partition size) of the co-located CU in depth to control the quadtree of a currently coded texture CU. Based on our assumption, we propose to force a texture CU to be at least as partitioned as its corresponding depth CU. In other words, the quadtree of the texture CU is *initialized* from the quadtree of the depth CU; it can be further partitioned but not less. This has two benefits: it can reduce encoder runtime since certain coding modes are no longer checked, and allow a predictive coding of the quadtree syntax elements of the texture which, in turn, will increase coding efficiency. The rest of this section will detail the texture quadtree initialization (QTI) and its associated predictive coding part (PC).

Figure 5.2 shows all possible depth CU partitions and the allowed texture CU partitions in each case, at a specific depth level L. It does not show the allowed texture sub-CU partitions at level $L+1$ if the CU at depth level L was split. Indeed, this scheme is recursive; it can be applied the same way for each sub-CU.

If the depth CU is split or has a partition size of N×N , as in case (a), the texture CU is forced to be split or partitioned in N×N respectively. Other partitions are not checked, and no split flag or partition size is sent for the texture CU. In case (b) and (c), the only partition allowed for the texture CU is N×2N (respectively 2N×N). Splitting the texture CU is also possible. Hence, the encoder only sends the split flag. If the CU is not split, the partition size is inferred from the depth CU. In case (d), all texture modes and partitions are checked, the encoder needs to send both the split flag and the partition size in this case. Note that in all cases, splitting the texture is always an option. Hence, in QTI, the recursion is never stopped at any level, it is the starting level that is set.

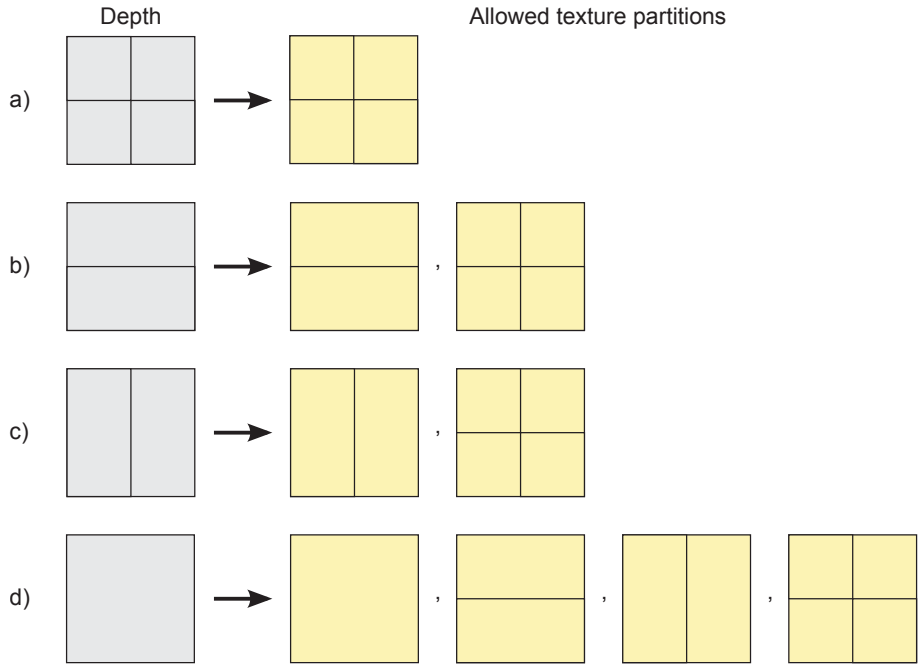


Figure 5.2: Allowed texture partitions in QTI

Flexible QTI+1 variant

The proposed scheme sometimes leads to a sub optimal (R-D wise) partitioning of a texture CU as it can force an unnecessary split or partition when an assumption failure occurs. The texture quadtree is said to be altered, meaning it has changed from what the RDO process intended it to be. We define the severity of a scheme in QTI by the amount of forced split and partitions the scheme imposes in a CTU. The least severe scheme is where no forcing is done (reference coding). The most severe scheme is where each CU is forced to be split or partitioned in N×N. Forcing splits and partitions however removes the need of sending split flags and / or partition sizes. Consequently, the more severe the scheme is, the more it alters the texture quadtree,

but the more bitrate reduction it allows as well. An efficient coding scheme has a severity level that achieves a good compromise between the amount of alteration and the amount of bitrate reduction.

In this chapter, we propose a less severe scheme for QTI that we call QTI+1 which relies on the following assumption:

Assumption 2 *A texture CU is at least one depth level less partitionned than its co-located depth CU.*

This scheme forces to split the texture CU at depth level L only if the depth CU is split at depth level $L+1$ (hence the name), as shown in Figure 5.3. The split flag for the texture CU does not need to be sent in this case. In any other case, the decision is left for the R-D based optimization process, and consequently the split flags and partition sizes need to be sent. Compared to Assumption 1, Assumption 2 fails less often, as can be seen in Table 5.1. Hence, this flexible scheme alters the texture quadtree less often but at the expense of a lower bitrate reduction potential. Note that this scheme is recursive as well, it can be applied in the same way for each sub-CU at depth level $L+1$ if the CU at level L is split.

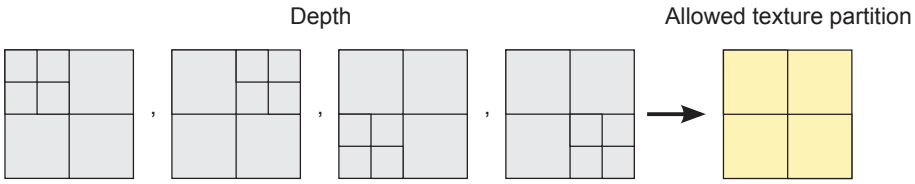


Figure 5.3: The QTI+1 variant

5.3.2 Depth quadtree limitation

Proposed scheme

When texture is coded first, the depth encoder has access to the split flag and the partition size of the co-located texture CU to control the quadtree of a currently coded depth CU. The same assumption as in QTI is considered: a depth CU is, at most, as partitioned as its co-located texture CU. In other words, the depth quadtree is *limited* to that of the texture. This also has two benefits since encoder runtime can be reduced, and coding efficiency increased. The rest of this section will detail the depth quadtree limitation (QTL) and its associated predictive coding part (PC).

Figure 5.4 shows the allowed depth partitions per possible texture partition in QTL. If the texture is partitioned either in 2N×2N, N×2N, or 2N×N as in case (a), the depth is forced to be in 2N×2N. The encoder will not check smaller partition sizes, and will not try to split the depth CU. Also, it does not need to send the split

flag or partition size for that CU. In case (b), all partition sizes are checked and splitting the depth CU is allowed. The split flag and partition size need to be sent. This case does not yield any encoder runtime reduction nor coding gains compared to a reference coding.

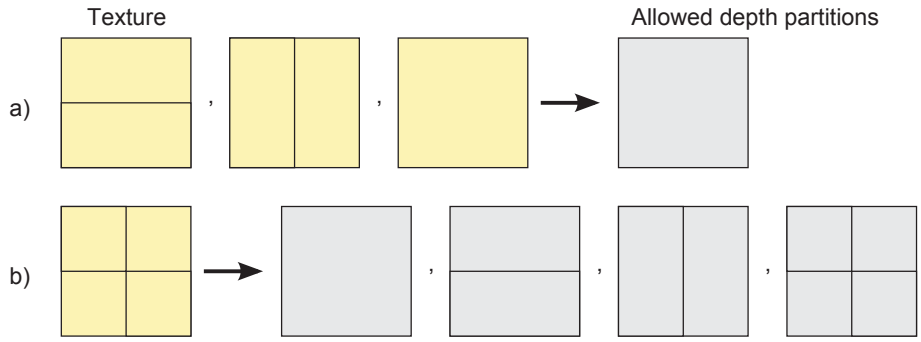


Figure 5.4: Allowed depth partitions in QTL

Strict QTL-1 variant

The proposed depth quadtree limitation scheme has a certain level of severity. Just like in QTI, we can adjust this severity level to obtain a better trade-off between the amount of quadtree alteration and the potential of bitrate reduction. A more severe scheme for QTL was thus studied in this work, which relies on the following assumption:

Assumption 3 *A depth CU is at most one depth level less partitionned than its co-located texture CU.*

In this scheme, the depth CU at depth level L is only allowed to be split or rectangularly partitioned if the texture is split at depth level $L+1$. In any other case, the depth CU is forced to be partitioned in 2N×2N. This scheme, called QTL-1, is more severe than QTL, because no split flags or partition sizes are sent for both cases of the original QTL scheme. However, the alteration level is also higher since there are more assumption failures, as can be seen from Table 5.1.

5.3.3 Impact on the codec architecture

QTL+PC and QTI+PC include a predictive coding scheme where the split flag and/or the partition size for a CU are not sent in some cases. Similarly as for the MPI tool, this implies a parsing dependency since the decoder needs to check the texture or depth quadtree to know whether to parse or not the two syntax elements. It is most likely that any realistic implementation of 3D-HEVC will perform at most a parallel parsing with one CTU delay given that parallel decoding is forbidden

by the following tools: DMM and MPI. This is possible with both QTL+PC and QTI+PC.

On the encoder side, this whole set of tools enables a one CTU delay parallel encoding, whatever the component coding order. This one CTU delay parallelization strategy also enables a very limited increase of the memory requirements, resulting from the storage of the quadtree information due to QTL, MPI or QTI, tiny compared to other data to be stored at the CTU level.

5.4 Experimental results

5.4.1 Experimental setting

We implement QTI, QTL, their predictive coding parts, and their variants in HTM-4.0 [Tecb]. The standard texture-before-depth coding order in HTM-4.0 allows testing QTL+PC. An extension of the HTM-4.0 software [Tecc] includes the Flexible Coding Order (FCO) tool. FCO allows changing the coding order to depth-before-texture for dependent views only, which in turn allows testing QTI+PC for the side views. FCO cannot change the coding order in the base view since the processing of the base view needs to remain HEVC compatible. Hence QTI+PC cannot be applied for the base view. In our QTI experiments, the coding of the base view remains consequently unchanged. It is important to note that we could apply QTL+PC in the base view to code the depth, but that would not allow for a fair evaluation of QTI since results would be mixed in with QTL results.

Aside from the changed coding order for QTI, we strictly follow the CTCs [RMV13]. Note that the following encoder shortcuts are enabled to conform to CTCs: fast encoder control (FEN), fast decision for Merge RD cost (FDM).

5.4.2 QTI results

Tables 5.3, 5.4 and 5.5 present the BD-Rate coding results of QTI, QTI+PC, and the variant QTI+1+PC (positive values are losses). QTI succesfully reduces encoder runtime by 10% because certain coding modes and partition sizes are no longer checked for CUs, as explained in Section 5.3.1. This is however accompanied by around 1% bitrate increase on coded views and on synthesized views.

When the predictive coding part is added, the BD-Rate losses are largely reduced. A bitrate reduction of 1.3% and 0.4% is achieved on side views and on coded and synthesized views respectively compared to QTI alone. This comes with a small increase in encoding runtime due to memory accesses to the depth component and additional checks, making a total runtime reduction of 6%. The bitrate reductions

Sequence	Video				Synt.	Coded+Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	2.6	2.2	0.9	0.8	0.8	91	100
Kendo	0.0	3.4	3.5	1.4	1.1	1.1	87	96
Newspaper	0.0	5.3	3.8	1.7	1.4	1.4	87	99
GT Fly	0.0	3.0	3.5	0.8	0.7	0.7	87	100
Poznan Hall2	0.0	2.8	2.6	1.1	0.9	0.9	92	99
Poznan Street	0.0	1.8	2.0	0.6	0.5	0.5	93	100
Dancer	0.0	2.4	2.5	0.7	0.6	0.6	92	99
Average	**0.0**	**3.0**	**2.9**	**1.0**	**0.9**	**0.9**	**90**	**99**

Table 5.3: BD-Rate coding results per sequence, in %, of QTI

Sequence	Video				Synt.	Coded+Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	1.7	1.4	0.6	0.5	0.5	90	100
Kendo	0.0	1.9	2.1	0.8	0.7	0.7	91	97
Newspaper	0.0	3.0	2.3	1.0	0.9	0.8	87	93
GT Fly	0.0	1.7	2.2	0.5	0.4	0.4	94	103
Poznan Hall2	0.0	1.3	1.6	0.6	0.5	0.5	104	101
Poznan Street	0.0	1.1	1.1	0.3	0.3	0.3	97	101
Dancer	0.0	1.1	1.1	0.3	0.3	0.3	101	108
Average	**0.0**	**1.7**	**1.7**	**0.6**	**0.5**	**0.5**	**95**	**100**

Table 5.4: BD-Rate coding results per sequence, in %, of QTI+PC

Sequence	Video				Synt.	Coded+Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	0.1	0.0	0.0	0.0	0.0	87	105
Kendo	0.0	0.2	0.2	0.1	0.1	0.1	86	96
Newspaper	0.0	0.1	0.1	0.0	0.1	0.1	86	92
GT Fly	0.0	0.3	0.6	0.1	0.1	0.1	92	117
Poznan Hall2	0.0	-0.1	0.0	0.0	0.0	0.0	105	101
Poznan Street	0.0	-0.2	0.2	0.0	0.0	0.0	102	99
Dancer	0.0	0.1	0.1	0.0	0.0	0.0	101	99
Average	**0.0**	**0.1**	**0.2**	**0.0**	**0.1**	**0.0**	**94**	**101**

Table 5.5: BD-Rate coding results per sequence, in %, of QTI+1+PC

however are not large enough to compensate all the losses in QTI, hence the remaining overall BD-Rate losses in QTI+PC.

The QTI+1 variant relaxes the severity of the scheme by only forcing to split a texture CU if the co-located depth CU is split at the next depth level. Here, the texture quadtree is less altered but the bitrate reduction potential is also reduced. QTI+1+PC still achieves the same encoder runtime savings as QTI+PC but with only a small coding loss of 0.1% on coded and synthesized views.

5.4.3 QTL results

Objective results

Tables 5.6, 5.7 and 5.8 shows the BD-Rate coding results and runtimes of QTL, QTL+PC, and QTL-1+PC. Notice that in these tables, the "Video" columns are removed since QTL does not affect texture data. A new column, "Depth", is added to show BD-Rate coding results evaluated only on the depth component. Only the depth PSNR and depth bitrate are considered in this computation. Another column, "Video total" is added wherein the average PSNR of the coded texture videos is measured against the total texture+depth bitrate. Since QTI does not affect depth, these two columns are not shown in Tables 5.3, 5.4 and 5.5.

Sequence	**Depth**	**Video total**	**Synt.**	**Coded+Synt.**	**Runtimes**	
					Enc	**Dec**
Balloons	-15.7	-1.4	1.3	0.4	68	98
Kendo	-11.5	-1.3	0.9	0.2	64	98
Newspaper	-15.1	-1.8	2.0	0.7	64	100
GT Fly	-20.6	-1.2	0.0	-0.4	62	100
Poznan Hall2	-24.5	-2.3	0.6	-0.3	69	98
Poznan Street	-15.3	-1.0	0.1	-0.2	74	87
Dancer	-37.8	-1.0	0.2	-0.2	65	99
Average	**-20.1**	**-1.4**	**0.7**	**0.0**	**66**	**97**

Table 5.6: BD-Rate coding results per sequence, in %, of QTL

As seen in Table 5.6, QTL achieves a significant 34% encoder runtime reduction, while introducing coding losses on synthesized views. The predictive coding part brings an additional 0.3% bitrate reduction in both the synthesized and the coded+synthesized columns but with a little runtime penalty due to the additional access to texture data. The QTL-1+PC variant is a more radical scheme: it achieves 34% encoder runtime reduction, but at the expense of a bigger coding loss of 1.1% on synthesized views. Consequently, QTL+PC achieves a better coding gain vs. encoder runtime reduction trade-off than QTL-1+PC.

Sequence	Depth	Video total	Synt.	Coded+Synt.	Runtimes	
					Enc	Dec
Balloons	-18.2	-1.7	1.0	0.1	76	102
Kendo	-14.3	-1.6	0.6	-0.2	68	99
Newspaper	-17.9	-2.3	1.6	0.3	66	100
GT Fly	-23.7	-1.4	-0.2	-0.6	70	99
Poznan Hall2	-28.1	-2.9	0.0	-0.9	65	97
Poznan Street	-18.7	-1.2	-0.1	-0.5	69	87
Dancer	-40.6	-1.3	-0.1	-0.5	69	99
Average	**-23.1**	**-1.8**	**0.4**	**-0.3**	**69**	**97**

Table 5.7: BD-Rate coding results per sequence, in %, of QTL+PC

Sequence	Depth	Video total	Synt.	Coded+Synt.	Runtimes	
					Enc	Dec
Balloons	-24.7	-2.2	1.6	0.4	71	103
Kendo	-15.8	-2.0	1.4	0.3	65	98
Newspaper	-23.4	-2.9	2.8	0.9	66	103
GT Fly	-37.3	-2.0	0.2	-0.5	67	99
Poznan Hall2	-35.3	-3.6	0.6	-0.7	61	99
Poznan Street	-25.5	-1.6	0.2	-0.3	70	88
Dancer	-47.3	-1.9	0.6	-0.2	64	99
Average	**-29.9**	**-2.3**	**1.1**	**0.0**	**66**	**98**

Table 5.8: BD-Rate coding results per sequence, in %, of QTL-1+PC

Subjective results

Table 5.7 shows that QTL+PC gives losses, on average, on synthesized views. These are due to smoothing out wrong edges in depth, which is in fact an improvement brought by the tool. To show that QTL+PC does not add any new artefacts on synthesized views, a subjective viewing session was conducted during the 2nd JCT-3V meeting [JM12b]. A 3D and a 2D test were performed. For each test, two sequences (Balloons and Newspaper) at two QPs (25 and 35) each were evaluated. QP 25 was selected because it yields the highest coding losses on synthesized views. QP 35 was selected to provide complementary information for lower bitrates.

In the 3D test, a stereo sequence was constructed from the two synthesized views that are closest to the center view, and was projected onto a 3D screen. In the 2D test, the closest synthesized view to the left of the center view was selected for projection. These synthesized views were rendered from texture and depth coded using the reference software HTM-4.0 on the one hand, and with our tool enabled on the other hand. Hence two sequences, A and B, were projected to nine (5 JCT-3V experts, and 4 non-experts) viewers in this order: A B A B. The viewers did not

Test	Sequence + QP	Viewer									AVG	AVG / Test
		1	2	3	4	5	6	7	8	9		
3D	Balloons 25	0	-1	-1	-1	0	0	-1	-1	0	-0.6	-0.1
	Newspaper 25	0	0	0	1	0	0	0	0	0	0.1	
	Balloons 35	1	1	0	0	0	1	1	0	0	0.4	
	Newspaper 35	0	0	-1	0	0	0	0	-1	0	-0.2	
2D	Balloons 25	1	0	0	0	0	0	0	1	0	0.2	-0.2
	Newspaper 25	-1	-1	-1	-1	0	0	-1	0	-1	-0.7	
	Balloons 35	0	0	0	1	-1	-1	0	1	0	0	
	Newspaper 35	0	-1	0	-1	-1	1	0	-1	0	-0.3	

Table 5.9: Coding scores from subjective viewing experiments for 2D and 3D tests

know what A and B corresponded to. Actually, A was set as the reference for some viewings and as the proposed method for others. The viewers were asked to rate if A is largely better (a score of +3 is given), better (+2), slightly better (+1), same as (0), slightly worse (-1), worse (-2) or largely worse (-3) than B. Table 5.9 shows the coding scores for the 2D and the 3D test, where a negative value represents an improvement resulting from the use of the proposed QTL+PC method.

In both the 2D and the 3D test, no score above 1 or below -1 is reported. It is thus confirmed that no new annoying artifact is noticeable. In the 3D test, results were particularly consistent: for each sequence, there was either no difference, or one of the method claimed as slightly better. Said differently, for one given experiment, it is not possible to find a score of +1 and -1 simultaneously. The proposed method is even favored with an average score of -0.1 and -0.2 in the 3D and 2D case respectively. This confirms that smoothing out the wrong edges in QTL+PC actually tends to be beneficial.

5.4.4 Comparison with state-of-the-art encoder shortcuts

Table 5.10 shows the average coding gains of QTL+PC and QTI+1+PC compared to state-of-the-art encoder shortcuts: ECU, CBF and EDCU, defined in Section 5.1. ECU and CBF are 2D video shortcuts, applicable to both texture and depth components. Thus, for fairness of evaluation, we have compared ECU and CBF applied on texture only (resp. depth only) with QTI+PC (resp. QTL+PC). EDCU is a depth only encoder shortcut and was thus compared only with QTL+PC.

Table 5.10 shows that QTL+PC achieved more bitrate and encoder runtime reductions than its competitors. For texture coding, ECU-T and CBF-T achieved more encoder runtime reductions than QTI+1+PC, at the expense of a bigger loss in the “Video Total” column.

Method	Video				Video total	Synt.	Enc
	0	1	2	Avg			
QTL+PC	0.0	0.0	0.0	0.0	-1.8	0.4	69
ECU-D	0.0	0.0	0.0	0.0	-2.2	1.1	76
EDCU	0.0	0.0	0.0	0.0	-1.4	0.4	82
CBF-D	0.0	0.0	0.0	0.0	-0.2	0.3	85
QTI+1+PC	0.0	0.1	0.2	0.0	0.0	0.1	94
ECU-T	0.3	-1.2	-1.1	0.5	0.7	-0.1	63
CBF-T	0.9	0.5	0.5	0.9	1.0	0.7	72

Table 5.10: Average BD-Rate coding results of QTI+1+PC, QTL+PC and other recent encoder shortcuts

5.5 Results interpretation and analysis

5.5.1 Analysis of QTI results

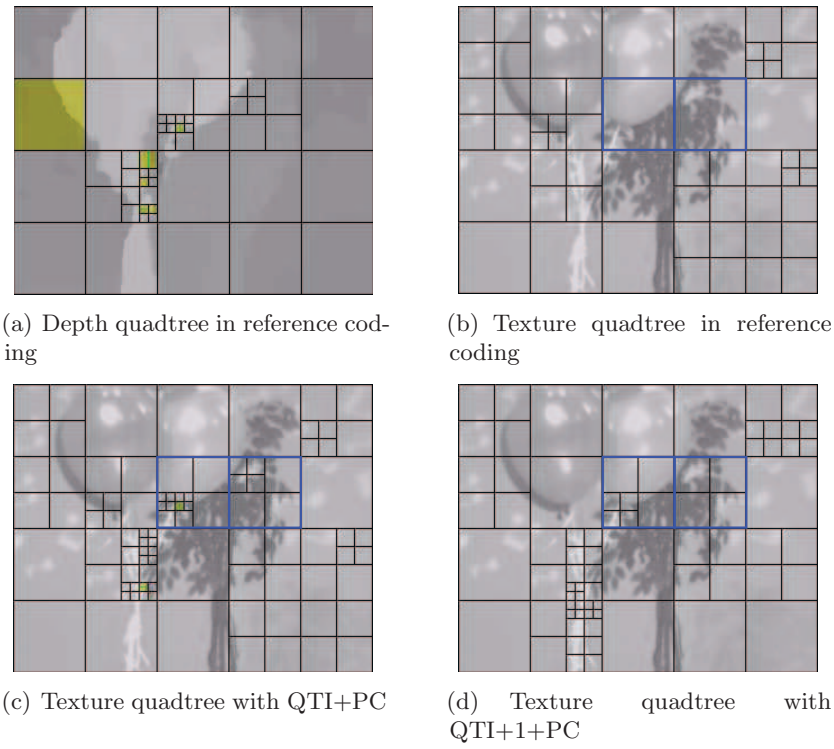
Table 5.11 shows in which cases encoder runtime savings, bitrate savings and coding losses occur in QTI+PC. When a depth CTU is homogeneous (not split), QTI+PC

CTU		Texture	
		Homogeneous 64×64	Textured/Edge 8×8
Depth	Homogeneous 64×64	QTI+PC: no impact QTL+PC: runtime saving + bitrate reduction	QTI+PC: no impact QTL+PC: no impact
	Textured/Edge 8×8	QTI+PC: coding loss + runtime saving QTL+PC: runtime saving + bitrate reduction + synth. PSNR drop	QTI+PC: bitrate reduction + runtime saving QTL+PC: bitrate reduction

Table 5.11: Analysis of impact on coding loss, bitrate reduction and encoder runtime in QTI+PC and QTL+PC under different combinations of depth and texture CTUs

does not impact the texture quadtree construction nor its coding. When a depth CTU contains an edge and is split accordingly, if the texture CTU is split as well, QTI brings encoder runtime savings since coding modes and partition sizes are no longer checked for low depth level CUs. The predictive coding part further brings bitrate reductions since the split flag and partition size for low level CUs are not sent in the bitstream. If the texture CTU is homogeneous while the depth CTU is split, our initial assumption no longer holds. Runtime savings are achieved but coding losses occur since the texture quadtree is altered. This is shown in Figure 5.5: while the highlighted texture CTUs are homogeneous in reference coding as seen

in Figure 5.5(b), they become as partitioned as the co-located depth CTUs with QTI+PC, as seen in Figure 5.5(c). This situation is relatively unfrequent, as can be seen from Table 5.1, but it is responsible for the coding losses shown in Table 5.3 for QTI alone.



(a) Depth quadtree in reference coding

(b) Texture quadtree in reference coding

(c) Texture quadtree with QTI+PC

(d) Texture quadtree with QTI+1+PC

Figure 5.5: Texture quadtree in reference coding, with QTI+PC and QTI+1+PC. Gray CUs are coded in SKIP mode, and green CUs in Inter

We can see from Table 5.11 that encoder runtime savings in QTI only occur when the depth CTU is split. Actually, they are possible only if a depth CU is split or partitioned in N×2N or 2N×N, which correspond to cases (a), (b) and (c) respectively in Figure 5.2. The percentage of these three cases, that we call FT for Favorable for Texture, is shown in Table 5.12 for each sequence. This table shows that overall, the percentage of CUs wherein encoder runtime savings are possible is only 9.2% which explains why the runtime savings in QTI were not larger than 10%.

Furthermore, the predictive coding part of the tool brings bitrate reductions only on those 9.2% of CUs as well. Consequently, the amount of bitrate reduction is not sufficient to compensate all the losses in QTI that are due to assumption failures. Hence, coding losses are still reported in QTI+PC. In QTI+1+PC, the more flexible scheme reduces the alteration of the texture quadtree while decreasing the FT percentage in the process. Figure 5.5(d) indeed shows that QTI+1+PC allowed the highlighted texture CTUs, to be one level less partitioned than the

Sequence	FT	FD
Kendo	10.3	82.1
Newspaper	11.4	87.0
Balloons	8.8	87.5
Dancer	11.4	80.9
GT Fly	7.4	84.5
Poznan Hall2	5.5	89.2
Poznan Street	9.8	82.5
Average	**9.2**	**84.9**

Table 5.12: The FT and FD percentages per sequence

depth CTUs, hence reducing the texture quadtree alteration compared to QTI+PC, and the losses that come along with it. The resulting trade-off nearly removes all coding losses (only a small 0.1% loss is reported).

When applied on the texture component only, ECU reduces significantly the encoder runtime by 37% because it is applied often. It gives losses on the central view but significant gains on the dependent views. Indeed, a significant PSNR drop is noticeable in these views, but well compensated by an equally significant bitrate reduction. This PSNR drop is no longer compensated when taking into account the PSNR and bitrate of the central view hence the 0.5% loss in the "Video Avg" column in Table 5.10. When also taking into account the depth bitrate in the BD-Rate computation, the results are even worse (0.7% loss). QTI+1+PC is applied less often than ECU, hence only reducing encoder runtime by 6%. However, it also does not alter the texture enough to cause such a major PSNR drop, hence not presenting any coding loss when evaluating the coded views PSNR against the coded views bitrate or against the coded views + depth bitrate. In mobile applications for instance where encoder runtime savings are primordial, ECU on texture could be preferred over QTI+1+PC. In other applications which do not involve view synthesis, and where only the R-D performance of the coded texture views counts, QTI+1+PC could be more suitable than ECU-T. Note that QTI+1+PC also gives lower losses in general than CBF-T in all evaluation scenarios, although CBF-T gives larger runtime reductions.

5.5.2 Analysis of QTL results

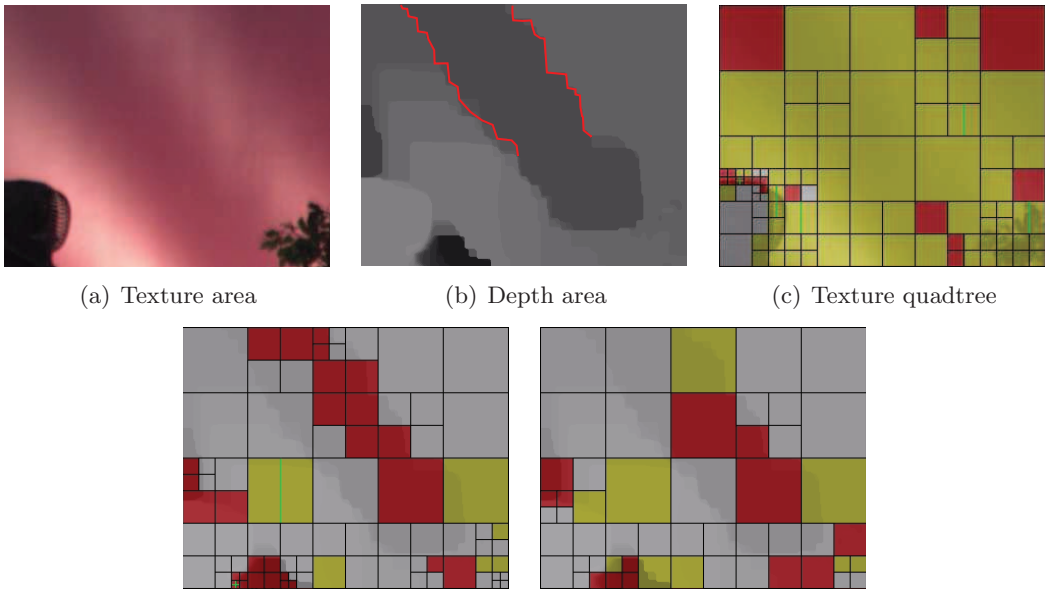
Table 5.11 shows that when the texture and depth CTUs are homogeneous, runtime savings are achieved since coding modes and partition sizes are no longer checked for high level CUs in depth. The recursion is stopped at depth level 0, hence bringing significant runtime reduction. Also, a small bitrate saving is achieved with the predictive coding part since neither the split flag or the partition size of the CTU

needs to be sent. If only the depth is split to achieve better prediction on an edge, then our assumption no longer holds. In this case, the runtime and bitrate reductions are accompanied by a loss of quality on synthesized views. This does not happen very frequently however, as can be seen in Table 5.1. When the texture is split, if the depth CTU is homogeneous, QTL+PC has no impact on coding the depth CTU. If the depth CTU is also split, only bitrate reductions are achieved by not sending the quadtree information for high level depth CUs.

We can see from Table 5.11 that the CUs wherein bitrate reductions and runtime savings are possible in depth have an homogeneous co-located texture CU. Actually, these reductions occur when the texture CU is partitioned in 2N×2N, N×2N, or 2N×N, which correspond to case (a) in Figure 5.4. The percentage of these CUs, that we call Favorable for Depth (FD), is given in Table 5.12, for each sequence. Overall, the FD percentage equals 84.9%, which is relatively high. This explains the significant runtime reduction and bitrate savings in QTL+PC. The large difference between the FT and FD percentages also explains the performance gap between QTI+PC and QTL+PC in terms of runtime reductions. The predictive coding part achieved however nearly the same coding gains in both tools, as it gave -0.3% and -0.4% on coded+synthesized in QTL and QTI respectively. Since FD is much higher than FT, the predictive coding in QTI should have given less gains. However, the cost of transmitting the split flag and partition size in the bitstream is higher in texture than in depth (6.5% against 1.6%), as shown in Section 5.2.2, so even though the predictive coding is less used in QTI, it has more impact there than in QTL.

The example in Figure 5.6 shows how QTL successfully smoothes out wrong edges in depth. Indeed, the texture background area, as shown in Figure 5.6(a) is planar. Its corresponding CTUs are not split (or at most, split once). In depth, the corresponding area should normally be planar as well, but a badly performed stereo matching created a wrong edge in that area, highlighted in Figure 5.6(b). Consequently, when QTL is not used, the CTUs corresponding to that depth area are split, as seen in Figure 5.6(d). When QTL is used, the depth quadtree is limited to the texture quadtree: the depth CTUs are not split, hence smoothing out the wrong edge, as shown in Figure 5.6(e).

These wrong edges are still present in reference coding, and when comparing to an uncompressed synthesized view, rendered with an original depth that contains these wrong edges, the reference coding appears to be better than QTL+PC. This is visible in the "Synth." column in Table 5.7 which reports losses, due to a PSNR drop on synthesized views from smoothing out wrong edges. While there is a significant depth bitrate reduction brought by QTL+PC, evaluated at 23.1% on the depth



(a) Texture area (b) Depth area (c) Texture quadtree

(d) Depth quadtree with reference (e) Depth quadtree with QTL+PC

Figure 5.6: Depth quadtree with reference coding and with QTL. Gray CUs are coded in SKIP mode, green CUs in Inter and red CUs in Intra

component as shown in Table 5.7, this reduction is not able to compensate this PSNR drop.

However, smoothing out the wrong edges in depth is actually an improvement since the quality of the synthesized views will be increased. Indeed, the subjective results in Table 5.9 confirm that there is no actual degradation in the quality of the synthesized views. In fact, on average, our tool was found to be better than the reference, especially on the 2D test where the potential artefacts resulting from the use of our tool, if any, would be more noticeable anyway.

In the coded+synthesized column, the PSNR considered is the average between the 6 PSNRs of the synthesized views and the 3 PSNRs of the coded texture views, which do not change using our method since QTL does not affect the texture component. Consequently, the PSNR drop on the synthesized views is attenuated in this column, and can therefore be compensated by the bitrate reduction on the depth component, leading to the shown overall BD-Rate gains.

Furthermore, since neither the PSNR nor the bitrate of the coded texture videos change in QTL+PC, the depth bitrate reductions are directly visible in the "Video total" column, and although attenuated by the added texture bitrate, they are still significant.

Compared to QTL+PC, ECU applied only on depth achieves lower runtime reduction than QTL+PC. ECU however gives a more significant bitrate reduction on

the depth component alone, which is visible in the "Video total" column in Table 5.10 (-2.2% gain). ECU, which was presented initially as a 2D encoder shortcut in HM, is based on the assumption that if a CU is best coded in Skip mode, splitting it further would not allow to find a better configuration, R-D wise. Altough the assumption is valid for texture, it is not appropriate for depth since any failures in this assumption would cause smoothing out an edge in depth, hence heavily impacting the synthesis. And unlike in QTL+PC, these are not necessarily wrong edges. Consequently, a significant PSNR drop on the synthesized views is observed, causing a 1.1% loss. These losses are not present in QTL+PC. EDCU conditions the application of ECU using an additional constraint: the co-located texture CU must be coded in Skip as well. Consequently, EDCU is less applied than ECU, hence giving lower runtime reductions (18% instead of 24%), but also lower losses on synthesized views, and an R-D performance only slightly worse than QTL+PC. As for CBF applied only on depth, it gives a worse R-D performance than QTL+PC and a lower runtime reduction.

Finally, note that at the time of developping QTI+PC and QTL+PC, the most recent HTM software version was HTM-4.0. Since then, many tools have been added or removed from the software. In the more recent HTM-8.0 version, released in September 2013, the coding performance of QTL+PC is only slightly reduced: QTL+PC achieves a 1.2% loss on synthesized views (but as said earlier, these are not real losses) and -1.4% gain in the "Video total" column. However, the encoder runtime reductions QTL+PC offers have increased to 41%. This is due to the adoption in HTM-8.0 of two encoder shortcuts for texture: early Merge mode decision and early CU splitting termination [ZCL+13]. The first shortcut omits the examination of Intra and Inter modes after Merge and Skip modes if all five inter-view neighboring blocks (the base view CU and its four directly adjacent CUs corresponding to the current CU in the dependent view) are coded in Merge modes and if the R-D performance of Skip mode is better than Merge mode for the current CU. The second shortcut stops the CU splitting process if the CU splitting depth of the current CU is equal to or larger than the maximum depth of the CU splitting depths of the five inter-view neighboring blocks and if Skip mode is selected as the best prediction mode for the current CU after checking all possible prediction modes. These two shortcuts tend to make the dependent texture views less partitionned, allowing to apply more frequently the limitation in depth coding, and consequently increasing the encoder runtime savings.

5.6 Conclusion

In this chapter, we have presented the texture quadtree initialization (QTI) and the depth quadtree limitation (QTL) coding tools and their associated predictive coding (PC) part. QTI+PC and QTL+PC are able to reduce encoder runtime and achieve coding gains by exploiting texture-depth correlations.

QTI+PC brings 6% encoder runtime reduction. The method is promising, although the assumption failures, essentially due to wrong edges in depth, cause coding losses. In a more flexible scheme, the runtime reductions are maintained, and the coding losses are almost entirely compensated by the bitrate reductions the predictive coding part brings. In the future, a more flexible scheme that achieves a better compromise between level of texture quadtree alteration and bitrate reduction potential needs to be found.

QTL+PC, on the other hand, reduces encoder runtime significantly by 31% while achieving -0.3% coding gain on average for coded and synthesized views, -23.1% on depths, and -1.8% on coded videos when the depth bitrate is considered. It can also smooth out wrong edges in depth, which eventually leads to a better synthesis quality. QTL+PC was presented in the 2$^{\text{nd}}$ JCT-3V meeting and was adopted in both the 3D-HEVC working draft and software [JM12a]. Furthermore, a journal article on QTI+PC and QTL+PC was published in the IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) [MJCPP13b].

Chapter 6

3D video coding using warped optical flows

Contents

In computer vision, optical flow is a technique that allows to obtain a dense motion vector field that maps the motion of each pixel from one frame to another. This dense motion estimation technique differs from block-based techniques, currently used in most video codecs, which determine a single vector for all pixels in a block. While the dense motion vector fields obtained using optical flow give more accurate block predictions, transmitting a motion vector for each pixel is costly and it can quickly compensate all potential gains of such a technique. The use of optical flows in video compression has thus always been very limited, its main applications being in computer vision.

In this chapter, we propose a method in 3D-HEVC that employs dense motion estimation using optical flows to improve temporal predictions in dependent views, without the need of signaling the resulting dense motion vector field. This is done using a smart decoder, which also performs the dense motion estimation on compressed frames in the base view.

In a first section of this chapter, we present a state-of-the-art on the use of optical flow in video compression, and on smart decoder coding approaches. Our method is motivated in a second section, then presented and detailed, along with two other variants, in a third section. The experimental results associated with our method and its variants are presented and interpreted in a fourth section, followed by a conclusion where we underline possibilites for future work.

6.1 State-of-the-art

6.1.1 Optical flow

Optical flow (OF) is a method for deriving a Dense Motion Vector Field (DMVF) for a given frame or parts of a frame, where each pixel is associated with a motion vector. Many OF algorithms have been developed over the past thirty years, prominently the Horn and Schunck [HS81] and Lucas and Kanade [LK81] methods which are both based on the spatio-temporal derivatives in a video sequence.

OF is mostly used in video indexing [ALC96, CHWZ08], object detection / recognition / tracking [DFS09, Zha10, SLP13], and feature extraction in video sequences [RSP10, HYK03]. In video compression, some dense motion estimation techniques similar to OF have been used in Distributed Video Coding (DVC) [CMPP09, CMMPP09, MMCPP09] but only because in DVC, the motion estimation is performed at the decoder, and hence there is no need to transmit the vectors. In classical video coding, OF has rarely been a serious contender against block-based motion estimation, since the signaling overhead associated with coding the motion vectors of each pixel is too large.

Some techniques attempt to reduce this overhead. In [LSZ97], a Discrete Cosine Transform (DCT) is applied to the motion vectors of the OF field to exploit redundancies and increase coding efficiency. However, the experimental protocol is questionable and the results are not particularly convincing. In [KMW95], a Hierarchical Finite Elements (HFE) representation is used to code the motion vectors of the OF field. It is reported that the method brings 0.6 dB over block matching algorithms (BMA), but it is unclear in which standard this method was implemented in. Also, the method is fairly old, and it is questionable if this representation might still work in recent video standards such as HEVC which already provides significant

coding gains over older standards. In [KCCL96], the motion vectors of the OF field are transformed using Arbitrary Shaped Transforms (ASTs). However, the extra overhead resulting from signaling which shape was used to transform which block of vectors is also costly. Finally in [CTEC07], the OF is used in a way to derive only one vector per macroblock instead of a DMVF. The technique comes close to the coding performance of BMA but with a reported complexity saving. It obviously fails however to exploit the full potential of OF.

6.1.2 Smart decoder approaches

In classical video coding schemes, the encoder performs an exhaustive search over all coding parameters (prediction mode, partitionning, motion vectors) and transmits the best (RD-wise) coding choices in the bitstream. The decoder simply parses sequentially these coding parameters and reconstructs the video. All the computational complexity is thus concentrated at the encoder while the normalized decoder is light in terms of computations.

These classical video coding schemes maximize the quality of the prediction by exploiting spatio-temporal redundancies through diverse coding choices. This intense competition bounds the global coding performance through the induced signaling costs and the ever-decreasing coding gain margins.

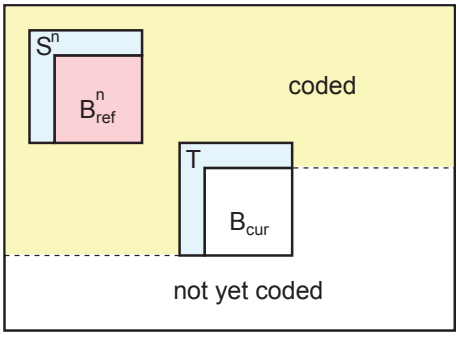
In this context, transferring parts of the operations to the decoder can avoid coding specific information while maintaining the accuracy of the predictions. This complexity switch is not new, it has already been proposed in the Distributed Video Coding (DVC) paradigm [GARRM05]. DVC moves the entire computational complexity from the encoder to the decoder. This is useful for recording systems with low computational power (like cellphones or surveillance cameras) that require coding a video with a limited complexity while guaranteeing a satisfactory video quality after decoding. More conventional coding schemes where only some operations are moved to the decoder, hence resulting in a more equitable load at both sides, were developped as well. The decoder in these cases becomes referred to as a "smart decoder".

Examples of smart decoder approaches include template matching in the Intra and Inter coding modes. In Intra, the basic idea, proposed in [TBS06], defines a shape composed of causal pixels neighboring the current block ($\mathrm{B_{cur}}$) as a target T, as shown in Figure 6.1(a). An exhaustive search accross a number of N blocks, is then performed in a similar way at the encoder and at the decoder, to find a causal block ($\mathrm{B^n_{ref}}$), whose neighboring pixels (S^n) minimize a matching criterion like the

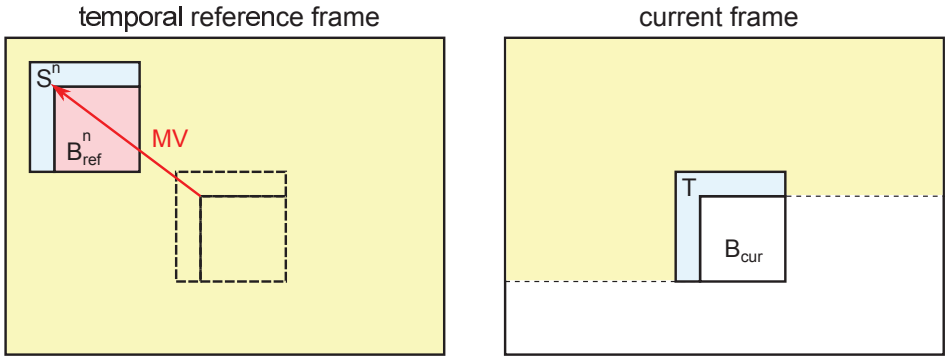
pixel-wise SSD with T:

$$n = \arg \min_{\text{i=0..N-1}} \left\{ SSD(T, S^i) \right\} \tag{6.1}$$

B^n_{ref} will then serve as a predictor for B_{cur}. However, even though they have similar neighboring pixel values, B_{ref} might not be the most efficient predictor for B_{cur}. Hence, in [MADB10], it is proposed to compute the RD cost associated with each predictor in a subset of N' ($N' < N$) best potential predictors (who best minimize the SSD between T and S^i), and signal the index of the one associated with the lowest cost in the bitstream. This reportedly brings 4% bitrate reduction in an all-Intra coding configuration. In [CGT+12], template matching and block matching in Intra replace two of the eight Intra prediction modes in H.264/AVC. This brings -24.8% and -16.7% gain on low and high bitrates respectively.



(a) Intra



(b) Inter

Figure 6.1: Template matching in Intra and Inter coding modes

In Inter, the template matching process is the same, except that B^n_{ref} is searched for in an already coded temporal reference frame, as shown in Figure 6.1(b). The advantage of the template matching approach is that it allows testing a large number of predictors at no additional coding cost. Indeed, since the decoder can perform the same template matching process, nothing needs to be signaled in the bitstream.

Other smart decoder approaches involve reducing the number of predictors in

competition. In [LJPP10], the number of Intra modes is reduced by analyzing the corresponding predictor signals in the transform domain and eliminating the similar patterns until all remaining patterns are complementary. This reduction is performed similarly at the encoder and decoder sides, and it allows coding the selected Intra mode for the current block with less bits. This method is a generalization of [KHL08], where it is adaptively decided to use either a single Intra mode (DC) or the whole set of Intra modes of H.264/AVC according to a criterion which can be evaluated similarly at the encoder and at the decoder. This criterion measures the similarity of the neighboring causal pixels of the current block. The rationale behind this is that if the values of these pixels are similar, there is no need to put all the Intra modes in competition because the result will be equivalent to using the DC predictor anyway.

Furthermore, a smart decoder can also be used in MVD systems. In [SKSM11], it is proposed to synthesize at the level of the currently coded view a virtual view by extrapolation from another view and its associated depth video. All the signaling information (modes, partitions, motion vectors) are determined for this virtual view at both the encoder and decoder sides. Then, it is proposed to use this information to code the current view, without the need of signaling it in the bitstream. This method, put in competition with classical RD mode selection, brings 10% gain over an MVC anchor.

6.2 Motivation

The DMVF obtained using OF gives more accurate frame predictions than the coarse block-based MVF. To verify this assumption, we perform some tests on a subset of the JCT-3V test sequences: Kendo, Balloons, Newspaper and PoznanHall2. Only the left texture video of each sequence is considered for these tests. Frames 0 and 2 (F_0 and F_2) of each video are coded with respect to CTCs [RMV13] at four QPs, using HTM-5.1 to which the following encoder changes are made: first, the lambda (λ) parameter used in Lagrangian cost computations is set to 0, meaning that the encoder minimizes the distortion, regardless of the resulting bitrate. Second, the Intra mode is removed for P & B-slices: no CUs in F_2 are coded in Intra (all PUs have a motion vector). At the end of the encoding stage, we output the resulting coarse motion vector field (C) which, alongside the reconstructed $\widetilde{F_0}$, allows us to construct offline the prediction of F_2 using motion compensation and obtain $\widehat{F}_2^C$. We compute the residual R_2^C as such: $R_2^C = F_2 - \widehat{F}_2^C$. The energy of R_2^C is then computed as such:

$$E_2^C = \frac{1}{MN} \sum_i \sum_j \left(R_2^C[i][j] - \bar{R}_2^C \right)^2 \tag{6.2}$$

where M and N are the width and height of the frame, and $\bar{R}_2^C$ the mean of R_2^C:

$$\bar{R}_2^C = \frac{1}{MN} \sum_i \sum_j R_2^C[i][j] \tag{6.3}$$

In parallel, we also input F_2 and $\widetilde{F_0}$ into an optical flow algorithm (we use the one which can be found in [Liu]). The resulting DMVF (D) is used to motion compensate $\widetilde{F_0}$ hence obtaining another prediction of F_2: $\widehat{F}_2^D$. The corresponding residual R_2^D and its energy E_2^D are then computed. The same process can be repeated a third time if the reconstructed $\widetilde{F_2}$ and $\widetilde{F_0}$ are inputted into the OF algorithm instead, hence yielding another prediction and residual energy: E_2^{DQ}. Table 6.1 gives the values of E_2^C, E_2^D, and E_2^{DQ} for the four considered sequences and QPs.

Sequence	**QP**	$\mathbf{E_2^C}$	$\mathbf{E_2^D}$	$\mathbf{E_2^{DQ}}$
Balloons	25	5.7	4.1	4.2
	30	6.3	4.8	5.0
	35	8.1	6.3	6.9
	40	11.8	8.9	10.3
Kendo	25	28.1	8.5	8.8
	30	15.8	8.7	9.3
	35	16.6	9.3	10.2
	40	17.9	10.0	12.1
Newspaper	25	8.6	5.8	6.0
	30	9.5	7.6	8.0
	35	12.8	10.5	11.4
	40	18.0	15.0	16.8
PoznanHall2	25	9.3	6.3	6.4
	30	8.2	5.9	6.2
	35	7.3	6.3	6.8
	40	8.3	7.2	7.9

Table 6.1: Residual energy values per motion estimation method for the considered sequences and QPs

We can see from Table 6.1 that $E_2^D < E_2^C$ for all tested sequences and QPs: the DMVF computed between F_2 and $\widetilde{F_0}$ using the OF gives a more accurate prediction than the most accurate coarse MVF that the HTM encoder could provide. However, the DMVF has to be sent in the bitstream because the decoder does not have access to F_2 to redo the motion estimation. The gains obtained by lowering the residual energy will thus be countered by the losses resulting from the extra signaling of one motion vector per pixel.

If the DMVF is computed between two already coded frames ($\widetilde{F_2}$ and $\widetilde{F_0}$ for instance), no MV signaling is necessary because the motion estimation can be performed at the decoder side. Although the resulting DMVF will suffer from the

quantization noise of both frames, our experiments show that it is still more accurate than the HTM's coarse MVF as $E_2^{DQ} < E_2^C$ for all considered sequences and QPs. The proposed method described in the next section exploits this conclusion.

6.3 Proposed method

6.3.1 Algorithm description

In 3D-HEVC, the base view in any access unit is coded first, followed by the dependent views. Interview Motion Vector Prediction (IVMP) allows to use the motion vectors of the base view to increase the coding efficiency of the dependent views. Since the base view is coded with HEVC, the motion vectors are computed at PU-level using BMA. We propose to extend IVMP such that a DMVF at the level of the base view is computed using OF, at no additional cost since a smart decoder approach is used. A PU in the dependent view can then inherit a motion vector for each of its pixels from its corresponding PU in the base view, hence increasing the accuracy of the prediction and reducing the residual energy.

Let I_t^B and $\widetilde{I}_t^B$ respectively be the original and reconstructed base view frame at the current time instant t, and $\widetilde{I}_{t-1}^B$ and $\widetilde{I}_{t+1}^B$ the first frames in the L0 and L1 reference lists of I_t^B, respectively. The notations $t-1$ and $t+1$ thus do not necessarely imply that $\widetilde{I}_{t-1}^B$ and $\widetilde{I}_{t+1}^B$ are located in the adjacent past and future access units. Let I_t^D be the current frame to be coded in the dependent view, and $\widetilde{I}_{t-1}^D$ and $\widetilde{I}_{t+1}^D$ the reconstructed dependent view frames in the same access units as $\widetilde{I}_{t-1}^B$ and $\widetilde{I}_{t+1}^B$ respectively.

In the proposed method, after I_t^B is coded, two DMVFs are computed using OF between $\widetilde{I}_t^B$ and $\widetilde{I}_{t-1}^B$, and between $\widetilde{I}_t^B$ and $\widetilde{I}_{t+1}^B$. Two motion vectors for each pixel of $\widetilde{I}_t^B$ are derived this way. The decoder has access to these frames, and can thus perform this motion estimation. The two resulting DMVFs do not need to be transmitted. When coding a PU ($\mathrm{B_{cur}}$) in I_t^D, the NBDV process derives a DV ($\mathrm{DV_{IVMP}}$) for this PU, pointing to a corresponding PU ($\mathrm{B_{ref}}$) in $\widetilde{I}_t^B$. $\mathrm{B_{ref}}$ will thus have two motion vectors $\mathrm{MV_{ref}^0}$ and $\mathrm{MV_{ref}^1}$ in the L0 and L1 reference list (respectively), estimated at the base view encoding pass, and two DMVFs: $\mathrm{OF_{ref}^0}$ and $\mathrm{OF_{ref}^1}$. IVMP already inherits $\mathrm{MV_{ref}^0}$ and $\mathrm{MV_{ref}^1}$ into a single multi-view candidate in the Merge candidate list of $\mathrm{PU_{cur}}$. We propose to additionally inherit $\mathrm{OF_{ref}^0}$ and $\mathrm{OF_{ref}^1}$ (which we refer to as $\mathrm{WOF_{cur}^0}$ and $\mathrm{WOF_{cur}^1}$ at the level of the dependent view) as another **dense** Merge candidate for $\mathrm{PU_{cur}}$ located at the first position of the list. Figure 6.2 illustrates our method.

The warped optical flow DMVFs: $\mathrm{WOF_{cur}^0}$ and $\mathrm{WOF_{cur}^1}$ are only accessible by $\mathrm{PU_{cur}}$ at the motion compensation stage. If $\mathrm{PU_{cur}}$ is coded in Merge mode using the

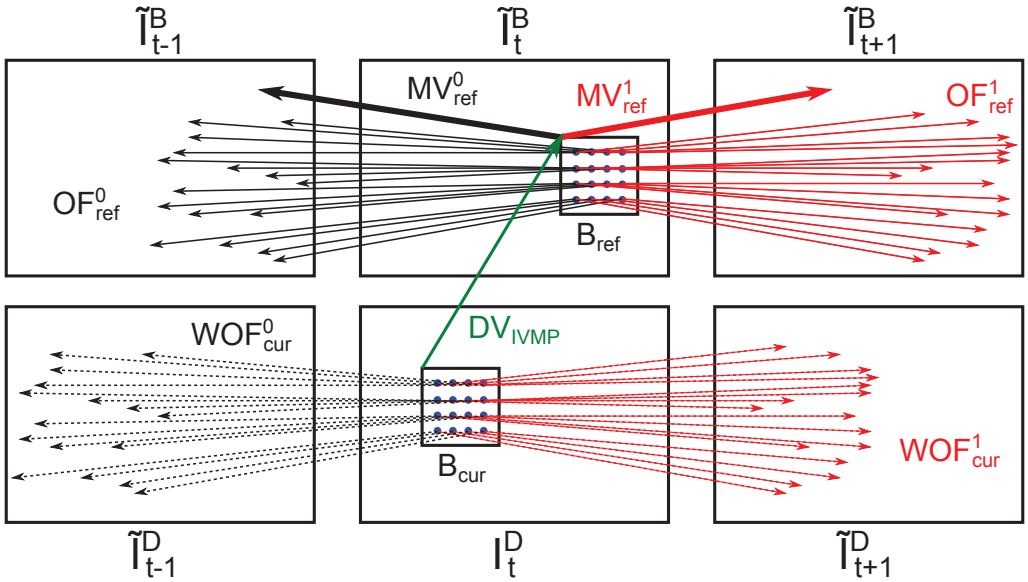


Figure 6.2: Proposed method for deriving dense Merge candidate

dense Merge candidate, other neighboring PUs regard it as being coded with MV^0_{ref} and MV^1_{ref}. These vectors actually will correspond to the left Merge and AMVP candidates for the right neighboring PU for instance.

Note that the method is only applied if I^B_t is not an I-frame (otherwise its reference lists would be empty). If I^B_t is a P-frame, I^B_{t+1} is not available, and hence only one DMVF is computed between $\widetilde{I}^B_t$ and $\widetilde{I}^B_{t-1}$, then warped and inherited by the current PU in the dependent view. Uni-predictive motion compensation with only WOF^0_{cur} is performed in this case.

6.3.2 Variants

In the proposed method, a single DV allows to disparity-compensate the current PU to find the corresponding PU in the base view. However, PU_{cur} may contain objects of different depths (like for instance parts of a foreground object and background), some of which might inherit incorrect motion from the base view due to an incorrect disparity shift. To solve this issue, we propose two variants of the original method.

FCO-WOF

The first variant can only be applied if the depth maps of the dependent views are coded before the texture views. This can be done using the non-normative Flexible Coding Order (FCO) tool. Let $\widetilde{Z}^D_t$ be the reconstructed dependent view depth map at the current time instant. When coding PU_{cur} in I^D_t, we propose in this variant to disparity compensate each pixel with its own DV. The DV of a pixel can be obtained

by converting the collocated depth value z in $\widetilde{Z}_t^D$ into a disparity value d using the following equation:

$$d = f \cdot t_x \cdot \left[\frac{z}{255} \left(\frac{1}{Z_{min}} - \frac{1}{Z_{max}} \right) + \frac{1}{Z_{max}} \right] \tag{6.4}$$

where f is the focal length, t_x the baseline between the base view and the dependent view, and Z_{min} / Z_{max} the extremal depth values. Since we are assuming a 1D parallel camera setup, the disparity is only horizontal: $DV = (d, 0)$.

Thus, in this variant, $\text{WOF}^0_{\text{cur}}$ and $\text{WOF}^1_{\text{cur}}$ will be composed of the motion vectors of the pixels in $\widetilde{I}_t^B$ pointed to by the different computed disparity vectors of PU_{cur}, as shown in Figure 6.3. PU_{cur} is still seen by other PUs as coded with MV^0_{ref} and MV^1_{ref} if the dense Merge candidate is selected, which means DV_{IVMP} is still used to fetch B_{ref} to get these two vectors.

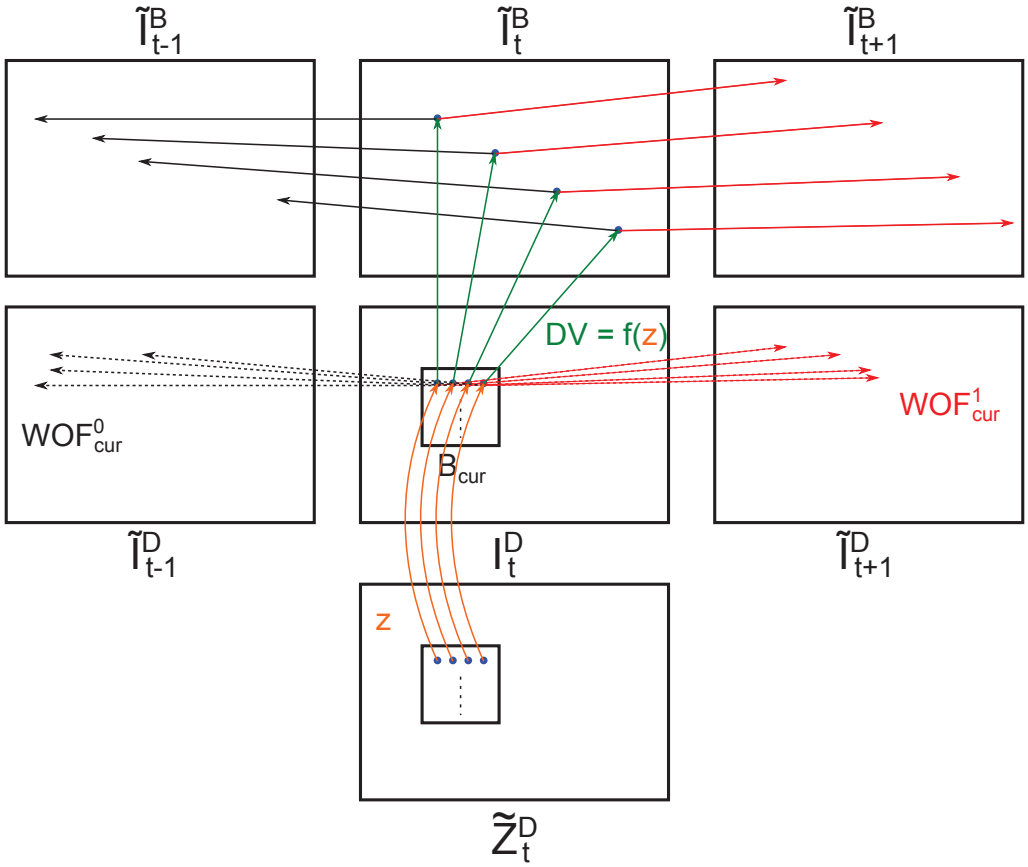


Figure 6.3: Proposed FCO-WOF variant for deriving dense Merge candidate

DO-WOF

If the standard texture-before-depth coding order is used, FCO-WOF cannot be applied because the decoder would not have access to $\widetilde{Z}_t^D$ to decode I_t^D. However, the decoder would have access to the reconstructed base view depth which can also be used to derive a dense disparity vector field to disparity-compensate each pixel in $\mathrm{PU}_{\mathrm{cur}}$.

Let $\widetilde{Z}_t^B$ the reconstructed base view depth. In this variant called Depth Oriented-WOF (DO-WOF), $\mathrm{DV}_{\mathrm{IVMP}}$, derived using NBDV, is used to fetch a depth block $\mathrm{Z}_{\mathrm{ref}}$ corresponding to $\mathrm{PU}_{\mathrm{cur}}$ in $\widetilde{Z}_t^B$. The depth value z of each pixel in $\mathrm{Z}_{\mathrm{ref}}$ is transformed into a disparity d using Equation 6.4, which is then used to disparity-compensate the corresponding pixel in $\mathrm{PU}_{\mathrm{cur}}$. The motion vectors of the pixels in $\widetilde{I}_t^B$ pointed to by the different computed disparity vectors of $\mathrm{PU}_{\mathrm{cur}}$ form $\mathrm{WOF}_{\mathrm{cur}}^0$ and $\mathrm{WOF}_{\mathrm{cur}}^1$, as shown in Figure 6.4.

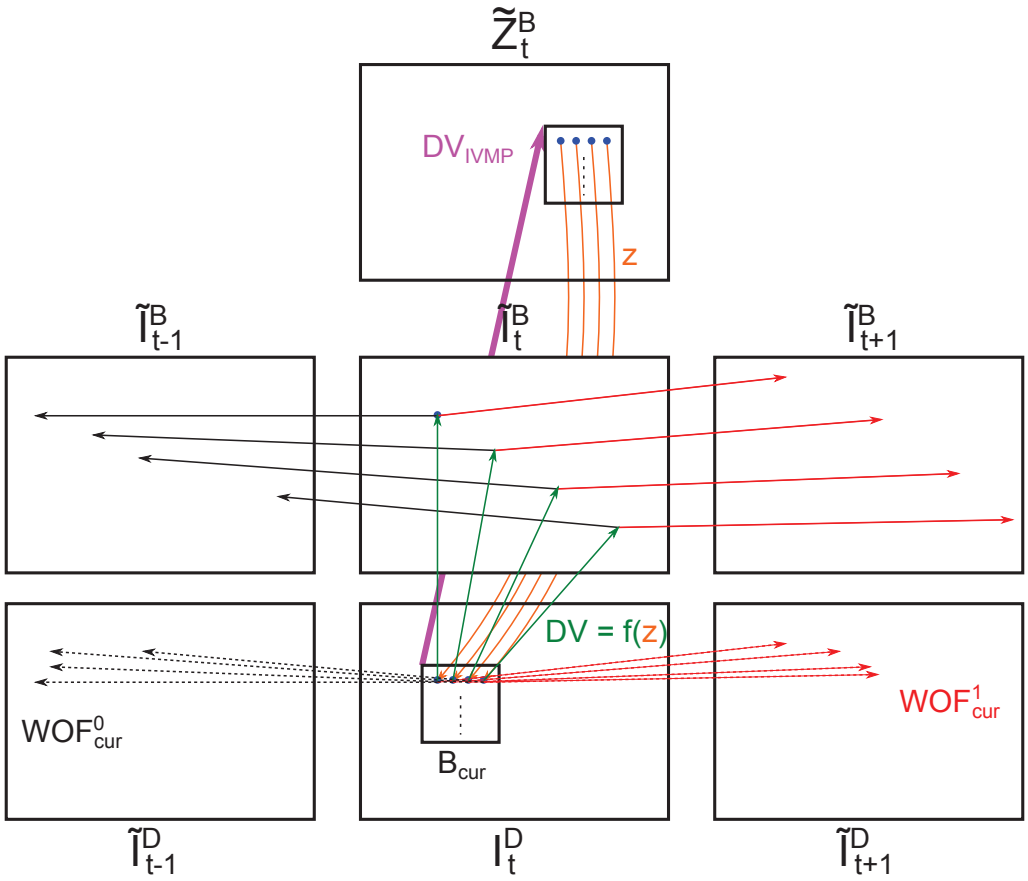


Figure 6.4: Proposed DO-WOF variant for deriving dense Merge candidate

6.3.3 Advantages and disadvantages of the proposed method

The proposed method and variants allow to motion-compensate with a DMVF computed using OF at no additional cost. The resulting prediction is more accurate, and the energy of the corresponding residual is reduced, hence bringing coding gains. Furthermore, in 3D-HEVC, CUs are often split in a quad-tree fashion to compensate for a lack of a good CU predictor signal, at the cost of an increased signaling: the MVF of an Inter-coded CU becomes denser as the number of transmitted split flags increases. The densest MVF associates a motion vector to each 4×4 PU of a 64×64 CU but requires 21 split flags to be sent, while the coarsest associates a single motion vector to the entire CU with only one split flag. In our method, the MVF of a CU is already dense without the need to split that CU since each pixel is associated with a motion vector. Hence, our method also allows savings on split flags.

The disadvantage of the proposed method is that unlike classical motion estimation which considers the original uncompressed block and finds the corresponding block in a compressed reference, the proposed dense motion estimation is performed between two compressed frames. The resulting DMVF is thus affected by the quantization noise of the base view frame and might not always be accurate, especially at low bitrates. However, as shown in Section 6.2, the method can still outperform classical motion estimation. Furthermore, there is a considerable amount of complexity that is added to the decoder, since it has to re-do the complex optical flow computation, but this is acceptable since this smart decoder approach allows significant savings on MV signaling.

6.4 Experimental results

6.4.1 Experimental setting

We implement the proposed method (referred to in the rest of this chapter as WOF) and its variants (DO-WOF & FCO-WOF) in HTM-5.1. The OF code is integrated into HTM-5.1 from the package downloadable from [Liu]. The OF configuration parameters are summarized in Table 6.2. The downsampling ratio parameter (R) is particularly interesting. Basically, the core OF algorithm employs a Gaussian pyramid, composed of N levels, N being driven by R (as R increases, N increases). At each level i, the two images between which a DMVF is to be computed are downsampled from their resolution at level $i-1$, level 0 corresponding to the full resolution images. At level $N-1$, a DMVF is computed between the smallest resolution images. Then, it is upsampled at level $N-2$ and updated by performing another dense motion estimation. This is repeated until a dense full resolution MVF

is determined for level 0. As N increases, the final DMVF becomes more accurate at the cost of additional computations. More details on the core OF algorithm, and on the significance of the other parameters in Table 6.2 can be found in [Liu09].

The method and the variants are tested on the 7 sequences of the JCT-3V test set and the coding configuration defined in the CTCs [RMV13] is respected with the exception of the length of the sequences to code which is limited to 5 seconds instead of 10 in order to speed-up the simulations.

Gains in the following tables are measured using the Bjontegaard delta-rate (BD-Rate) metric. For more information on the significance of the different columns, refer to Section 4.2. For WOF & DO-WOF, the reference anchor is HTM-5.1. The FCO-WOF variant requires changing the coding order so that dependent depth videos are coded before the dependent texture views. This is done by enabling FCO in the encoder configuration file. This variant is compared to a HTM-5.1 reference where FCO is enabled as well.

Parameter	**Description**	**Value**
Alpha	Regularization weight	0.012
Ratio	Downsampling ratio	0.4
MinWith	Width of the coarsest level	20
nOuterFPIterations	Number of outer fixed point iterations	7
nInnerFPIterations	Number of inner fixed point iterations	1
nSORIterations	Number of Successive Over Relaxation iterations	30

Table 6.2: Optical flow parameters

6.4.2 Coding results

Tables 6.3, 6.4 and 6.5 present the coding results of WOF, DO-WOF, and FCO-WOF respectively. Negative values in these tables indicate gains. WOF brings -7.3% and -6.9% gain on the dependent texture views, and -2.5% gain on synthesized views. DO-WOF is slightly better on the right view (Video 2) with -7.2% gain and slightly worse on the left view (Video 1) with -7.1% gain. In the "Video Total" results, DO-WOF is slightly better than WOF by -0.1%. FCO-WOF is better than both WOF and DO-WOF with -7.5% and -7.6% on dependent texture views, and -2.7% on synthesized views. Note that in these three tables, no gains are reported for "Video 0" because neither WOF, DO-WOF or FCO-WOF are applied for base view coding.

For WOF, we perform some additional shorter simulations with a lower number of frames (1 second of video), where we consider two other sets of QPs, corresponding to a low bitrate and a high bitrate test. The first set (low bitrate) is composed of the following texture / depth QPs: (30;39), (35;42), (40;45) and (45;48) while the

Sequence	Video				Video total	Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	-7.8	-7.0	-3.2	-3.1	-3.1	127	10514
Kendo	0.0	-5.4	-4.5	-1.9	-1.8	-2.0	126	11417
Newspaper	0.0	-3.1	-2.6	-1.1	-1.0	-1.0	141	12968
GT Fly	0.0	-21.1	-20.4	-6.0	-5.7	-5.9	135	11160
PoznanHall2	0.0	-7.9	-9.4	-3.8	-3.7	-3.7	124	11515
PoznanStreet	0.0	-3.6	-4.2	-1.1	-1.1	-1.4	142	12159
Dancer	0.0	-1.9	-0.4	-0.5	-0.4	-0.5	136	11373
Average	**0.0**	**-7.3**	**-6.9**	**-2.5**	**-2.4**	**-2.5**	**133**	**11587**

Table 6.3: BD-Rate coding results of the proposed method

Sequence	Video				Video total	Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	-8.2	-7.1	-3.2	-3.2	-3.1	130	11313
Kendo	0.0	-4.9	-3.9	-1.7	-1.6	-1.8	136	12813
Newspaper	0.0	-2.5	-2.3	-0.9	-0.9	-0.8	140	12523
GT Fly	0.0	-21.7	-22.0	-6.3	-6.0	-6.1	135	11025
PoznanHall2	0.0	-8.3	-10.5	-4.1	-4.0	-4.0	127	11473
PoznanStreet	0.0	-3.2	-4.1	-1.0	-1.0	-1.3	139	12763
Dancer	0.0	-1.2	-0.7	-0.5	-0.4	-0.5	144	11868
Average	**0.0**	**-7.1**	**-7.2**	**-2.5**	**-2.5**	**-2.5**	**136**	**11968**

Table 6.4: BD-Rate coding results of the DO-WOF variant

other (high bitrate) is composed of: (20;31), (25;34), (30;39) and (35;42). The set of QPs used in CTC corresponds to the middle bitrate test. For each test, the same set of QPs is used for the reference and for the proposed method. Table 6.6 presents the coding results, averaged accross all the sequences, per test. We can see that the lower the bitrate, the more our method becomes efficient. This might be surprising because on higher bitrates, the DMVF computed at the level of the base view is supposed to be more accurate as it is less affected by quantization noise, and consequently, one would expect the gains to increase as the bitrate increases. There is however a justification for this peculiar phenomenon, which is given in the next section.

We perform one final test where we gradually change the downsampling ratio (R) in the OF parameters in order to study its impact on the coding results and complexity of the proposed method. Short simulations on 1 second of video are also performed for this. Table 6.7 presents the average coding results and runtimes obtained for various values of R. We can see that as R decreases, the coding results are only slightly impacted while the encoder and decoder runtimes are significantly reduced.

Sequence	Video				Video total	Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
Balloons	0.0	-9.0	-9.0	-3.7	-3.3	-3.6	138	11175
Kendo	0.0	-5.3	-4.2	-1.8	-1.5	-2.2	134	11423
Newspaper	0.0	-2.5	-2.4	-0.9	-0.8	-1.0	134	12153
GT Fly	0.0	-22.1	-22.1	-6.3	-5.9	-6.4	137	10768
PoznanHall2	0.0	-8.7	-11.2	-4.3	-3.7	-3.9	148	11804
PoznanStreet	0.0	-3.6	-4.3	-1.1	-1.0	-1.4	137	7998
Dancer	0.0	-1.4	-0.3	-0.5	-0.4	-0.7	143	11834
Average	**0.0**	**-7.5**	**-7.6**	**-2.7**	**-2.4**	**-2.7**	**139**	**11022**

Table 6.5: BD-Rate coding results of the FCO-WOF variant

Test	Video				Video total	Synt.
	0	1	2	Avg		
Low bitrate	0.0	-5.9	-5.8	-2.0	-2.0	-2.0
Middle bitrate	0.0	-4.8	-4.4	-1.5	-1.5	-1.6
High bitrate	0.0	-2.7	-2.3	-0.8	-0.8	-0.8

Table 6.6: BD-Rate coding results of WOF for 1 second of video at low, middle, and high bitrates

6.4.3 Results interpretation

Origin of the gains

The significant gains that our method and its variants bring, as shown in Tables 6.3, 6.4 and 6.5, can be explained by analyzing three types of data: the selection percentage of the dense Merge candidate (SP = the number of PUs coded in Merge mode with the dense Merge candidate (DMC) over the total number of coded PUs), and the reduction in the number of split (RSP) and skip (RSK) flags sent in the bitstream compared to an HTM-5.1 anchor. Table 6.8 shows these three values, averaged across the four considered QPs, for each sequence and for each variant. Basically, the more the DMC is selected, the more it reduces the residual energy and brings coding gains. Additional gains come from the reduction in the number of transmit-

Ratio	Video				Video total	Synt.	Runtimes	
	0	1	2	Avg			Enc	Dec
$R = 0.85$	0.0	-4.7	-4.3	-1.5	-1.5	-1.6	275	52355
$R = 0.75$	0.0	-4.8	-4.4	-1.5	-1.5	-1.6	187	29142
$R = 0.65$	0.0	-5.0	-4.6	-1.6	-1.6	-1.5	175	22294
$R = 0.5$	0.0	-4.9	-4.6	-1.6	-1.6	-1.5	138	12023
$R = 0.4$	0.0	-4.9	-4.5	-1.6	-1.5	-1.5	134	10136

Table 6.7: Average BD-Rate coding results and runtimes of WOF with various values of R for 1 second of video

ted split and skip flags. The most significant bitrate reductions appear in the GT Fly sequence because both gain sources are present: the DMC is frequently selected ($SP = 69\%$) on the one hand, and the split and skip flag reductions are important ($RSP = 66\%$ and $RSK = 76\%$) on the other. In the Balloons sequence, the DMC is frequently selected ($SP = 76\%$) but the split and skip flag reductions are lower than in GT Fly ($RSP = 40\%$ and $RSK = 47\%$) hence the lower but still significant overall coding gains. In Dancer, the DMC is less selected ($SP = 42\%$) and the split and skip flag reductions are low ($RSP = 23\%$ and $RSK = 28\%$). This is why the smallest gains are reported for this sequence. The relatively high coding gain devi-

Sequence	**WOF**			**DO-WOF**			**FCO-WOF**		
	SP	**RSP**	**RSK**	**SP**	**RSP**	**RSK**	**SP**	**RSP**	**RSK**
Balloons	76	40	47	73	43	49	74	44	50
Kendo	60	28	32	58	29	33	58	29	33
Newspaper	67	29	33	65	29	33	65	29	33
GT Fly	69	66	76	68	68	77	68	68	77
PoznanHall2	78	40	42	78	41	43	79	41	42
PoznanStreet	64	39	48	62	40	48	63	40	48
Dancer	42	23	28	39	24	28	38	24	28
Average	**65**	**38**	**44**	**63**	**39**	**45**	**64**	**39**	**45**

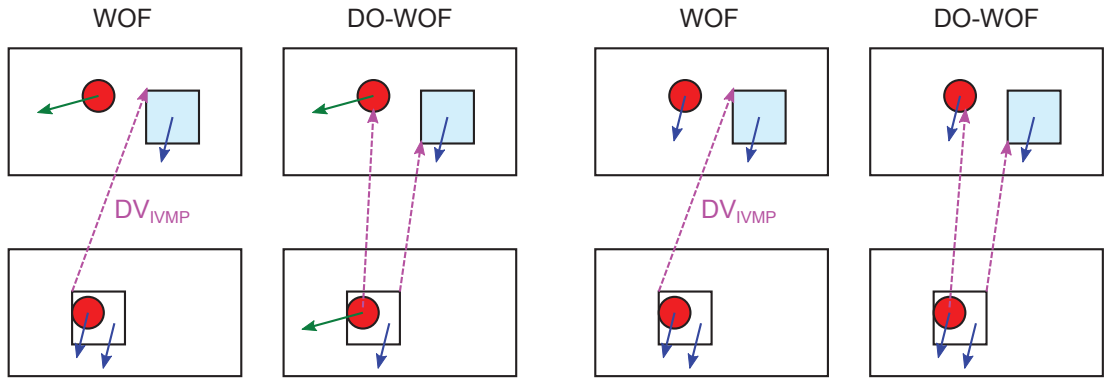
Table 6.8: Selection percentage of the dense Merge candidate and reduction in split and skip flags in the proposed method and variants

ation across the various sequences can be explained by the nature of the motion in each sequence. The Dancer sequence contains relatively complex motion including camera motion and frequent rotations of the dancer character. The OF algorithm fails in these conditions to detect the true motion of objects and gives inaccurate motion vectors. One solution is to tweak the OF parameters in order to increase the accuracy of the resulting DMVF and increase coding gains, although this implies additional computational cost. An adaptive setting of the OF parameters according to the motion in each sequence (or in each frame of a sequence) can be a more interesting alternative, but this solution is not trivial as it requires conceiving a way to detect and characterize the motion before the OF computation.

Performance comparison between WOF, DO-WOF and FCO-WOF

Furthermore, we can see from Tables 6.3 and 6.4 that the results of DO-WOF and WOF are similar. A more detailed analysis shows that DO-WOF is better than WOF in sequences where we have the highest gains. In the "Video Total" column, DO-WOF brings an additional -0.1%, -0.3% and -0.3% over the proposed method for Balloons, GT Fly and PoznanHall2, which already give the best gains. The gains

are particularly good on these sequences because the DMVF computed using the OF is particularly accurate, and this, in turn, allows for a better assignment of motion vectors to pixels in the dependent view when a dense disparity compensation is used, as in DO-WOF. Figure 6.5 shows the difference between WOF and DO-WOF if the DMVF is accurate, and if it is not. Let us assume two objects (a square and a circle) have two different motions. If the DMVF is accurate, a different motion vector will be associated to each object in the base view. In that case, the WOF method will incorrectly inherit the motion vector of the square for the circle, as opposed to the DO-WOF method, as shown in Figure 6.5(a). If the DMVF is not accurate, the two objects may be incorrectly assigned the same motion vector, making no difference if WOF or DO-WOF is used 6.5(b). As for FCO-WOF, we can see from Table 6.5 that it is better than both WOF and DO-WOF. This was expected, because the disparity compensation in this case is the most accurate.



(a) Accurate dense motion estimation (b) Inaccurate dense motion estimation

Figure 6.5: Difference between WOF and DO-WOF in case of an accurate and an inaccurate dense motion estimation

Coding results analysis of WOF at lower and higher bitrates

Table 6.6 shows that our method becomes more efficient as the bitrate decreases. This is due to the fact that on low bitrates, Merge mode in HTM-5.1 is already selected more often by PUs, as shown in Table 6.9. Indeed, the high quantization at low bitrates makes the temporal reference frames less detailed. In this case, the encoder can achieve the same prediction using a neighboring MV at no cost (Merge mode), or an estimated MV which needs to be signalled (Inter mode), so the choice obviously goes in favor of the Merge mode. In our method, the dense Merge candidate we add has more chances of being selected at low bitrates, simply because the Merge mode itself is in general more often selected. This is confirmed in Table 6.10.

Sequence	Merge mode selection			
	25	**30**	**35**	**40**
Balloons	89	92	95	97
Kendo	85	90	94	96
Newspaper	81	87	91	95
GT Fly	92	94	98	99
PoznanHall2	89	93	98	99
PoznanStreet	88	92	97	98
Dancer	83	87	93	96
Average	**87**	**91**	**95**	**97**

Table 6.9: Selection percentage of the Merge mode in the HTM-5.1 anchor per sequence and per QP

Sequence	SP					
	20	**25**	**30**	**35**	**40**	**45**
Balloons	36	67	82	90	93	96
Kendo	24	36	49	59	70	78
Newspaper	36	63	80	89	96	98
GT Fly	18	33	81	90	93	94
PoznanHall2	18	55	76	85	90	95
PoznanStreet	19	36	51	66	79	89
Dancer	8	16	37	67	81	89
Average	**23**	**44**	**65**	**78**	**86**	**91**

Table 6.10: Selection percentage of the dense Merge candidate per sequence and per QP

Complexity analysis

Finally, we can see from Table 6.3 that our method increases the encoder and decoder runtimes by 33% and 11487% respectively. While the increase can be considered acceptable at the encoder, it remains quite significant at the decoder. It is important to note that the OF code is directly copied from the OF package and integrated into HTM-5.1, and is thus not as optimized as the rest of the HTM-5.1 code. Hence, it is possible that some simplifications to the data structures used could reduce the decoder runtime.

Furthemore, we study in this chapter the impact of increasing the value of the downsampling ratio R on the coding results. We can see from Table 6.7 that increasing R (and therefore N) only slightly impacts the results but dramatically increases the encoder and decoder runtimes. Hence, another solution to reduce the decoder runtime, potentially at similar coding performance impacts, is to tweak other OF parameters (presented in Table 6.2).

Yet another way to achieve such savings is to perform an on-the-fly OF computation at PU level, instead of one at the frame level. The rationale behind this is

that when disparity-compensating PUs coded with the dense Merge candidate at the decoder, not all pixels in the base view are referenced. Table 6.11 shows that only 68% (67% for the variants) of base view pixels are referenced in a dependent view. Hence, the MVs of the remaining pixels do not need to be computed. This can alleviate some computations at the decoder and hence reduce runtime.

Sequence	**Base view pixel usage**		
	WOF	**DO-WOF**	**FCO-WOF**
Balloons	76	75	75
Kendo	68	68	67
Newspaper	70	66	67
GT Fly	69	69	69
PoznanHall2	76	78	78
PoznanStreet	67	66	67
Dancer	48	47	46
Average	**68**	**67**	**67**

Table 6.11: Percentage of referenced base view pixels at the decoder by a dependent view

6.5 Conclusion

We have presented in this chapter a method that allows PUs in dependent views to inherit a DMVF computed at the level of the base view using OF, at no additional signaling cost. Indeed, the DMVF is computed between two reconstructed frames also available at the decoder, hence there is no need to transmit the MVs. The DMVF is added in the first position of the Merge candidate list of dependent view PUs. Correspondences between pixels in the dependent view and in the base view are done using disparity compensation. Different disparity compensation methods exist: a compensation with a single DV for the entire PU using NBDV (WOF), a compensation with one DV per pixel computed from the associated and already coded dependent view depth map (FCO-WOF) and one with also one DV per pixel but computed from the already coded base view depth (DO-WOF). WOF brings -7.3%, -6.9% and -2.5% gain on the left and right dependent views and on synthesized views respectively. DO-WOF is better than WOF mostly on sequences where the computed dense MVF is particularly accurate and which consequently present significant gains. FCO-WOF is better than both methods with -7.5%, -7.6% and -2.7% gain on the two dependent views and synthesized views respectively. These results will be transcribed in a conference paper for the IEEE International Conference on Image Processing (ICIP) 2014.

The study performed in this work allows us to conclude that the coding gains

come from an improvement of the PU predictions which, in turn, reduces the residual energy, and from the reduction of the number of split and skip flags sent in the bitstream. Also, we conclude that the method and the variants are more efficient at low bitrates since the Merge mode, and *a fortiori*, the dense Merge candidate, are more frequently selected as the bitrate decreases. Finally, although the method increases decoder runtime significantly, some savings can be obtained by tweaking the OF parameters, or by implementing an on-the-fly, OF computation at PU-level, instead of computing the DMVF at the frame level.

In the future, a simplification of the OF code, and some general parameter tweaking will be performed to reduce the decoder runtime. A PU-level OF computation at the decoder will also be implemented. Finally, we will correct the inherited DMVFs by exploiting an epipolar constraint using additional depth maps in order to account for occlusions and disocclusions. This allows obtaining a more accurate DMVF and consequently higher coding gains.

Chapter 7

View synthesis exploiting temporal prediction

Contents

In previous chapters, we presented some coding tools designed to improve the coding efficiency in 3D-HEVC. In this chapter, we do not deal with the encoding / decoding process but rather with the synthesis stage after decoding. Traditionally, view synthesis only interpolates intermediate views from side views using the disparity information. Temporal correlation between different frames in the intermediate view can however be exploited to improve the synthesis. In our method[1], we use the optical flow, which already proved to be effective at increasing the coding effiency of

[1]This work has been initiated by Andrei Purica during his internship in Telecom ParisTech [Pur13]

texture data in 3D-HEVC (see Chapter 6), to derive dense motion vector fields at the side views, then warp them at the level of the intermediate views. This allows constructing different temporal predictions of the synthesized frame which would then be merged together. The remaining holes are then inpainted.

The first section of this chapter presents a state-of-the-art of view synthesis techniques. The proposed method is described in the second section. The results obtained are summarized in section three with a detailed interpretation, followed by a conclusion where we underline possibilities for future work.

7.1 State-of-the-art in view synthesis techniques

View synthesis is the process of extrapolating or interpolating a view from other available views. It is a popular research topic in computer vision, and numerous methods have been developped in this field over the past four decades. View synthesis techniques can be mainly classified in three categories [SK00]. The methods in the first category require explicit geometry information, like depth or disparity maps to warp the pixels in the available views to the correct position in the synthesized view. Methods in the second category require only implicit geometry like some pixel correspondances in the available and synthesized view, that can be computed using optical flow for instance. Finally, methods in the third category require no geometry at all. They appropriately filter and interpolate a pre-acquired set of samples (examples of tools in this category include light field rendering [LH96], lumigraph, concentric mosaics, etc.).

The second and third category of methods are out of the scope of this thesis, and thus, will not be further discussed. We focus on the first category of methods, also referred to as Depth Image Based Rendering (DIBR) techniques, in this state-of-the-art. We first discuss the rendering technique used in the reference software for view synthesis, and in the rendering software used in JCT-3V. Then, an overview of some rendering techniques found in literature is presented.

7.1.1 Reference softwares

View Synthesis Reference Software

In 2010, MPEG expressed a significant interest in MVD formats for their ability to support 3D video applications. Several exploration experiments were established around the subject at the 94$^{\text{th}}$ MPEG meeting (this activity corresponded to the second part of FTV, the first being MVC). An experimental framework was developped as well to conduct the different experiments [EXP10]. This framework

defined a View Synthesis Reference Software (VSRS), which would later become an anchor to several new rendering techniques in literature.

VSRS inputs two texture views and their two associated depth views, along with intrinsic and extrinsic camera parameters. The output is a synthesized intermediate view. VSRS allows synthesizing frames using two operational modes: general mode and 1D mode. Hence, both 1D-parallel, where cameras are aligned in a straight line perpendicularly to their optical axes, and non-parallel (*e.g.*cameras aligned in an arc) camera setups are supported.

Figure 7.1 illustrates the rendering process in the general mode of VSRS. First, the left and right reference depth maps (D_L and D_R) are warped to the virtual view position, giving D'_L and D'_R. The 3D image warping process and the warping equations are described in Section 1.2.2. The occlusions are handled by warping the pixel with the highest depth value (closest to the camera). D'_L and D'_R are then median filtered to fill small holes, giving D''_L and D''_R. A binary mask is maintained for each view to memorize larger holes caused by disocclusions. D''_L and D''_R are then used to warp the texture views T_L and T_R to the virtual view position, giving T'_L and T'_R (this reverse warping process wherein the depths are warped first and then used to warp the texture is reported to give a higher rendering quality). Holes in one of the warped views are filled with collocated non-hole pixels from the other warped view, if available. This gives T''_L and T''_R, which are then blended together to form a single representation. The blending can be a weighted average according to the distance of each view to the virtual view point (Blending-On mode), or it can simply consist in taking the closest view to the virtual view point, and discarding the other (Blending-Off mode). The binary masks of each view are merged together at this stage in order to form a single mask which identifies the remaining holes. These are filled at the final stage of the algorithm by propagating the color information inward from the region boundaries. VSRS also includes a Boundary Noise Removal (BNR) option that allows expanding the holes into the background. These areas are then filled in T'_L and T'_R from the other reference view (respectively T'_R and T'_L). This reduces the cases where foreground pixels are falsely projected into background objects due to depth errors.

The 1D mode of VSRS works a bit differently. In this mode, the camera setup is assumed to be 1D parallel. This allows to make a number of simplifications to the warping process which is reduced to a simple horizontal shift (see Equation 1.8). First, the color video is up-sampled for half-pixel or quarter pixel accuracy. A "splatting" option allows a reference pixel to be warped to two sample locations in the target view. Furthermore, the "CleanNoiseOption" and "WarpEnhancementOption" avoid warping unreliable pixels, hence reducing the synthesis artefacts due to the

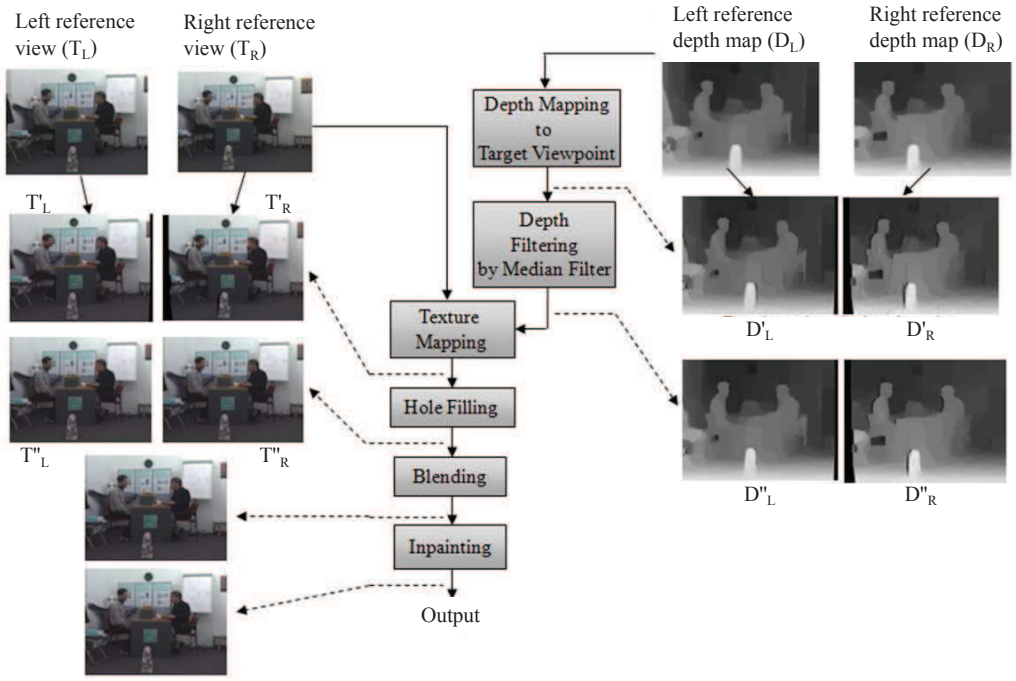


Figure 7.1: Flow diagram for VSRS general mode

texture-depth misalignment at object boundaries and wrongly categorized holes in the foreground. The warping process gives two warped images, two warped depth maps, and two binary masks from the left and right reference views. Each pair is then merged together. When a pixel gets mapped from both the left and the right reference views, a specific blending method, driven by the "MergingOption" parameter, decides the final pixel value. When "MergingOption" equals 0, the blending process relies solely on depth, which means that the pixel closer to camera is selected. When it equals 1, an averaging is performed. Remaining holes are filled by propagating the background pixels into the holes along the horizontal row. Finally, the image is downsampled to its original size.

View Synthesis Reference Software 1D Fast

Each contribution to the 3D-HEVC standardization which proposes to modify the coding of dependent views or depth data, is required to present coding results on synthesized views. The software used for synthesizing the intermediate views is a variant of VSRS, called View Synthesis Reference Software 1D Fast (VSRS-1DFast). This software is included in the HTM package, and is documented in the 3D-HEVC test model [ZTWY13]. VSRS-1DFast allows inputting two or three texture and depth views along with their corresponding camera parameters, and synthesize an arbitrary number of intermediate views. Just like the 1D mode of VSRS, VSRS-1DFast assumes that the camera setup is 1D parallel.

Figure 7.2 illustrates the different steps of the rendering algorithm used in VSRS-1DFast. The texture views $s_{T,l}$ and $s_{T,r}$ are first upsampled to obtain $\widehat{s}_{T,l}$ and $\widehat{s}_{T,r}$: the luma component is upsampled by a factor of four in the horizontal direction, and the chroma by a factor of eight in horizontal direction and two in vertical direction, thus yielding a 4:4:4 representation. The warping, interpolation and hole filling are carried out for $\widehat{s}_{T,l}$ and $\widehat{s}_{T,r}$ line-wise and within a line, interval-wise. This gives two representations of the synthesized frame: $s'_{T,l}$ and $s'_{T,r}$. Then, two reliability maps $s'_{R,l}$ and $s'_{R,r}$ are determined indicating which pixels correspond to disocclusions (reliability of 0). A similarity enhancement stage then adapts the histogram of $s'_{T,l}$ to the one of $s'_{T,r}$. Finally, $s'_{T,l}$ and $s'_{T,r}$ are combined. If the "interpolative rendering" option is activated, the combination would depend on the warped depth maps and the two reliability maps created. If not, the synthesized view is mainly rendered from one view and only the holes are filled from the other view. The resulting combination is later down-sampled to the original size of the texture views.

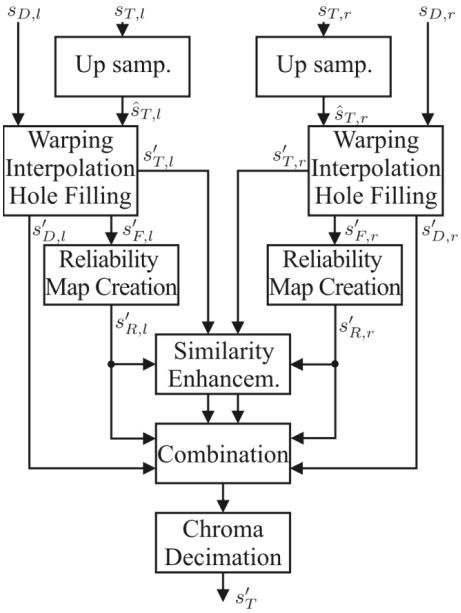


Figure 7.2: Flow diagram for VSRS-1DFast

7.1.2 Rendering techniques in literature

In [FLG13], a rendering technique called View Synthesis using Inverse Mapping (VSIM) is introduced. It operates at full-pel accuracy and assumes a 1D-parallel camera setting. The left and right texture views are warped to the synthesized view position using simple horizontal shifts, also called column shifts (see Equation 1.8). A table is maintained for the left and right interpretations of the synthesized frame which records the column shift of each pixel. Holes in these two tables are filled using

a median filter. Then, the two representations are merged and the remaining holes are filled by checking the collocated value in the tables, and inverse mapping the pixel back to its original value in the left or right view. Residual holes are filled by simply assuming that their depth is the same as the depth of the collocated pixels in the original views. VSIM outperforms VSRS, on average, by 0.412 dB at quarter-pel accuracy and 1.350 dB at full-pel accuracy. However, the rendering runtime is not provided, making it difficult to assess the complexity of the method.

In [LE10], the depth maps are pre-processed with an adaptive smoothing filter in order to reduce holes after synthesis. The filter is only applied to edges in the depth map (corresponding to an abrupt transition in depth values) as those are the main cause for holes. The method is thus less complex than methods which apply a symetric or asymetric smoothing filter to the depth map as a whole. Furthermore, if hole regions correspond to vertical edges, an asymetric Gaussian smoothing filter is used to further pre-process the depth map. No objective gains are reported, but a visible improvement is noticed on some synthesized sequences.

A technique that does not necessarily require pre-processing the depth map is introduced in [WLS+11]. A hole in the synthesized texture image is filled by the color of the neighboring pixel (between the 8 direct neighboring pixels) with the smallest depth value in the synthesized depth map (this is referred to as Horizontal, Vertical and Diagonal Extrapolation (HVDE)). The two warped texture images are complemented (holes in one are filled with available pixel values in the other), and later blended, giving a final image W. The same process (HVDE, complementation, and blending) can also be performed in case the depth maps were pre-processed with a bi-lateral smoothing filter, giving an image A, which would then be used to fill remaining holes in W. This technique outperforms basic DIBR by 1.78 dB.

Another method for improving the quality of the synthesis is to apply a non-linear transformation to the depth maps [WZ11]. Specifically, the depth range of points in the background is compressed, such that these points would have the same or slightly different depths. This reportedly reduces holes in the synthesis. The transformation depends on the depth map histogram. Objective gains are not presented but a visible improvement is noticed on the shown images.

7.1.3 Conclusion

The rendering techniques used in the reference softwares, and in most contributions in literature, are all based on 3D image warping using depth maps. Pixels from reference views are mapped to pixels in the virtual view using the disparity information that the depth maps convey. However, we believe that the synthesis can be improved by extending DIBR to the temporal axis. In this work, we present a render-

ing method where temporal correlations between different frames in the synthesized views are exploited to improve the quality of the synthesis. Our method is detailed in the next section.

7.2 Proposed method

Traditional rendering techniques synthesize an intermediate frame only from the left and right reference views, at the same time instant. By exploiting the temporal correlation in the synthesized view we are able to obtain additional reconstructions from past and future frames, and merge them together to obtain the synthesized frame. In this section, we describe the epipolar constraint, on which the proposed method is based. We then provide a description of the algorithm and propose two synthesis schemes for a Group Of Pictures (GOP) that exploit this idea.

7.2.1 Epipolar constraint

Figure 7.3 shows the relation between the positions of a real point projection in different views and different time instants. Let us consider I^r_{t-1}, I^r_t, I^s_{t-1}, I^s_t which are, respectively, the reference (r) view frames and the synthesized (s) view frames at time instants $t-1$ and t. Let $S=(x,y)$ be a point in I^r_{t-1}, $\mathbf{V}_r(S)$ its associated motion vector, pointing to a corresponding point in I^r_t, and $\mathbf{D}_{t-1}(S)$ its associated disparity vector, pointing to a corresponding point in I^s_{t-1}. Let $\mathbf{V}_s(s+D_{t-1}(S))$ the motion vector of the projection of S in I^s_{t-1} and $\mathbf{D}_t(S+V_r(S))$ the disparity vector of the projection of S in I^r_t. Since there is only one projection of S in I^s_t, those two vectors will point to the same position. This defines a so-called epipolar constraint on S, which can be written as:

$$\mathbf{V}_r(S)+\mathbf{D}_t(S+\mathbf{V}_r(S))=\mathbf{D}_{t-1}(S)+\mathbf{V}_s(S+\mathbf{D}_{t-1}(S)) \tag{7.1}$$

7.2.2 Method description

The goal of the method is to synthesize I^s_t. Knowing $\mathbf{V}_r$, $\mathbf{D}_t$, and $\mathbf{D}_{t-1}$, $\mathbf{V}_s$ can be derived using Equation 7.1 for every pixel in I^s_{t-1} that has a correspondence in I^r_{t-1}:

$$\mathbf{V}_s(S+\mathbf{D}_{t-1}(S))=\mathbf{V}_r(S)+\mathbf{D}_t(S+\mathbf{V}_r(S))-\mathbf{D}_{t-1}(S) \tag{7.2}$$

$\mathbf{V}_r$ can be obtained by inputting I^r_{t-1} and I^r_t in an optical flow algorithm (like the one downloadable from [Liu]). The result is a dense motion vector field $\mathbf{V}_r$

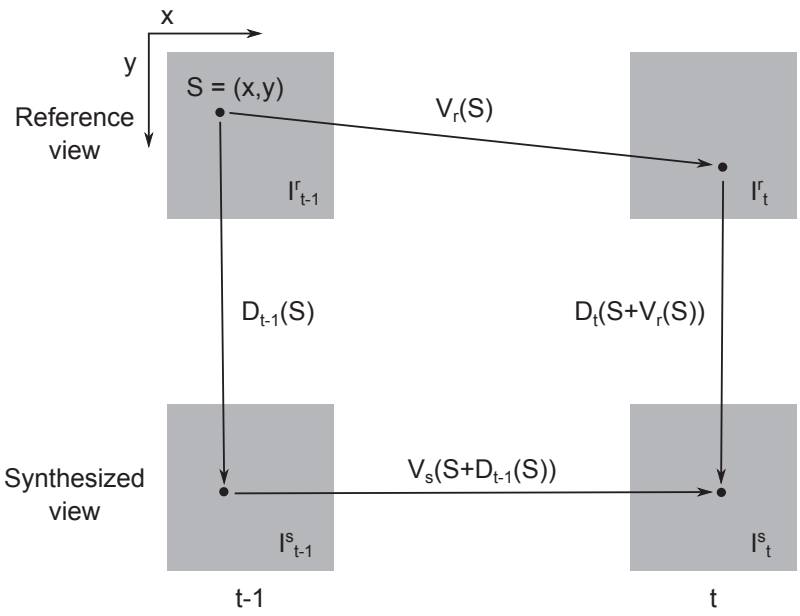


Figure 7.3: Epipolar constraint

where each pixel in I^r_{t-1} is associated with a motion vector (see Chapter 6 for more information on the optical flow). The disparity maps $\mathbf{D}_t$ and $\mathbf{D}_{t-1}$ can be obtained by simply converting the values in the depth maps Z^r_t and Z^r_{t-1} associated with I^r_t and I^r_{t-1} respectively into disparity values. We assume that we are dealing with a 1D parallel camera setup, and that only horizontal disparities exist. In this simple setup, the disparity value for a point of coordinates (x, y) in I^r_{t-1} can be written as:

$$\mathbf{D}^x_t(x,y) = f \cdot t_x \cdot \left[\frac{Z^r_t(x,y)}{255} \left(\frac{1}{Z_{min}} - \frac{1}{Z_{max}} \right) + \frac{1}{Z_{max}} \right] \tag{7.3}$$

$$\mathbf{D}^y_t(x,y) = 0 \tag{7.4}$$

where f is the focal length of the camera, t_x the baseline between the reference and synthesized views, and Z_{min} / Z_{max} the extremal depth values. The same formula can be applied to obtain $\mathbf{D}_{t-1}$. If we decompose Equation 7.2 for the x and y components seperately, we obtain:

$$\mathbf{V}^x_s(x + \mathbf{D}^x_{t-1}(x,y), y) = \mathbf{V}^x_r(x,y) + \mathbf{D}^x_t(x + \mathbf{V}^x_r(x,y), y + \mathbf{V}^y_r(x,y)) - \mathbf{D}^x_{t-1}(x,y) \tag{7.5}$$

$$\mathbf{V}^y_s(x + \mathbf{D}^x_{t-1}(x,y), y) = \mathbf{V}^y_r(x,y) \tag{7.6}$$

There will be holes in $\mathbf{V}_s$ that coincide with disocclusions (areas that are occluded in the reference view and which become visible in the synthesized one) created when warping I^r_{t-1} with the $\mathbf{D}_{t-1}$ disparity vector field. If two or more samples in I^r_{t-1},

S_1 and S_2 for instance, are warped to the same position S_3 in I^s_{t-1} (occlusion), the vector $\mathbf{V}_s(S_3)$ retained is the one which corresponds to the pixel with the highest depth value, as shown in Equation 7.7. The motion vectors for occluded points of the scene are thus ignored.

$$\mathbf{V}_s(S_3) = \begin{cases} \mathbf{V}_r(S_1) + \mathbf{D}_t(S_1 + \mathbf{V}_r(S_1)) - \mathbf{D}_{t-1}(S_1) & \text{if } Z^r_{t-1}(S_1) > Z^r_{t-1}(S_2) \\ \mathbf{V}_r(S_2) + \mathbf{D}_t(S_2 + \mathbf{V}_r(S_2)) - \mathbf{D}_{t-1}(S_2) & \text{otherwise} \end{cases} \tag{7.7}$$

Using $\mathbf{V}_s$ and I^s_{t-1}, a prediction of I^s_t, referred to as $\widehat{I}^s_t$, can be made, although it will contain holes due to the holes in $\mathbf{V}_s$. A total of four predictions can be made by exploiting the epipolar constraint, one for each reference view (left and right) and each time instant (past and future): $\widehat{I}^s_t[i]$ where $i \in \{0, 1, 2, 3\}$. This is shown in Figure 7.4.

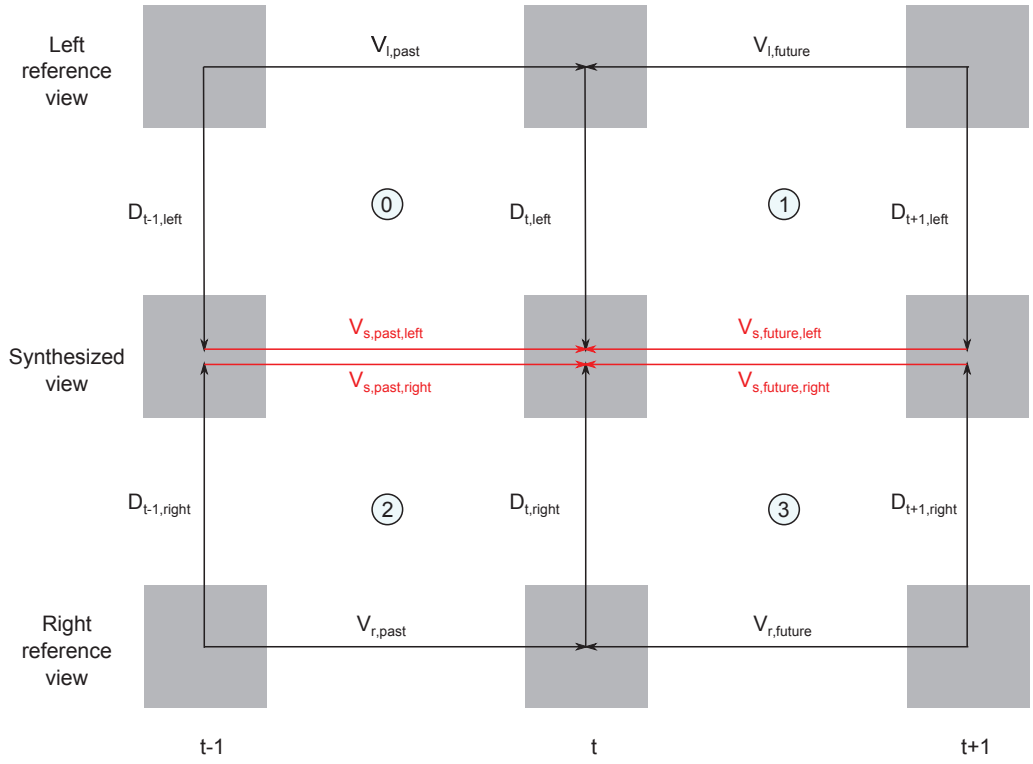


Figure 7.4: Four predictions using the epipolar constraint

The four predictions are then merged into one prediction $\widetilde{I}^s_t$, where the value of each pixel equals the average of the non-hole pixel values in the four predictions. Otherwise the pixel value equals 0. Indeed, while the four predictions contain holes, the majority of these holes are not the same in all predictions and thus, they will be

filled after the merging step.

$$\widetilde{I}_t^s(S) = \begin{cases} \dfrac{\sum_{i=0}^{3} \widehat{I}_t^s[i](S)}{A(S)} & \text{if } A(S) \neq 0 \\ 0 & \text{if } A(S) = 0 \end{cases} \tag{7.8}$$

where $A(S)$ is given by:

$$A(S) = \sum_{i=0}^{3} \left(\widehat{I}_t^s[i](S) \neq 0 \right) \tag{7.9}$$

Remaining holes (zero values) are filled using the same inpainting method used in VSRS-1DFast. The inpainting is done line-wise from left to right. If a hole is encountered, it is filled with its left neighboring pixel. If the first pixel in a line is a hole, it is filled with the first non-hole pixel to its right.

7.2.3 Prediction schemes in a GOP

The synthesized view is rendered GOP-wise in our algorithm. The GOP structure is the one used to code the left and right reference views. Figure 7.5 shows the open GOP structure of a reference view (8 frames per GOP in this example). We choose to send only the first frame per GOP of the synthesized view in the bitstream. These frames, referred to in the rest of this work as key frames, are coded using 3D-HEVC with the left view serving as an inter-view reference (the base view). The rest of the frames are synthesized using our method. For the first frame actually synthesized in each GOP, the key frame of the current GOP and the one of the future GOP represent respectively the past and future reference synthesized frames used in our algorithm (I_{t-1}^s and I_{t+1}^s). Other frames in the GOP are synthesized using one of two proposed prediction schemes.

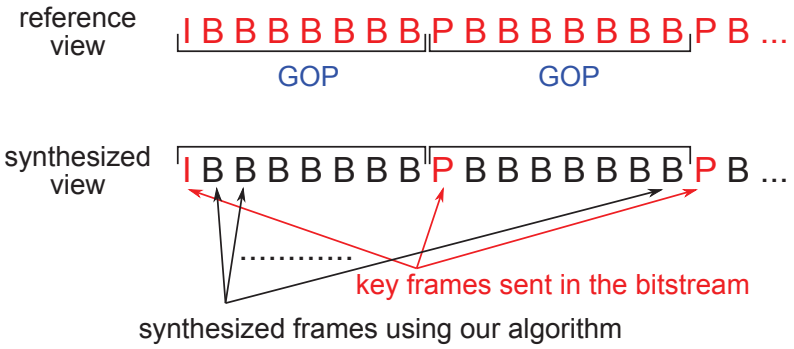
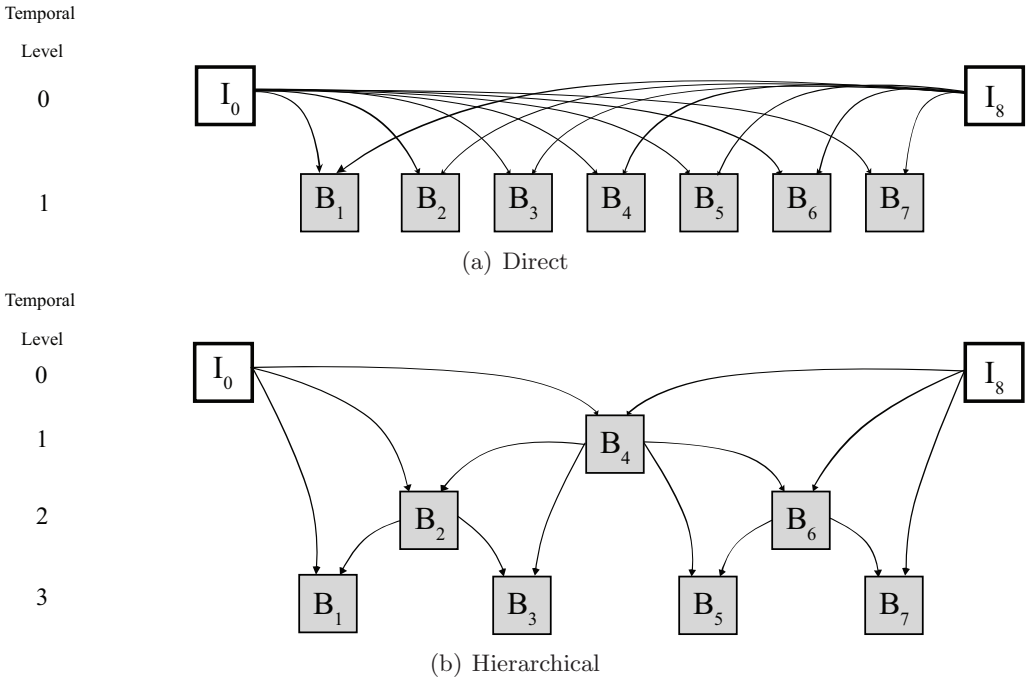


Figure 7.5: GOP-wise rendering in our method

Figure 7.6 shows the difference between these two schemes. The “Direct” scheme

uses the key frame of the current GOP and the one of the next GOP as past and future reference frames for all remaining frames to synthesize in the GOP. This results in an asymetric prediction, with two different temporal distances between each of the two key frames and the current frame. The temporal distance can be as high as the GOP size minus one, and an optical flow computation with such large temporal distances can give imprecise motion vector fields. Since there are no weights introduced in the merging process to lower the impact of predictions with high temporal distance, the "Direct" scheme can thus be inefficient. An alternative scheme, called the "Hierarchical" scheme, is proposed in this work in which temporal layers are used to perform symetric predictions (with equal temporal distances). In each layer, the past and future references for the current frame are either the key frames or already synthesized frames in lower layers. The maximal temporal distance in this scheme equals half the GOP size.



(a) Direct

(b) Hierarchical

Figure 7.6: Prediction schemes in a GOP

7.2.4 Discussion on the method

Our rendering method assumes that the synthesized view is available at the encoder side, since one frame per GOP of that view is transmitted in the bitstream. This is a reasonable assumption since in dense camera rig systems, all the views are available but only a subset is coded and sent in the bitstream, the rest being synthesized at the receiver side. Indeed, we have shown in Table 1.4 that synthesizing the intermediate

views instead of sending them is a more efficient alternative. Our method can be seen as set in between those two scenarios: we only send some information on the synthesized views, which we exploit to improve the synthesis. Consequently, in this work, we not only propose a rendering method, but also a change in the design of the whole transmission stage. Note that we could have proposed a method where the key frames in the synthesized view are rendered with the left and right reference views using VSRS for instance, but then the rendering artifacts created in those key frames would be propagated to the rest of the frames in the motion compensation stage.

Furthermore, we use a "backward" motion compensation stage in our method: the vectors in $\mathbf{V}_s$ point from I^s_{t-1} (or I^s_{t+1}) to I^s_t, and not the other way around. We can have a $\mathbf{V}_s$ that points from I^s_t to a past or future reference if the vectors in $\mathbf{V}_r$ point in the same direction (*e.g.*, from I^r_t to I^r_{t-1} (or I^s_{t+1})). This can easily be done if the inputs of the optical flow algorithm that outputs $\mathbf{V}_r$ are inversed. In this case, and if $S = (x, y)$ is a point in I^r_t, Equation 7.2 becomes:

$$\mathbf{V}_s(S + \mathbf{D}_t(S)) = \mathbf{V}_r(S) + \mathbf{D}_{t-1}(S + \mathbf{V}_r(S)) - \mathbf{D}_t(S) \tag{7.10}$$

From Equation 7.10, we can see that $\mathbf{V}_s$ is now defined for every pixel in I^r_t that has a correspondence in I^s_t. The holes in $\mathbf{V}_s$ (and in the corresponding prediction) correspond to disocclusions when warping from I^r_t to I^s_t. Even if we use a different time instant $(t+1)$, the holes in the corresponding prediction would still come from warping I^r_t to I^s_t and thus will coincide with the holes of the first prediction. The merging process will not be able to fill these holes and they will eventually have to be inpainted. In our method, the holes correspond to disocclusions when warping from I^r_{t-1} to I^s_{t-1} in the first prediction, and from I^r_{t+1} to I^s_{t+1} in the second. The holes do not necessarily coincide, and thus, they can be efficiently filled in the merging process.

7.3 Experimental results

7.3.1 Experimental setting

We implement our view synthesis method, which we refer to in this section as View Synthesis exploiting Temporal Prediction (VSTP), in MATLAB. Our algorithm takes as input two coded left and right views with their associated depth videos and camera parameters, and one frame per GOP of the intermediate view, and outputs the whole intermediate view after synthesizing the rest of the frames. We thus consider a two-view scenario in these experiments in which we code two views

(left and right) and synthesize the middle one. We could not consider a three-view scenario in which we code three views and synthesize three other intermediate views between each pair, as indicated in the CTCs [RMV13], because we need to compare the synthesis result to the original intermediate sequences to measure the PSNR and those are not available for all the sequences considered. The coding configuration defined in the CTCs is nevertheless used for coding the left and right view. The optical flow algorithm downloadable from [Liu] is used in our method, with the configuration parameters reported in Table 6.2.

We test our method on four sequences of the test set defined in the CTCs: Balloons, Kendo, Newspaper and PoznanHall2. Each sequence is composed of three views. The CTCs indicate to use the middle view as base view, and the left and right views as dependent views. However here, we want the left and right views to be decodable without the middle view because only the first frame in each GOP of that view will be sent in the bitstream. We thus set the left view as base view, and the others as dependent views. Also, we choose to only code 3 seconds of video of each sequence in order to speed up the simulations. We believe this is acceptable, because coding the equivalent of 87 frames for Balloons, Kendo and Newspaper, and 58 frames for PoznanHall2 covers around 10 and 7 GOPs respectively, and in our method, the synthesis in one GOP is independent of the synthesis in the others. Note that the number of coded frames is lower in PoznanHall2 because its frame rate is lower as well (cf. Table 3.2).

We compare our synthesis method to the reference VSRS-1DFast rendering. We evaluate the performance of the reference and the proposed methods using the Bjontegaard delta-PSNR (BD-PSNR) metric on the synthesized view. The PSNR is evaluated against the original intermediate view. The rate in the reference method is the sum of the rates needed to code the left and right views with their associated depth videos. The same rate is considered in the proposed method, to which is added the rate needed to code the first frame in each GOP of the intermediate view. We could not use the Bjontegaard delta-Rate (BD-Rate) metric here because in some cases, the ranges of PSNR values are not superposed enough to justify the computation of the "average rate reduction at the same PSNR" (see Figure 7.7). On the contrary, the rate ranges are superposed so we are able to compute the BD-PSNR.

7.3.2 Synthesis results

Table 7.1 gives the BD-PSNR values obtained with the two prediction schemes in the proposed method. A positive value in this table indicates a gain. On average, our method brings 0.717 dB and 1.391 dB PSNR increase with the "Direct" and the "Hierarchical" schemes respectively.

Sequence	BD-PSNR (in dB)	
	Direct	Hierarchical
Balloons	3.150	3.511
Kendo	-0.556	0.205
Newspaper	-0.404	0.442
PoznanHall2	0.678	1.406
Average	**0.717**	**1.391**

Table 7.1: BD-PSNR values obtained with both prediction schemes in the proposed method

The RD curves for the reference and the proposed method (for both schemes) are given in Figure 7.7. We can see that while both schemes outperform the reference method for Balloons and PoznanHall2, our method outperforms the reference only with the "Hierarchical" scheme in Kendo and Newspaper. This is also represented in BD-PSNR values for these two sequences which are only positive in the "Hierarchical" scheme, as shown in Table 7.1.

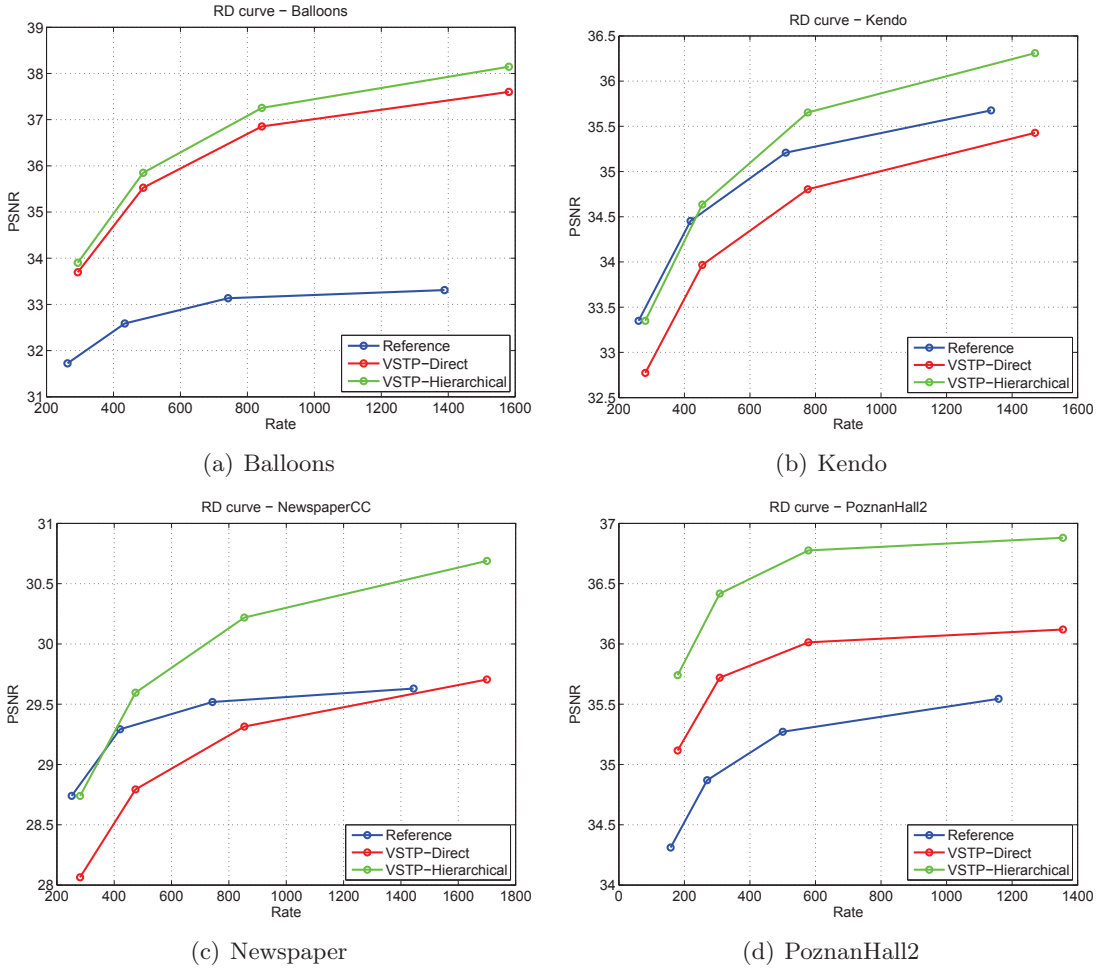
Figure 7.8 shows, for the four tested sequences, the variation of the PSNR of the synthesized view over time with the reference and the proposed method (both schemes). Only one QP (25) is represented for simplicity as the behavior of any curve is similar across all QPs. In the proposed method and for all sequences, we notice periodic peaks in the synthesized view PSNR, which correspond to the first frame of each GOP. Since these frames are not synthesized but rather decoded, their PSNR is higher than any other frames in the GOP. For the Balloons sequence, the proposed method outperforms the reference VSRS-1DFast rendering for all frames. For the other sequences, our method is better only in certain parts.

Figure 7.9 shows parts of frames synthesized using the reference and the proposed method. For fairness of comparison, for our method, we show frames that are actually synthesized and not decoded. We can notice a clear improvement in the synthesis quality with our method: the artifacts obtained with VSRS-1DFast (highlighted in red in the figures) are efficiently removed.

7.3.3 Results interpretation

The results of Table 7.1 and the RD curves in Figure 7.7 show that the "Hierarchical" scheme outperforms the "Direct" scheme, which was expected, since the prediction distances are shorter in the first. Note that in a GOP of 8 frames, the fifth frame is synthesized in the same way in both shemes, which is why the curves of Figure 7.8 corresponding to the two schemes, intersect not only in the first frame of each GOP but also in the fifth frame.

Figure 7.8 also show that the proposed method sometimes does not perform well



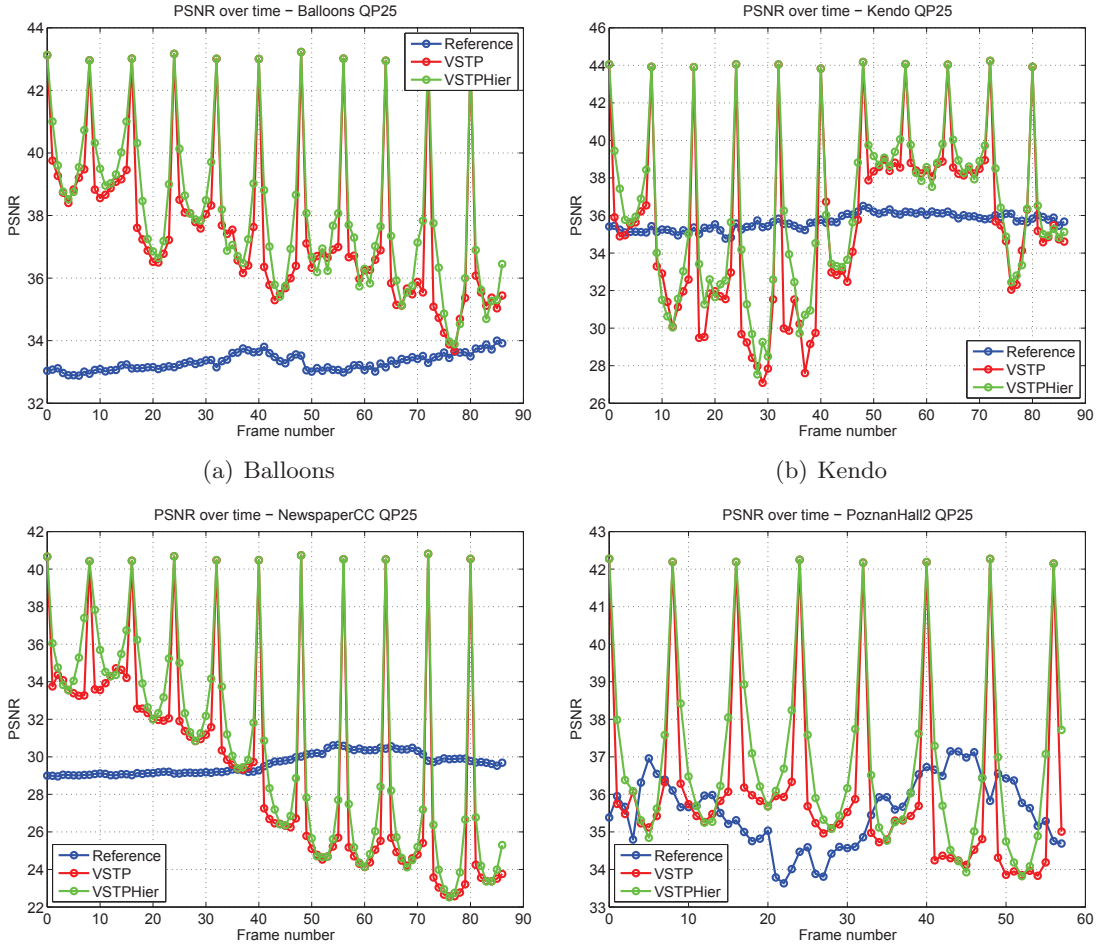
(a) Balloons (b) Kendo

(c) Newspaper (d) PoznanHall2

Figure 7.7: RD curves of the reference and proposed method for the four tested sequences

on some series of frames, especially in the Kendo or Newspaper sequence. This is due to the dense motion estimation process in the reference view which gives incorrect motion vectors when there is high intensity motion. Tweaking the optical flow parameters can account for this and would thus solve the problem but that would imply an additional rendering complexity and coding overhead if those parameters are to be sent for each frame.

Our method improves the quality of the synthesis on two levels: first, it accounts for a difference in illumination between the coded reference views and the synthesized view, which rendering techniques such as VSRS-1DFast cannot do. Indeed, while VSRS-1DFast cannot warp a different illumination level from the reference views into the synthesized view, our method propagates the correct illumination level of the sent key frames accross the rest of the frames using motion compensation. Second, our method fills holes due to disocclusions more efficiently than VSRS-1DFast. Indeed, these holes are filled using inpainting in the latter, hence creating artifacts such as



(a) Balloons (b) Kendo

(c) Newspaper (d) PoznanHall2

Figure 7.8: Variation of the PSNR of the synthesized view over time for the reference and proposed method at QP 25

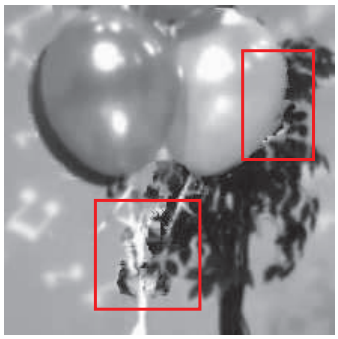
the ones highlighted in Figure 7.9. In our method, the disocclusion areas can be found in previously synthesized frames. In addition, four predictions (as opposed to two in VSRS-1DFast) are merged to give the final synthesized frame, hence reducing the chance of having residual holes after merging. This explains how our method efficiently removed the aforementionned artifacts, as shown in Figure 7.9.

Finally, since our method is implemented in MATLAB, whereas VSRS-1DFast is a binary compiled from C++ code, it was difficult to compare the computation complexity of the two methods. We believe our method is more inherently complex than VSRS-1DFast due to the complex dense motion estimation / compensation stage, which has to be repeated four times per frame. Shortcuts that can reduce the complexity of our method, at the price of losing some prediction accuracy, include block-based motion estimation / compensation and uni-predictive motion compensation (predict using only a past frame, or only a future frame).

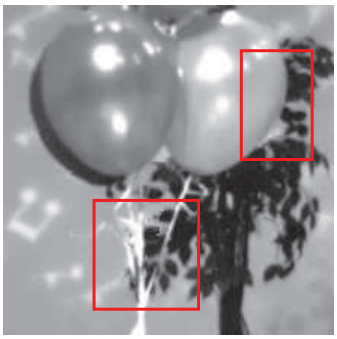
7.4 Conclusion

In this chapter, we presented a view synthesis technique that exploits temporal predictions in order to improve the quality of the synthesis. Namely, some key frames of the synthesized view are sent in the bitstream, and the rest are interpolated using motion compensation with vectors warped from reference views. Four predictions using the left and right reference view, and a past and future time instant can be constructed and then merged together into a single prediction of the synthesized frame. Two prediction schemes referred to as "Direct" and "Hierarchical" are presented in this work. The first synthesizes frames using motion compensation with only key frames, while the other motion compensates with previously synthesized frames, hence reducing the prediction distances. Our method brings 0.717 dB and 1.391 dB PSNR increase with the "Direct" and "Hierarchical" schemes respectively. At the time of writing this thesis, a journal article reporting these results and investigating ways to further increase the gains is being drafted.

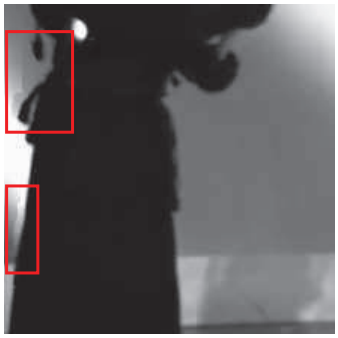
In the future, the method can be implemented in C++ to speed-up the rendering simulations and allow direct comparisons with VSRS-1DFast. Furthermore, two additional predictions can be obtained by simply warping the left and right reference views to the level of the synthesized view, as done in VSRS-1DFast. An adaptive blending can judiciously select either VSTP or VSRS-1DFast to synthesize a specific pixel. Finally, the frequency at which key frames are sent in our method, which, in the current version, follows the GOP structure used for coding the reference views, can be modified: lower frequencies allow bitrate savings since less key frames will be sent but they also imply motion estimation between distant frames, which will decrease the prediction accuracy. The trade-off is an interesting research subject.



(a) Balloons - Reference - QP 25



(b) Balloons - VSTP "Direct" - QP 25

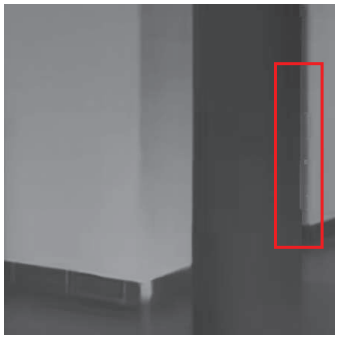


(c) Kendo - Reference - QP 30

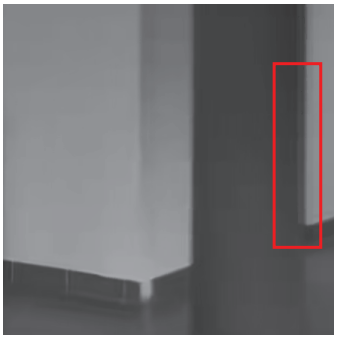
(d) Kendo - VSTP "Hierarchical" - QP 30

(e) Newspaper - Reference - QP 35

(f) Newspaper - VSTP "Hierarchical" - QP 35



(g) PoznanHall2 - Reference - QP 40



(h) PoznanHall2 - VSTP "Direct" - QP 40

Figure 7.9: Parts of frames synthesized with the reference and the proposed method

Conclusions & future work

Thesis objectives

The purpose of this thesis was to develop original coding methods aimed to increase the coding efficiency of dependent views, depth videos and synthesized views in 3D-HEVC. Two research phases have structured this thesis. The methods developped during the first phase comply to real-world constraints in terms of complexity, memory usage, and practicality. They are thus more aimed towards standardization and potential adoption in 3D-HEVC or in future standards. The second phase was more dedicated to research, in the sense that all constraints were dropped, hence giving more freedom to develop unconventional and innovative techniques to increase coding efficiency, regardless of the introduced complexity. The methods developped during this second phase are based on motion estimation using optical flow. The common denominator of all methods though is the improvement of predictions, whether they are pixel, motion vector, Intra mode or partition information predictions, using already available information (*i.e.*without sending any additional side information).

Summary

Methods aimed towards standardization

To increase the coding efficiency of 3D-HEVC, we first perform a test to see which signaling information costs the most to code in the bitsream since targetting that information allows to maximize coding gains. We find that the rate needed to code the depth Intra modes is evaluated at 25% of the depth bitstream and is the most costly element. Thus, the first proposed method aims at reducing this rate by introducing for each depth PU, a new Intra mode predictor. The Intra mode of the texture PU associated with the current depth PU appears to be a reasonable choice for a predictor, since texture and depth represent the same scene at the same time instant. However, when analyzing the Intra modes of texture and depth, we find that they match mostly on PUs where there is a sharp edge in texture. Consequently,

we propose to inherit, only for these PUs, the texture Intra mode for depth by inserting it in the MPM list of the depth PU, where it will serve as a predictor for the depth Intra mode. The inheritance is thus driven by a criterion, GradientMax, which measures the sharpness of the edge present in the texture PU, and only occurs when the criterion value exceeds a specified threshold. The method brings coding gains on depth videos, and on views synthesized using these depth videos. A further analysis shows that the texture / depth Intra mode matchings actually occur on PUs where there is a single sharp directional edge in texture. Indeed, having two sharp edges, or one sharp edge with no dominant direction decreases the pertinence of the texture Intra mode as a depth mode predictor. A new criterion, DominantAngle, which measures the sharpness as well as the direction of the edge, is presented. We find that DominantAngle yields more coding gains and is also more robust than GradientMax with respect to the threshold selection.

After improving the Intra coding mode in 3D-HEVC, we now focus on Inter coding. We first notice that DCP is not selected often in 3D-HEVC: there are more PUs coded with a MV than a DV. While it is true that there are, in general, more correlations along the temporal axis than in the view axis, the main reason behind poor DCP selection is the lack of DV candidates in the Merge list. Not having DV candidates in this list penalizes DCP as the only other option to code a PU in DCP is to send a DV residual, which is often costly. Merge mode with a MV candidate which might not be as accurate as a DV is still preferred by the HTM encoder. Note that IVMP adds a multiview candidate in the list, but this candidate is preferred to be a MV rather than a DV. It is basically the MV of the base view PU corresponding to the current PU in the dependent view. The correspondance is made using $\mathrm{DV}_{\mathrm{IVMP}}$, a DV derived using NBDV. The multiview candidate corresponds to $\mathrm{DV}_{\mathrm{IVMP}}$ if and only if the MV is not available. We propose to always add $\mathrm{DV}_{\mathrm{IVMP}}$ as an inter-view candidate along side the multiview candidate if the latter is a MV (otherwise no new candidates are added). Two insertion methods are proposed, one which inserts the candidate in the secondary Merge candidate list (AIMC-1), and the other in the primary list (AIMC-2). In both cases, a redundancy check with all preceding candidates is performed. Both methods achieve coding gains, and AIMC-2 was adopted in the 3D-HEVC working draft and software. The NBDV process to derive $\mathrm{DV}_{\mathrm{IVMP}}$ can also be improved. Indeed, NBDV is sub-optimal because the first neighboring DV found in the NBDV search process is selected as $\mathrm{DV}_{\mathrm{IVMP}}$ with no guarantee that this DV is the most coding efficient. There might be other neighbors which have a better DV, and those are not checked in NBDV because the search process stops after finding the first DV. We propose to save all neighboring DVs in a list (the search process is never stopped). Redundant vectors are removed from the

list, and the median of the remaining vectors is set as DV_{IVMP}. The method brings interesting coding gains and a variant, which only saves up to four DVs in the list, successfully reduces the maximum number of vectors for median computation from 14 to 4 with only a slight decrease in coding efficiency.

Another costly signaling element is partition information, which is used in 3D-HEVC to describe the quadtree partitionning of a frame into CUs and PUs. Indeed, texture and depth partition information take respectively 6.5% and 1.6% of the total bitstream in HTM-4.0. When analyzing the quadtree information of a texture frame and its associated depth map, we find that in general, the texture quadtree is more finely partitionned than the depth quadtree. Two methods are developped based on that assumption: a texture quadtree initialization (QTI) where the texture quadtree is initialized from the depth quadtree, and a depth quadtree limitation (QTL) where the depth quadtree is limited to that of the texture. QTI and QTL allow encoder runtime savings because in the first case, a split may be directly forced without the need for RDO to check coding modes at the current depth level, and in the latter, splitting may be prohibited hence alleviating RDO checks in smaller partitions. A predictive coding (PC) of the partition information, allowing bitrate savings, can be applied in each method. On the one hand, QTI+PC yields 5% encoder runtime savings with 0.5% coding loss on coded and synthesized views. A more relaxed variant called QTI+1+PC where a texture CU is allowed to be at least one level less partitionned than its collocated depth CU outperforms QTI+PC with 6% encoder runtime savings and no coding loss. Other encoder shortcuts like Early CU termination (ECU) applied on texture yield much higher runtime savings (37%) at the cost of higher coding losses though (0.7% on coded videos). QTI+1+PC might thus be preferred in applications where coding performance is paramount. On the other hand, QTL+PC brings 31% encoder runtime savings with -0.3% gain on coded + synthesized views and 0.4% loss on synthesized views alone. However, we show that these losses are not actual losses as they are due to smoothing false contours. Subjective results on synthesized views actually show an improvement resulting from the use of QTL+PC. QTL+PC also clearly outperform all depth-based state-of-the-art encoder shortcuts in both coding gains and runtime savings. QTL+PC was adopted in the 3D-HEVC working draft and software.

Methods based on optical flow

Optical flow is a method used to estimate a dense motion vector field (DMVF) describing the motion of each pixel from one frame to another. In the field of video compression, it can replace block matching algorithms in Inter mode which estimate a single vector for an entire block of pixels. A dense motion vector field allows

constructing a more accurate block predictor hence reducing the residual energy and giving bitrate reductions. However, sending one vector per pixel is costly and it can quickly compensate all the potential coding gains of a dense motion estimation. We propose to combine optical flow with a smart decoder approach in 3D-HEVC. Specifically, two dense motion vector fields are computed in the base view between the frame at the current time instant and the first frames in its L0 & L1 reference lists. All these frames are reconstructed so that the optical flow computation can be performed at the decoder in a similar way. For a currently coded PU in the dependent view at the current time instant, a corresponding PU is found in the base view using disparity-compensation with a single DV derived using NBDV, and the two DMVFs of that PU are inherited as a single bi-predictive dense Merge candidate in the candidate list of the current PU. Two variants that disparity-compensate with a dense disparity vector field using either the reconstructed base depth view (DO-WOF) or the already coded dependent view depth map (FCO-WOF) in case the coding order is depth-before-texture are proposed. Significant coding are reported for all three variants, FCO-WOF being slightly better than the other two. The gains result from an improvement of the block predictions, and from savings on split and skip flags. We also find that the methods become more efficient at low bitrates since Merge mode is already selected more often there.

Optical flow can also be used to improve view synthesis. We propose a novel view synthesis algorithm that computes four synthesized frame predictions and blends them together to obtain the final synthesized frame. The predictions are constructed using motion-compensation with four DMVFs. These DMVFs are computed at the level of the two left and right coded reference views using a past and a future reference, then warped to the level of the synthesized view using an epipolar constraint. For initialization of the algorithm, the first frames in each GOP (also called key frames) of the synthesized view must be sent in the bitstream. Two prediction schemes are proposed: a direct scheme where the key frames are used as past and future references for all remaining frames in the GOP, or an hierarchical scheme where the references are either key frames or other previously synthesized frames in the GOP. We find that both schemes achieve significant PSNR increase on the synthesized view. The hierarchical scheme outperforms the direct one because it reduces the prediction distances.

Perspectives for future work

At the time of concluding this manuscript, several interesting perspectives can be proposed to further continue the work done in this thesis. The primary points

concern the optical flow-based approaches, but some short-term improvements to the methods detailed in the first part of the manuscript can be proposed as well.

For the texture Intra mode inheritance for depth coding approach, we saw that the method was particularly sensitive to the selected threshold. For maximal coding results, the threshold should be optimized not only per sequence, but per frame. To address this, the threshold can be made content-adaptive. It can be initialized to a certain value, then updated as the number of coded PUs increases. Furthermore, a higher additional threshold can be adaptively set as well. If the metric exceeds this larger threshold, this means that the texture Intra mode will most probably match the depth Intra mode, and in this case, the depth PU can be directly coded with the inherited texture Intra mode, hence saving both on runtime (no need for RDO checks of other Intra modes) and on bitrate (no need to code the inherited mode). If the metric falls between the two thresholds, the proposed method is applied. Otherwise, the texture Intra mode is not inherited.

As for the added DV inter-view Merge candidate for dependent view PUs, it could be driven by a criterion which estimates, using decoded information to avoid sending any additional information, the benefits of actually having the candidate in the list. Indeed, not inserting the candidate can, in some cases, be more beneficial if it would increment the Merge indices of better candidates further down the list, hence increasing their signaling cost. Regarding the modification of the NBDV process, all the neighboring DVs could be tested by RDO and the most efficient DV can be selected as $\mathrm{DV}_{\mathrm{IVMP}}$ (and even $\mathrm{DV}_{\mathrm{IVRP}}$). This implies sending the index of the most efficient neighbor in the bitstream but given the significant impact a better DV selection can bring on coding gains, the method is worth to be tested.

For the texture quadtree initialization, alternative schemes with different severity levels can be tested. The more severe the scheme is the more bitrate reductions (on partition information) and runtime savings it allows to make, but the more it alters the quadtree from what the RDO process intended it to be (hence yielding coding losses) as well. We believe the severity level of QTI can be tweaked in order to obtain a more efficient scheme. As for QTL, its interaction with assymetric motion partitions (AMP), recently enabled in HTM, can now be studied.

For unconstrained approaches developped in the second phase of the thesis, more challenging improvements can be proposed: the coding efficiency of the warped optical flow method in 3D-HEVC can be increased if the DMVF in the base view is corrected when warped to the level of the dependent view using the epipolar constraint described in Chapter 7. Furthermore, to decrease complexity, the OF code can be revised to use less complex data structures, the OF parameters can be tweaked, and a DMVF can be estimated per block rather than per frame. For

VSTP, several improvements can be made as well: first, two additional predictions can be constructed by warping the left and right reference views to the level of the synthesized view as it is done in VSRS-1DFast. An adaptive blending, judiciously selecting to synthesize pixels using either a blend of the four VSTP predictions or a blend of the two VSRS-1DFast predictions, can be proposed. This allows selecting VSRS-1DFast in areas where there is high motion in the video, and in which the optical flow computation gives incorrect motion vectors. Second, the GOP size used in VSTP can be tweaked: lowering it shortens the prediction distances and gives more accurate predictions but requires sending more frames of the synthesized view in the bitstream, and vice-versa. An optimal GOP size yielding the most advantageous trade-off can be found this way. Finally, implementing VSTP in a C++ environment while using more appropriate data structures can considerably reduce the rendering runtime, and allows comparing VSTP directly with VSRS-1DFast to assess its complexity. Efficient implementations of the OF estimation in C++ or on GPU can also fasten the encoding and decoding runtime of these methods.

Publications

Journal articles

1. E.G. Mora, J. Jung, M. Cagnazzo and B. Pesquet-Popescu, "Depth video coding based on Intra mode inheritance from texture", *APSIPA Transactions on Signal and Information Processing* , 2013.

2. E.G. Mora, J. Jung, M. Cagnazzo and B. Pesquet-Popescu, "Initialization, limitation and predictive coding of the depth and texture quadtree in 3D-HEVC Video Coding", *IEEE Transactions on Circuits and Systems for Video Technology* , 2013

Conference papers

1. E.G. Mora, J. Jung, B. Pesquet-Popescu and M. Cagnazzo, "Codage de vidéos de profondeur basé sur l'héritage des modes Intra de texture", Proceeding of *COmpression et REprésentation des Signaux Audiovisuels (CORESA)* , Lille, France, May 2012.

2. E.G. Mora, J. Jung, M. Cagnazzo and B. Pesquet-Popescu, "Modification of the Merge candidate list for dependent views in 3D-HEVC", Proceeding of *IEEE International Conference on Image Processing (ICIP)* , Melbourne, Australia, September 2013.

3. E.G. Mora, J. Jung, B. Pesquet-Popescu and M. Cagnazzo, "Modification of the disparity vector derivation process in 3D-HEVC", Proceeding of *IEEE International workshop on Multimedia Signal Processing (MMSP)* , Sardinia, Italy, September 2013.

Book Chapters

1. E.G. Mora, G. Valenzise, J. Jung, B. Pesquet-Popescu, M. Cagnazzo and F. Dufaux, "Depth video coding technologies", in *Emerging Technologies for 3D Video* , chapter 7, pp. 121-137, Wiley, 2013

2. E.G. Mora, J. Jung, B. Pesquet-Popescu and M. Cagnazzo, "Méthodes de codage de vidéos de profondeur", in *Vidéo 3D : Capture, traitement et diffusion* , chapter 12, pp. 233-250, Hermes, 2013

Patents

1. J. Jung and E.G. Mora, “Procédé de codage vidéo mettant en jeu un héritage contrôlé d’informations de codage”, 2012

2. J. Jung and E.G. Mora, “Procédé de codage vidéo 3D permettant d’améliorer le calcul d’un vecteur de disparité”, 2013

3. J. Jung, E.G. Mora, B. Pesquet-Popescu, and M. Cagnazzo “Procédé de codage vidéo utilisant une estimation de mouvement dense”, 2013

Technical standardization contributions

1. J. Jung and E. Mora, “3D-CE3.h: Depth Quadtree Prediction for 3DHTM 4.1”, JCT3V-B0068, October 2012

2. E.G. Mora, B. Pesquet-Popescu, M. Cagnazzo and J. Jung, “3D-CE5.h related: Modification of the Merge Candidate List for Dependant Views in 3DV-HTM”, JCT3V-B0069, October 2012

Non-technical standardization contributions

1. J. Jung and E. Mora, “CE6e: Cross check of experiment 1f+2b+3b (LG - JCTVC-Exxxx)”, JCTVC-E450, March 2011

2. F. Henry, G. Clare, E. Mora and J. Jung, “3D-CE5.h: Cross-check of reducing the coding cost of merge index by dynamic merge index re-allocation (JCT3V-B0078)”, JCTVC-B0097, October 2012

3. F. Henry, G. Clare, E. Mora and J. Jung, “3D-CE5.h: Cross-check of merge candidate list for disparity compensated prediction (JCT3V-B0080)”, JCT3V-B0098, October 2012

4. J. Jung and E. Mora, “3D-CE3.h related: Cross check of JCT3V-B156 on Improved Motion Parameter Inheritance”, JCT3V-B0205, October 2012

5. J. Jung and E. Mora, “3D-CE5.h related: Cross check of JCT3V-B0136 on support of parallel merge in disparity vector derivation”, JCT3V-B0208, October 2012

6. J. Jung and E. Mora, “3D-CE5.h related: Cross check of JCT3V-B0089 on Improvement of MV candidates in 3DVC”, JCT3V-B0211, October 2012

7. J. Jung and E. Mora, “Subjective Test Results on QuadTree limitation and predictive coding (JCT3V-B0068)”, JCT3V-B0222, October 2012

8. J. Jung, E. Mora, “3D-CE5.h: Cross check of JCT3V-C0148 on additional merge candidates derived from shifted disparity candidate predictors”, JCT3V-C0151, January 2013

9. J. Jung and E. Mora, “3D-CE6.h related: Crosscheck of JCT3V-C0160 on modified depth coding in random access unit”, JCT3V-C0167, January 2013

10. J. Jung and E. Mora, “3D-CE3.h-related: Comments on contribution JCT3V-D0121 on early depth CU decision”, JCT3V-D0303, April 2013

Bibliography

[3Dv] "3D Vivant", http://www.3dvivant.eu/. *Cited in Sec.* 1.1.2

[ALC96] E. Ardizzone and M. La Cascia, "Video indexing using optical flow field", in *IEEE International Conference on Image Processing (ICIP)*, vol. 3, pp. 831–834 vol.3, 1996. *Cited in Sec.* 6.1.1

[Ali] "Ali Express", www.aliexpress.com. *Cited in Sec.* 1.1.2

[BF06] A. Bourge and C. Fehn, "Auxiliary video data representations", ISO/IEC JTC1/SC29/WG11 23002-3 N8038, April 2006. *Cited in Sec.* 1.3.2

[BHO+13] B. Bross, W.-J. Han, J.-R. Ohm, G. Sullivan, Y.-K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & last call)", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCTVC-L1003, January 2013. *Cited in Sec.* 1.3.1

[Bjo01] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves", in *VCEG Meeting*, Austin, USA, April 2001. *Cited in Sec.* 1.3.2, 3.3.1

[BLU11] O. Bici, J. Lainema, and K. Ugur, "Non-CE13: Simplification of merge mode", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCTVC-G593, November 2011. *Cited in Sec.* 2.3.3

[BSM+09] H. Brust, A. Smolic, K. Mueller, G. Tech, and T. Wiegand, "Mixed resolution coding of stereoscopic video for mobile devices", in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pp. 1 –4, May 2009. *Cited in Sec.* 1.2.1

[BYN11] G. Bang, S. Yoo, and J. Nam, "Description of 3D video coding technology proposal by ETRI and Kwangwoon university", ISO/IEC JTC1/SC29/WG11 MPEG2011/M22625, November 2011. *Cited in Sec.* 2

[CFP11] "Call for Proposals on 3D video coding technology", ISO/IEC JTC1/SC29/WG11 N12036, March 2011. *Cited in Sec.* 1.3.2

[CGT+12] S. Cherigui, C. Guillemot, D. Thoreau, P. Guillotel, and P. Perez, "Hybrid template and block matching algorithm for image intra prediction", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 781–784, 2012. *Cited in Sec.* 6.1.2

[CHWZ08] Y. Chen, W. Hu, O. Wu, and X. Zeng, "Data indexing for video shot mining based on optical flow", in *5th International Conference on Visual Information Engineering*, pp. 88–93, 2008. *Cited in Sec.* 6.1.1

[CJ12] K. Choi and E. S. Jang, "Fast coding unit decision method based on coding tree pruning for High Efficiency Video Coding". *Optical Engineering*, vol. 51 (3), pp. 030502–1–030502–3, 2012. *Cited in Sec.* 5.1

[CLLY08] C.-M. Cheng, S.-J. Lin, S.-H. Lai, and J.-C. Yang, "Improved novel view synthesis from depth image with large baseline", in *19th International Conference on Pattern Recognition (ICPR)*, pp. 1–4, 2008. *Cited in Sec.* 1.2.2

[CMMPP09] M. Cagnazzo, W. Miled, T. Maugey, and B. Pesquet-Popescu, "Image interpolation with edge-preserving differential motion refinement", in *16th IEEE International Conference on Image Processing (ICIP)*, pp. 361–364, 2009. *Cited in Sec.* 6.1.1

[CMPP09] M. Cagnazzo, T. Maugey, and B. Pesquet-Popescu, "A differential motion estimation method for image interpolation in Distributed Video Coding", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1861–1864, 2009. *Cited in Sec.* 6.1.1

[CTEC07] Y. Chi, T. Tran, and R. Etienne-Cummings, "Optical flow approximation of sub-pixel accurate block matching for video coding", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, pp. I–1017–I–1020, 2007. *Cited in Sec.* 6.1.1

[CTMS03] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors". *ACM Transactions on Graphics*, vol. 22 (3), pp. 569–577, July 2003. *Cited in Sec.* 1.2.1

[CWTL12] Y.-L. Chang, C.-L. Wu, Y.-P. Tsai, and S. Lei, "CE5.h related: Depth-oriented Neighboring Block Disparity Vector (DoNBDV) with virtual depth retrieval", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0090, October 2012. *Cited in Sec.* 4.3.1

[CWU+09] Y. Chen, Y.-K. Wang, K. Ugur, M. Hannuksela, J. Lainema, and M. Gabbouj, "The emerging MVC standard for 3D video services". *EURASIP Journal on Advances in Signal Processing*, 2009. *Cited in Sec.* 1.3.1, 1.3.2

[CYBH03] S. Cho, K. Yun, B. Bae, and Y. Hahm, "Disparity-compensated coding using MAC for stereoscopic video", in *IEEE International Conference on Consumer Electronics (ICCE)*, pp. 170–171, 2003. *Cited in Sec.* 1.3.2

[Dar09] I. Daribo, *Codage et rendu de séquence vidéo 3D; et applications à la télévision tridimensionnelle (TV 3D) et à la télévision à base de rendu de vidéos (FTV)*, Ph.D. thesis, Télécom ParisTech, 2009. *Cited in Sec.* 1.2.2

[DFS09] S. Denman, C. Fookes, and S. Sridharan, "Improved simultaneous computation of motion detection and optical flow for object tracking", in *Digital Image Computing: Techniques and Applications (DICTA)*, pp. 175–182, 2009. *Cited in Sec.* 6.1.1

[DGK+11] M. Domanski, T. Grajek, D. Karwowski, K. Klimaszewski, J. Konieczny, M. Kurc, A. Luczak, R. Ratajczak, J. Siast, O. Stankiewicz, J. Stankowski, and K. Wegner, "Technical description of Poznan University of Technology proposal for Call on 3D Video Coding Technology", ISO/IEC JTC1/SC29/WG11 MPEG2011/ M22697, November 2011. *Cited in Sec.* 1.3.1, 1

[Dod05] N. Dodgson, "Autostereoscopic 3D displays". *Computer*, vol. 38 (8), pp. 31–36, 2005. *Cited in Sec.* 1.1.2

[DTPP08] I. Daribo, C. Tillier, and B. Pesquet-Popescu, "Adaptive wavelet coding of the depth map for stereoscopic view synthesis", in *10th IEEE workshop on Multimedia Signal Processing (MMSP)*, pp. 413 –417, October 2008. *Cited in Sec.* 1.2.2

[Ebr12] T. Ebrahimi, "Towards 3D visual quality assessment for future multimedia", in *Key note presentation at CORESA conference*, 2012. *Cited in Sec.* 1.1.1

[EXP10] "Report on experimental framework for 3D video coding", ISO/IEC JTC1/SC29/WG11 MPEG2010/N11631, October 2010. *Cited in Sec.* 7.1.1

[FCSK02] C. Fehn, E. Cooke, O. Schreer, and P. Kauff, "3D analysis and image-based rendering for immersive TV applications". *Signal Processing: Image Communication*, vol. 17 (9), pp. 705 – 715, 2002. *Cited in Sec.* 1.2.1

[FLG13] M. S. Farid, M. Lucenteforte, and M. Grangetto, "Depth image based rendering with inverse mapping", in *15th IEEE workshop on Multimedia Signal Processing (MMSP)*, 2013. *Cited in Sec.* 7.1.2

[GARRM05] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed Video Coding". *Proceedings of the IEEE*, vol. 93 (1), pp. 71–83, Jan. 2005. *Cited in Sec.* 6.1.2

[GGG12a] T. Guionnet, L. Guillo, and C. Guillemot, "CE5.h: Merge candidate list for disparity compensated prediction", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0080, October 2012. *Cited in Sec.* 4.1.1

[GGG12b] ———, "CE5.h: Reducing the coding cost of merge index by dynamic merge index re-allocation", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0078, October 2012. *Cited in Sec.* 4.1.1

[GLL11] R. H. Gweon, Y.-L. Lee, and J. Lim, "Early termination of CU encoding to reduce HEVC complexity", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCTVC-F045, July 2011. *Cited in Sec.* 5.1

[H2605] "Advanced Video Coding for generic audiovisual services", ITU-T Recommendation H.264 and ISO/IEC 14496-10 AVC, 2005. *Cited in Sec.* 1.3.1

[HEV13] "High Efficiency Video Coding", ITU-T Recommendation H.265 and ISO/IEC 23008-2 HEVC, April 2013. *Cited in Sec.* 1

[HMK+10] W.-J. Han, J. Min, I.-K. Kim, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y. M. Hong, M.-S. Cheon, N. Shlyakhov, K. McCann, T. Davies, and J.-H. Park, "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20 (12), pp. 1709 –1720, December 2010. *Cited in Sec.* 2.3.1

[HOB+12] P. Helle, S. Oudin, B. Bross, D. Marpe, M. Bici, K. Ugur, J. Jung, G. Clare, and T. Wiegand, "Block merging for quadtree-based partitioning in HEVC". *IEEE Transactions on Circuits and Systems for Video Technology*, (99), 2012. *Cited in Sec.* 2.3.3

[HRS+13] M. Hannuksela, D. Rusanovskyy, W. Su, L. Chen, R. Li, P. Aflaki, D. Lan, M. Joachimiak, H. Li, and M. Gabbouj, "Multiview-Video-Plus-Depth coding based on the Advanced Video Coding standard". *IEEE Transactions on Image Processing*, vol. 22 (9), pp. 3449–3458, 2013. *Cited in Sec.* 1.3.1

[HS81] B. K. Horn and B. G. Schunck, "Determining optical flow". *Artificial Intelligence*, vol. 17, pp. 185 – 203, 1981. *Cited in Sec.* 6.1.1

[HSYSg12] J. Heo, E. Son, S. Yea, and Seocho-gu, "CE6.h: Region Boundary Chain coding for depth-map", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-A0070, July 2012. *Cited in Sec.* 2.5.2

[HYK03] Y. Hu, B. Yin, and D. Kong, "A new facial feature extraction method based on linear combination model", in *IEEE/WIC International Conference on Web Intelligence*, pp. 520–523, 2003. *Cited in Sec.* 6.1.1

[Jag12a] F. Jager, "3D-CE1.h: Results on View Synthesis Prediction", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0034, October 2012. *Cited in Sec.* 1.3.1

[Jag12b] ———, "3D-CE6.h: Simplified Depth Coding with an optional Depth Lookup Table", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-B0036, October 2012. *Cited in Sec.* 2.5.3

[JL12] J. Jung and J.-L. Lin, "AHG 10 report on inter component dependencies", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-B0010, October 2012. *Cited in Sec.* 2.5.4

[JM12a] J. Jung and E. Mora, "3D-CE3.h: Depth quadtree prediction for 3DHTM 4.1", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0068, October 2012. *Cited in Sec.* 5.6

[JM12b] ———, "Subjective test results on quadtree limitation and predictive coding", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0222, October 2012. *Cited in Sec.* 5.4.3

[KCCL96] C.-W. Ku, Y.-M. Chiu, L.-G. Chen, and Y.-P. Lee, "Building a pseudo object-oriented very low bit-rate video coding system from a modified optical flow motion estimation algorithm", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2064–2067 vol. 4, 1996. *Cited in Sec.* 6.1.1

[KD10] J. Konieczny and M. Domanski, "Depth-based inter-view prediction of motion vectors for improved multiview video coding", in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 2010. *Cited in Sec.* 1.3.1

[KFM10] B. Kamolrat, W. Fernando, and M. Mrak, "Adaptive motion-estimation-mode selection for depth video coding", in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 702 –705, March 2010. *Cited in Sec.* 1

[KHAF12] S. Khattak, R. Hamzaoui, S. Ahmad, and P. Frossard, "Fast encoding techniques for Multiview Video Coding". *Signal Processing: Image Communication*, 2012. *Cited in Sec.* 5.1

[KHL08] D.-Y. Kim, K.-H. Han, and Y.-L. Lee, "Adaptive Intra mode bit skip in Intra coding", in *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 446–449, 2008. *Cited in Sec.* 6.1.2

[KMW95] R. Krishnamurthy, P. Moulin, and J. Woods, "Optical flow techniques applied to video coding", in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. 570–573 vol.1, 1995. *Cited in Sec.* 6.1.1

[KOL+09] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map distortion analysis for view rendering and depth coding", in *16th IEEE International Conference on Image Processing (ICIP)*, pp. 721 –724, November 2009. *Cited in Sec.* 2

[KOL+10] ———, "Depth map coding with distortion estimation of rendered view". *Proceedings of the SPIE Visual Information Processing Community*, pp. 75430B–75430B–10, 2010. *Cited in Sec.* 1.2.2

[LCHL12] J.-L. Lin, Y.-W. Chen, Y.-W. Huang, and S. Lei, "3D-CE5.h: Pruning process for the inter-view candidate", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0086, October 2012. *Cited in Sec.* 4.1.1

[LCT+11] J.-L. Lin, Y.-W. Chen, Y.-P. Tsai, Y.-W. Huang, and S. Lei, "Motion vector coding techniques for HEVC", in *13th IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1 –6, October 2011. *Cited in Sec.* 4.1.1

[LE10] P.-J. Lee and Effendi, "Adaptive edge-oriented depth image smoothing approach for depth image based rendering", in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–5, 2010. *Cited in Sec.* 7.1.2

[LH96] M. Levoy and P. Hanrahan, "Light field rendering", in *Proceedings of the 23rd annual conference on computer graphics and interactive techniques*, SIGGRAPH '96, pp. 31–42, ACM, New York, NY, USA, 1996. *Cited in Sec.* 7.1

[Liu] C. Liu, "Optical flow Matlab/C++ code". *Cited in Sec.* 6.2, 6.4.1, 7.2.2, 7.3.1

[Liu09] ———, *Beyond pixels: exploring new representations and applications for motion analysis*, Ph.D. thesis, Massachusetts Institute of Technology, May 2009. *Cited in Sec.* 6.4.1

[LJPP08] G. Laroche, J. Jung, and B. Pesquet-Popescu, "RD optimized coding for motion vector predictor selection". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18 (9), pp. 1247–1257, 2008. *Cited in Sec.* 2.3.3

[LJPP10] ———, "Intra coding with prediction mode information inference". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20 (12), pp. 1786–1796, 2010. *Cited in Sec.* 6.1.2

[LJS+12] H. Liu, J. Jung, J. Sung, J. Jia, and S. Yea, "3D-CE2.h: Results of illumination compensation for inter-view prediction", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-B0045, October 2012. *Cited in Sec.* 2.4.3

[LK81] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in *Proceedings of the DARPA Image Understanding Workshop*, pp. 121–130, April 1981. *Cited in Sec.* 6.1.1

[LK13] J. Y. Lee and C. Kim, "3D-CE3.h related: Removal of depth quadtree prediction and simple early depth CU decision for fast encoding", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-D0121, April 2013. *Cited in Sec.* 5.1

[LOL11] J. Lee, B. T. Oh, and I. Lim, "Description of HEVC compatible 3D video coding technology by Samsung", ISO/IEC JTC1/SC29/WG11 MPEG2011/M22633, November 2011. *Cited in Sec.* 3

[LRL13] L. Lucas, Y. Remion, and C. Loscos, "Fondements", in *Vidéo 3D: capture, traitement et diffusion*, Chap. 1, pp. 23–41, Hermes, 2013. *Cited in Sec.* 1.1.1

[LSZ97] S. Lin, Y. Shi, and Y.-Q. Zhang, "An optical flow based motion compensation algorithm for very low bit-rate video coding", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, pp. 2869–2872 vol.4, 1997. *Cited in Sec.* 6.1.1

[MADB10] M. Moinard, I. Amonou, P. Duhamel, and P. Brault, "A set of template matching predictors for Intra video coding", in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 1422–1425, 2010. *Cited in Sec.* 6.1.2

[McM97] L. McMillan, *An image based approach to three-dimensional computer graphics*, Ph.D. thesis, University of North Carolina, Chapel Hill, NC, USA, 1997. *Cited in Sec.* 1.2.2

[MD08] M. Maitre and M. Do, "Joint encoding of the depth image based representation using shape-adaptive wavelets", in *15th IEEE International Conference on Image Processing (ICIP)*, pp. 1768–1771, 2008. *Cited in Sec.* 1.2.2

[MdWF06] Y. Morvan, P. H. N. de With, and D. Farin, "Platelet-based coding of depth maps for the transmission of multiview images", in *Proceedings of the SPIE, Stereoscopic Displays and Applications*, pp. 93–100, 2006. *Cited in Sec.* 1.2.2

[MHK+10] K. McCann, W.-J. Han, I.-K. Kim, J.-H. Min, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y.-M. Hong, M.-S. Cheon, and N. Shlyakhov, "Samsung's response to the Call for Proposals on video compression technology", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCTVC-A124, April 2010. *Cited in Sec.* 2.3.3

[MJCPP13a] E. Mora, J. Jung, M. Cagnazzo, and B. Pesquet-Popescu, "Depth video coding based on Intra mode inheritance from texture". *APSIPA Transactions on Signal and Information Processing*, 2013. *Cited in Sec.* 3.6

[MJCPP13b] ———, "Initialization, limitation and predictive coding of the depth and texture quadtree in 3D-HEVC". *IEEE Transactions on Circuits and Systems for Video Technology*, 2013. *Cited in Sec.* 5.6

[MJCPP13c] ———, "Modification of the Merge candidate list for dependent views in 3D-HEVC", in *IEEE International Conference on Image Processing (ICIP)*, 2013. *Cited in Sec.* 4.5

[MJPPC12] E. Mora, J. Jung, B. Pesquet-Popescu, and M. Cagnazzo, "3D-CE5.h related: Modification of the Merge candidate list for dependant views in 3DV-HTM", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B0069, October 2012. *Cited in Sec.* 4.2.3

[MJPPC13] ———, "Modification of the disparity vector derivation process in 3D-HEVC", in *IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 206–211, 2013. *Cited in Sec.* 4.5

[MMCPP09] W. Miled, T. Maugey, M. Cagnazzo, and B. Pesquet-Popescu, "Image interpolation with dense disparity estimation in multiview Distributed Video Coding", in *3rd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pp. 1–6, 2009. *Cited in Sec.* 6.1.1

[MMS+08] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P. de With, and T. Wiegand, "The effect of depth compression on multiview rendering quality", in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pp. 245 –248, May 2008. *Cited in Sec.* 1

[MPE98] "Stereoscopic television MPEG-2 Multiview Profile", ITU-T Recommendation H.262 and ISO/CEI 13818-2, 1998. *Cited in Sec.* 1.3.2

[MVJ+13] E. G. Mora, G. Valensize, J. Jung, B. Pesquet-Popescu, M. Cagnazzo, and F. Dufaux, "Depth video coding technologies", in *Emerging Technologies for 3D Video*, Chap. 7, pp. 121–137, Wiley, 2013. *Cited in Sec.* 1.2.2

[MWG05] D. Marpe, T. Wiegand, and S. Gordon, "H.264/MPEG4-AVC fidelity range extensions: tools, profiles, performance, and application areas", in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, pp. I–593–6, 2005. *Cited in Sec.* 1.3.2

[Ohm13] J.-R. Ohm, "Overview of 3D video coding standardization", in *Proceedings of 3DSA2013, Keynote speech 2*, 2013. *Cited in Sec.* 1.3.2

[Ols08] R. Olsson, *Synthesis, coding and evaluation of 3D images based on integral imaging*, Ph.D. thesis, Mid Sweden University, 2008. *Cited in Sec.* 1.1.2

[PHL09] J.-H. Park, K. Hong, and B. Lee, "Recent progress in three-dimensional information processing based on integral imaging". *Optics Society of America*, vol. 48, 2009. *Cited in Sec.* 1.1.2

[Pur13] A. Purica, *View synthesis techniques in Multiview Video plus Depth coding*, Master's thesis, University "Politehnica" of Bucharest, July 2013. *Cited in Sec.* 1

[RCZS13] D. Rusanovsky, F.-C. Chen, L. Zhang, and T. Suzuki, "3D-AVC test model 7", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-E1003, July 2013. *Cited in Sec.* 1.3.1, 1

[RH11] D. Rusanovsky and M. Hannuksela, "Description of Nokia's response to MPEG 3DV Call for Proposals on 3DV video coding technologies", ISO/IEC JTC1/SC29/WG11 MPEG2011/M22552, November 2011. *Cited in Sec.* 1

[RHFL10] S. Reichelt, R. Haussler, G. Fatterer, and N. Leister, "Depth cues in human visual perception and their realization in 3D displays". *Proceedings of the SPIE Three-Dimensional Imaging, Visualization, and Display*, pp. 76900B–76900B–12, 2010. *Cited in Sec.* 1.1.1

[RMV13] D. Rusanovsky, K. Muller, and A. Vetro, "Common Test Conditions of 3DV Core Experiments", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-D1100, April 2013. *Cited in Sec.* 1.3.2, 3.3.1, 4.2.1, 4.4.1, 5.4.1, 6.2, 6.4.1, 7.3.1

[RSP10] A. Ramadass, M. Suk, and B. Prabhakaran, "Feature extraction method for video based human action recognitions: extended optical flow algorithm", in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 1106–1109, 2010. *Cited in Sec.* 6.1.1

[SBB11] H. Schwarz, C. Bartnik, and S. Bosse, "Description of 3D video technology proposal by Fraunhofer HHI", ISO/IEC JTC1/SC29/WG11 MPEG2011/M22571, November 2011. *Cited in Sec.* 1

[SCS05] C. Slinger, C. Cameron, and M. Stanley, "Computer-Generated Holography as a generic display technology". *Computer*, vol. 38 (8), pp. 46–53, 2005. *Cited in Sec.* 1.1.2

[SK00] H. Shum and S. B. Kang, "Review of image-based rendering techniques". *SPIE Visual Communications and Image Processing*, vol. 4067, pp. 2–13, 2000. *Cited in Sec.* 7.1

[SK10] S. Shimizu and H. Kimata, "Improved view synthesis prediction using decoder-side motion derivation for multiview video coding", in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 2010. *Cited in Sec.* 1.3.1

[SKSM11] S. Shimizu, H. Kimata, S. Sugimoto, and N. Matsuura, "Decoder-side macroblock information derivation for efficient multiview video plus depth map coding", in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 2011. *Cited in Sec.* 6.1.2

[SKY12] J. Sung, M. Koo, and S. Yea, "3D-CE5.h: Simplification of disparity vector derivation for HEVC-based 3D video coding", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-A0126, July 2012. *Cited in Sec.* 2.4.1

[SLA+11] L. Shen, Z. Liu, P. An, R. Ma, and Z. Zhang, "Low-complexity mode decision for MVC". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21 (6), pp. 837–843, 2011. *Cited in Sec.* 5.1

[SLP13] V. Spruyt, A. Ledda, and W. Philips, "Sparse optical flow regularization for real-time visual tracking", in *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2013. *Cited in Sec.* 6.1.1

[SLZ+12] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders". *IEEE Transactions on Multimedia*, (99), 2012. *Cited in Sec.* 5.1

[SOHW12] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22 (12), pp. 1649–1668, 2012. *Cited in Sec.* 2.3.3

[SPW+10] J. Seo, D. Park, H.-C. Wey, S. Lee, and K. Sohn, "Motion information sharing mode for depth video coding", in *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1 –4, June 2010. *Cited in Sec.* 2

[SSO09] A. Sanchez, G. Shen, and A. Ortega, "Edge-preserving depth-map coding using graph-based wavelets", in *Conference Record of the 43rd Asilomar Conference on Signals, Systems and Computers*, pp. 578–582, 2009. *Cited in Sec.* 1.2.2

[SZL12] L. Shen, Z. Zhang, and Z. Liu, "Inter mode selection for depth map coding in 3D video". *IEEE Transactions on Consumer Electronics*, vol. 58 (3), pp. 926–931, 2012. *Cited in Sec.* 5.1

[TAZ+04] W. J. Tam, G. Alain, L. Zhang, T. Martin, and R. Renaud, "Smoothing depth maps for improved steroscopic image quality". *Proceedings of the SPIE Three-Dimensional TV, Video and Display III*, pp. 162–172, 2004. *Cited in Sec.* 1.2.2

[TBS06] T. Tan, C. Boon, and Y. Suzuki, "Intra prediction by template matching", in *IEEE International Conference on Image Processing*, pp. 1693–1696, 2006. *Cited in Sec.* 6.1.2

[Teca] G. Tech, "HTM-0.3 software", Available: https://hevc.hhi.fraunhofer.de/svn/. *Cited in Sec.* 3.3.1

[Tecb] ———, "HTM-4.0 software", Available: https://hevc.hhi.fraunhofer.de/svn/. *Cited in Sec.* 5.4.1

[Tecc] ———, "HTM-4.0 software including FCO", Available: https://hevc.hhi.fraunhofer.de/svn/. *Cited in Sec.* 5.4.1

[Tecd] ———, "HTM-4.1 software", Available: https://hevc.hhi.fraunhofer.de/svn/. *Cited in Sec.* 4.2.1

[Tece] ———, "HTM-5.0.1 software", Available: https://hevc.hhi.fraunhofer.de/svn/. *Cited in Sec.* 4.4.1

[TLD07] Z. Tauber, Z.-N. Li, and M. Drew, "Review and preview: disocclusion by inpainting for Image-Based Rendering". *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37 (4), pp. 527–540, 2007. *Cited in Sec.* 1.2.2

[TLTY12] H. L. Tan, F. Liu, Y. H. Tan, and C. Yeo, "On fast coding tree block and mode decision for High-Efficiency Video Coding (HEVC)", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 825 –828, March 2012. *Cited in Sec.* 5.1

[TSMW12] G. Tech, H. Schwarz, K. Muller, and T. Wiegand, "3D video coding using the Synthesized View Distortion Change", in *Picture Coding Symposium (PCS)*, pp. 25 –28, May 2012. *Cited in Sec.* 2.6

[TWCY12] G. Tech, K. Wegner, Y. Chen, and S. Yea, "3D-HEVC test model 2", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCT3V-B1005, October 2012. *Cited in Sec.* 4.3.1

[TWCY13] ———, "3D-HEVC test model 3", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-C1005, January 2013. *Cited in Sec.* 2.4.1

[TZV13] D. Tian, F. Zou, and A. Vetro, "CE1.h: Backward View Synthesis Prediction using neighbouring blocks", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-C0152, January 2013. *Cited in Sec.* 2.4.4

[VCG+12] G. Valenzise, G. Cheung, R. Galvao, M. Cagnazzo, B. Pesquet-Popescu, and A. Ortega, "Motion prediction of depth video for depth-image-based rendering using don't care regions", in *Picture Coding Symposium (PCS)*, pp. 93–96, 2012. *Cited in Sec.* 3

[VM13] A. Vetro and K. Muller, "Depth-based 3D video formats and coding technology", in *Emerging Technologies for 3D Video*, Chap. 8, pp. 139–161, Wiley, 2013. *Cited in Sec.* 1.3.2, 2.1

[WLS+11] L. Wang, J. Liu, J. Sun, Y. Ren, W. Liu, and Y. Gao, "Virtual view synthesis without preprocessing depth image for depth image based rendering", in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pp. 1–4, 2011. *Cited in Sec.* 7.1.2

[WZ11] Z. Wang and J. Zhou, "A novel approach for depth image based rendering, based on non-linear transformation of depth values", in *International Conference on Image Analysis and Signal Processing (IASP)*, pp. 138–142, 2011. *Cited in Sec.* 7.1.2

[YI12] T. Yamamoto and T. Ikai, "Merge candidate refinement for uni-predictive block", ITU-T SG16 WP3 & ISO/IEC JTC1/SC29/WG11 JCTVC-I0293, May 2012. *Cited in Sec.* 4.1.1

[YKO10] F. Yaras, H. Kang, and L. Onural, "State of the art in holographic displays: a survey". *Journal of Display Technology*, vol. 6 (10), pp. 443–454, 2010. *Cited in Sec.* 1.1.2

[ZCK12] L. Zhang, Y. Chen, and M. Karczewicz, "3D-CE5.h related: Disparity vector derivation for multiview video and 3DV", ISO/IEC JTC1/SC29/WG11 MPEG2012/m24937, April 2012. *Cited in Sec.* 2.4.1

[ZCL+13] N. Zhang, Y.-W. Chen, J.-L. Lin, J. An, K. Zhang, S. Lei, S. Ma, D. Zhao, and W. Gao, "3D-CE3.h related: Fast encoder decision for texture coding", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-E0173, July 2013. *Cited in Sec.* 5.5.2

[Zha10] H.-Y. Zhang, "Multiple moving objects detection and tracking based on optical flow in polar-log images", in *International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 3, pp. 1577–1582, 2010. *Cited in Sec.* 6.1.1

[ZJYH09] Y. Zhang, G. Y. Jiang, M. Yu, and Y. S. Ho, "Adaptive multiview video coding scheme based on spatiotemporal correlation analyses". *ETRI Journal*, vol. 31 (2), April 2009. *Cited in Sec.* 4.1.2

[ZT05] L. Zhang and W. J. Tam, "Stereoscopic image generation based on depth images for 3D TV". *IEEE Transactions on Broadcasting*, vol. 51 (2), pp. 191–199, 2005. *Cited in Sec.* 1.2.2

[ZTWY13] L. Zhang, G. Tech, K. Wegner, and S. Yea, "3D-HEVC test model 5", ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 JCT3V-E1005, July 2013. *Cited in Sec.* 2.5.3, 7.1.1

[ZWPSxZy07] L. Zhan-Wei, A. Ping, L. Su-xing, and Z. Zhao-yang, "Arbitrary view generation based on DIBR", in *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 168–171, 2007. *Cited in Sec.* 1.2.2

Codage multi-vues multi-profondeur pour de nouveaux services multimédia

Elie Gabriel MORA

RESUME : Les travaux effectués durant cette thèse de doctorat ont pour but d'augmenter l'efficacité de codage dans 3D-HEVC. Nous proposons des approches conventionnelles orientées vers la normalisation vidéo, ainsi que des approches en rupture basées sur le flot optique.

En approches conventionnelles, nous proposons une méthode qui prédit les modes Intra de profondeur avec ceux de texture. L'héritage est conditionné par un critère qui mesure le degré de similitude entre les deux modes. Ensuite, nous proposons deux méthodes pour améliorer la prédiction inter-vue du mouvement dans 3D-HEVC. La première ajoute un vecteur de disparité comme candidat inter-vue dans la liste des candidats du Merge, et la seconde modifie le processus de dérivation de ce vecteur. Finalement, un outil de codage inter-composantes est proposé, où le lien entre les arbres quaternaires de texture et de profondeur est exploité pour réduire le temps d'encodage et le débit, à travers un codage conjoint des deux arbres.

Dans la catégorie des approches en rupture, nous proposons deux méthodes basées sur l'estimation de champs denses de vecteurs de mouvement en utilisant le flot optique. La première calcule un champ au niveau d'une vue de base reconstruite, puis l'extrapole au niveau d'une vue dépendante, où il est hérité par les unités de prédiction en tant que candidat dense du Merge. La deuxième méthode améliore la synthèse de vues : quatre champs sont calculés au niveau de deux vues de référence en utilisant deux références temporelles. Ils sont ensuite extrapolés au niveau d'une vue synthétisée et corrigés en utilisant une contrainte épipolaire. Les quatre prédictions correspondantes sont ensuite combinées. Les deux méthodes apportent des gains de codage significatifs, qui confirment le potentiel de ces solutions innovantes.

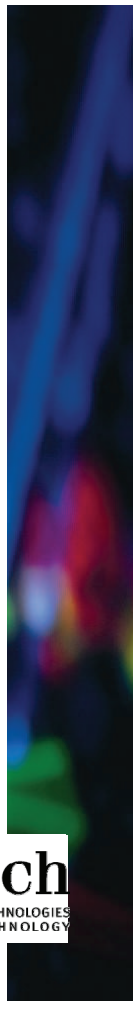
MOTS-CLEFS : 3D-HEVC, Flot optique, Synthèse de vues, Liste des candidats du Merge, Initialisation et limitation d'un arbre quaternaire, Vecteur de disparité, Vecteur de mouvement, Mode Intra.

ABSTRACT : This PhD. thesis deals with improving the coding efficiency in 3D-HEVC. We propose both constrained approaches aimed towards standardization, and also more innovative approaches based on optical flow.

In the constrained approaches category, we first propose a method that predicts the depth Intra modes using the ones of the texture. The inheritance is driven by a criterion measuring how much the two are expected to match. Second, we propose two simple ways to improve inter-view motion prediction in 3D-HEVC. The first adds an inter-view disparity vector candidate in the Merge list and the second modifies the derivation process of this disparity vector. Third, an inter-component tool is proposed where the link between the texture and depth quadtree structures is exploited to save both runtime and bits through a joint coding of the quadtrees.

In the more innovative approaches category, we propose two methods that are based on a dense motion vector field estimation using optical flow. The first computes such a field on a reconstructed base view. It is then warped at the level of a dependent view where it is inserted as a dense candidate in the Merge list of prediction units in that view. The second method improves the view synthesis process : four fields are computed at the level of the left and right reference views using a past and a future temporal reference. These are then warped at the level of the synthesized view and corrected using an epipolar constraint. The four corresponding predictions are then blended together. Both methods bring significant coding gains which confirm the potential of such innovative solutions.

KEY-WORDS : 3D-HEVC, Optical flow, View synthesis, Merge candidate list, Quadtree initialization and limitation, Disparity vector, Motion vector, Intra mode.



TELECOM ParisTech

EDITE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY