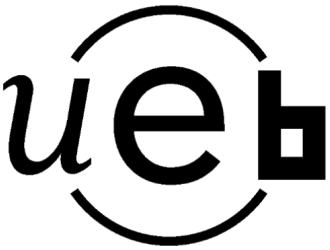
Layered Depth Images for Multi-View Coding

Vincent Jantet

ANNÉE 2012



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

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal et Télécommunications

Ecole doctorale Matisse

présentée par

Vincent Jantet

préparée à l'unité de recherche TEMICS – UMR 6074 IRISA / INRIA
TraitemEnt, Modélisation d'Images & CommunicationS
Université de Rennes 1

Layered Depth Images for Multi-View Coding

Thèse soutenue à Rennes
le 23 novembre 2012

devant le jury composé de :

Béatrice PESQUET
Professeur, Télécom ParisTech / rapporteur

Ferran MARQUÉS
Professeur, Université Polytechnique de Catalogne, Espagne / rapporteur

Edmond BOYER
Directeur de Recherche, INRIA Grenoble / examinateur

Aljoscha SMOLIC
Ingénieur de Recherche, Disney Research, Zurich, Suisse / examinateur

Christine GUILLEMOT
Directeur de Recherche, INRIA Rennes / directrice de thèse

Luce MORIN
Professeur, INSA-Rennes / co-directrice de thèse

Contents

Glossary

3DAV: 3D Audio-Visual

3DTV: 3D TeleVision

AVC: Advanced Video Coding

DCT: Discrete Cosine Transform

DERS: Depth Estimation Reference Software

DI: Depth Image (color map + depth map)

DIBR: Depth-Image-Based Rendering

DV: Depth Video (DI sequence)

GBR: Geometry-Based Rendering

I-LDI: Incremental Layered Depth Image

IBR: Image-Based Rendering

IEC: International Electrotechnical Commission

ISO: International Organization for Standardization

JVT: Joint Video Team

LDI: Layered Depth Image

LDV: Layered Depth Video

MPEG: Moving Picture Experts Group

MSE: Mean Square Error

MVC: Multi-view Video Coding

MVD: Multiple Video-plus-Depth sequence

MVV: Multi-View Video

O-LDI: Object-based Layered Depth Image

PSNR: Peak Signal to Noise Ratio

SADERS: Semi-Automatic Depth Estimation Reference Software

SSIM: Structural SIMilarity index

VCEG: Video Coding Experts Group

VDTM: View-Dependant Texture Mapping

Résumé en français: Compression multi-vues par representation LDI

La banalisation des dispositifs stéréoscopiques impose l'ajout d'une troisième dimension aux données multimédias classiques. L'acquisition d'une scène par plusieurs caméras synchronisées (MVV pour *Multi-View Video*) est un des procédés permettant l'estimation de la géométrie de la scène. La connaissance de cette géométrie est indispensable a l'ajout de fonctionnalités 3D comme l'affichage en relief (3DTV) ou la sélection de points de vue (FVV pour *Free Viewpoint Video*).

Les informations géométriques peuvent être de différentes formes (maillage, fonction plénoptique, polygones, cartes de profondeur, ...). Chacune de ces formes constitue un format de données indépendant, et est associée à un algorithme de synthèse de vues approprié. En fonction du type d'informations géométriques qu'ils utilisent, ces algorithmes de rendu peuvent être classés selon deux catégories: les rendus basée géométrie et les rendus basée images.

Les Rendus Basés Géométrie (GBR) utilisent un modèle 3D évolué et détaillé. Ils sont idéaux pour le rendu de scènes synthétiques, mais peu adaptés à des données réelles, où le modèle géométrique est difficile à obtenir.

Les Rendus Basés Images (IBR) [Shum & Kang, 2000, Shum et al., 2007] nécessitent une géométrie moins détaillée (souvent des cartes de profondeur), mais ont besoin d'un grand nombre de textures en entrée. Les données d'entrée sont donc moins compactes, mais les images synthétisées sont de meilleure qualité et plus réalistes.

Les LDI (pour *Layered Depth Images*) [Shade et al., 1998, Yoon et al., 2007] constituent l'une de ces approches IBR. La géométrie est modélisée par une image en plusieurs couches, où chaque couche est accompagnée de la carte de profondeur correspondante. Chaque point de la LDI est donc associé à un ensemble de pixels, constitués chacun d'une couleur et d'une profondeur. Ces pixels représentent l'ensemble des objets (visibles ou masqués dans la scène) traversés par un rayon reliant le centre optique de la caméra au point image considéré. Cette représentation selon un seul point de vue, réduit efficacement les redondances inter-vues, tout en permettant une synthèse de vue réaliste pour des points de vue éloignés du point de vue de référence. C'est cette représentation LDI qui sera utilisée dans cette étude, pour compresser le format multi-vues, en éliminant la plupart des redondances.

Ce résumé de mémoire est organisé en trois sections, partant de la construction des LDI, passant par leur compression, et allant jusqu'à leur utilisation pour synthétiser des vues virtuelles.

La section 1 décrit les différents algorithmes de construction de LDI, à partir de séquences multi-vues [Cheng et al., 2007a]. Elle commence par introduire les équations de projection, indispensable à la bonne compréhension des mécanismes de capture d'une scène par les caméras. Ces équations sont ensuite utilisées pour détailler deux méthodes de construction des couches des LDI. La première approche est classique, mais ne permet pas d'éliminer toutes les corrélations entre les couches. C'est pourquoi une seconde méthode est proposée, basée sur une construction incrémentale, qui permet d'éliminer beaucoup plus de corrélations entre les couches. Une restructuration des couches est finalement proposée, pour regrouper les pixels d'un même objet sur une même couche, et ainsi améliorer la prédiction spatiale lors de la compression, tout en réduisant les artefacts lors de la synthèse de vues.

La section 2 traite des problèmes de compression de cette représentation LDI [Yoon et al., 2006a, Duan & Li, 2003]. L'approche proposée repose sur le codeur MVC (pour *Multi-View Coding* [Merkle et al., 2007]), qui est initialement conçu pour s'appliquer directement sur des vidéos multi-vues. L'innovation consiste ici à détourner le codeur de son cadre usuel, pour l'utiliser sur les différentes couches des LDI qui sont déjà dé-corrélées, et ainsi mieux tirer parti des capacités du codeur.

La section 3 aborde les difficultés de la synthèse de vues virtuelles à partir de LDI. Les différents artefacts usuels sont détaillés, à savoir les craquelures, les découvrements, et les contours fantômes. Une nouvelle méthode de projection est ensuite proposée, la JPF pour *Joint Projection Filling*, permettant d'éliminer les craquelures et les contours fantômes, en évitant le surcoût d'un post-traitement. Les découvrements sont ensuite comblés par une solution d'*inpainting* innovante, qui tire parti des informations de profondeur pour sélectionner les bons patchs de texture à dupliquer.

Quelques résultats de mise en œuvre des LDI sont commentés dans la section 4, ce qui permet de valider les trois étapes mises à la chaîne: Construction, Compression puis Rendu.

1 Construction des LDI

Pour construire la LDI représentative d'une scène réelle, la plupart des méthodes s'appuient en entrée sur des vidéos multi-vues [Cheng et al., 2007a], où chaque vue est accompagnée de sa carte de profondeur (MVD pour *Multi-View plus Depth*). Le logiciel de référence qui permet cette construction (MVD2LDV du groupe MPEG [Tanimoto et al., 2008]) présente un certain nombre de limitations. Le nombre de vues en entrée est par exemple limité à trois, et le nombre de couches générées est limité à deux. La limitation la plus restrictive reste cependant la disposition des caméras d'entrée, qui doivent être parfaitement rectifiées. Cette hypothèse simplifie les équations de projection (présentées dans la section 1.1), mais impose aux caméras d'être alignées, et d'avoir des objectifs parfaits, qui n'ajoutent aucune déformation intrinsèque aux images. Cette propriété, difficile à avoir dès l'acquisition, est généralement obtenue par un traitement qui transforme les vues capturées.

Deux méthodes de construction de LDI sont donc proposées pour remplacer MDV2LDV, et remédier à ses limitations. La première, dite "construction naïve", est présentée dans la

section 1.2. Simple à mettre en œuvre, elle présente le défaut de générer un grand nombre de couches, encore partiellement corrélées les unes aux autres. La seconde approche, appelée I-LDI, est détaillée dans la section 1.3. Elle repose sur une construction incrémentale pour éliminer les corrélations. Les couches générées sont moins nombreuses, et moins remplies, ce qui facilite leurs compression ultérieure. Ces deux approches reposent essentiellement sur les équations de projection, détaillées dans la section 1.1.

Finalement, une proposition de réorganisation des couches est évoquée dans la section 1.4, basée sur une segmentation de la scène par objets. Les pixels représentant un même objet de la scène sont regroupés sur une même couche, ce qui augmente l'efficacité des prédictions spatiales lors de la compression, et réduit les artefacts lors de la synthèse de vues.

1.1 Équations de projection

Dans une image, chaque pixel $p = (u, v, 1)$ en coordonnées homogènes, est la projection perspective d'un point 3D $M = (X, Y, Z)$ selon les équations (1). Ces équations nécessitent la connaissance des paramètres internes à la caméra K, de la matrice de rotation R_C et de la position t_0 de la caméra par rapport au repère global.

$$p = \omega K R_C^{-1}(M - t_0) \tag{1}$$

avec ω le coefficient de normalisation en coordonnées homogènes.

K est une matrice 3×3 contenant les paramètres intrinsèques de la caméra (taille des pixels, longueur de la focale, position du centre optique). R_C est la matrice de rotation 3×3 représentant l'orientation de la caméra par rapport au repère global. t_0 est le vecteur 1×3 indiquant la position de la caméra dans le repère global. L'ensemble de ces paramètres sont détaillés dans le manuscrit à la section 3.1.

Les équations inverses (2) permettent de retrouver la position 3D d'un point M, à partir du pixel p le représentant, et de sa coordonnée de profondeur Z dans le repère camera. Des explications détaillées permettant d'obtenir cette équation est proposé à la section 3.1.3.

$$M = Z R_C K^{-1} p + t_0 \tag{2}$$

Ces équations sont utilisées par les techniques de synthèse de vues basées sur des cartes de profondeur [Yoon et al., 2007, Cheng et al., 2007a]. Elles génèrent des effets de craquelures sur les textures et des zones de découvrements qui seront détaillés dans la section 3. Les paramètres K, R_C et t_0 peuvent être estimés à partir des vidéos MVD par des techniques de calibration. Nous supposerons dans la suite que ces paramètres caméra sont connus.

1.2 Construction naïve des LDI

Une LDI est un concentré d'information selon un même point de vue. La méthode de construction naïve consiste donc à projeter toute l'information présente dans la MVD selon un unique point de vue de référence, puis à fusionner ces informations pour les organiser par couches.

Le point de vue de référence peut être quelconque, mais l'on choisit généralement le point de vue central parmi les caméras d'acquisition. Les vues sont considérées comme des ensembles de pixels, chacun ayant une couleur et une profondeur. Tous les pixels de chaque vue sont projetés selon le point de vue de référence. Lorsque deux pixels p_1 et p_2, des vues respectives V_1 et V_2, sont les projection du même point 3D M, alors ils seront nécessairement projetés sur le même pixel p selon le point de vue de référence, et à la même profondeur Z. Pour éviter cette redondance, un test sur la profondeur est utilisé et les pixels de profondeurs voisines (seuil Δ_Z) sont fusionnés. Cette construction simpliste est schématisée dans la figure 1.

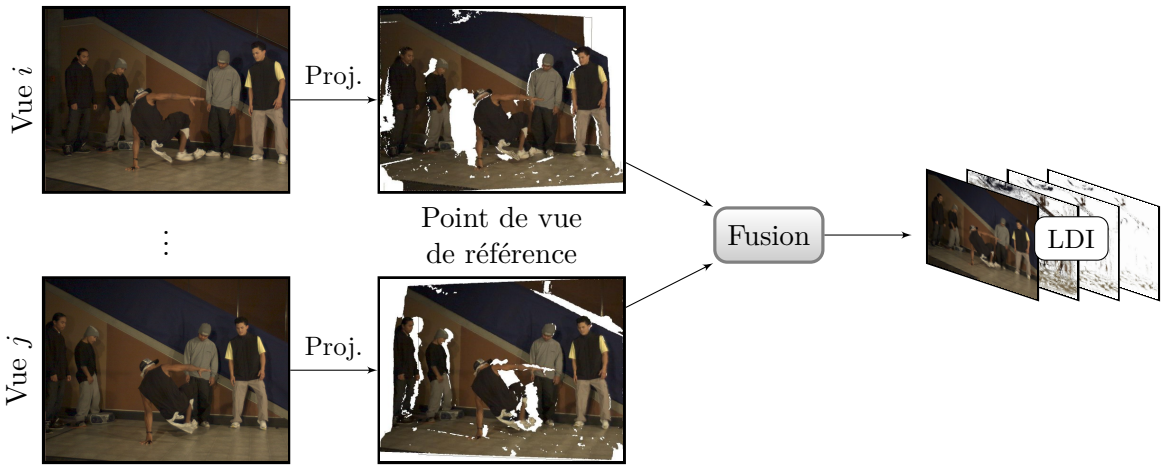


Figure 1: Schéma de construction naïve des LDI.

Les trois premières couches d'une LDI sont présentées sur la figure 2. Chaque couche contient l'ensemble des pixels les plus proches de la caméra, qui ne soient pas déjà dans une des couches précédentes. La première couche, contenant tous les pixels visibles depuis le point de vue de référence, est une image classique. Les autres couches, ne contiennent que les pixels masqués par un objet de la scène, sont partiellement vides.

(a) 1^{ere} couche (b) 2^{eme} couche (c) 3^{eme} couche

Figure 2: Premières couches d'une LDI naïve (Ref. = Vue 4) générée en utilisant les 8 caméras d'acquisition. ($\Delta_Z = 0.1$)

Le seuil Δ_Z est critique pour l'élimination des redondances, mais difficile à déterminer à l'avance. Une autre méthode de construction est donc proposée, utilisant un critère totalement différent.

1.3 Construction incrémentale des LDI

Pour éliminer les corrélations entre les vues, une autre construction est proposée, désignée par I-LDI, pour Incrémentale-LDI. Basée sur la méthode d'extraction d'information résiduelle [Müller et al., 2008a], cette construction permet de réduire le nombre de couches et leur taux de remplissage. L'algorithme peut se décomposer de la façon suivante:

Initialisation : Le point de vue de référence est fixé arbitrairement, il n'a pas besoin d'être un des points de vue des caméras d'acquisition. La I-LDI est initialisée vide, elle ne contient aucun pixel.

Itérations : La I-LDI est projetée selon le point de vue d'une des caméras d'acquisition, ce qui génère des zones de découvrement. Cette image virtuelle est comparée à l'image acquise de ce point de vue. L'ensemble des pixels permettant de combler les zones de découvrement est appelé information résiduelle. Seule cette information résiduelle est re-projetée selon le point de vue de référence pour être ajoutée à la I-LDI. Cette étape est réitérée pour chacune des caméras d'acquisition, ajoutant de plus en plus de pixels dans la I-LDI.

Finalisation : L'ensemble des pixels est réparti par couches, chaque couche contenant les pixels les plus proches, non déjà contenus dans une couche précédente. On remarque que la première couche est nécessairement l'image 2D+Z observée selon le point de vue de référence.

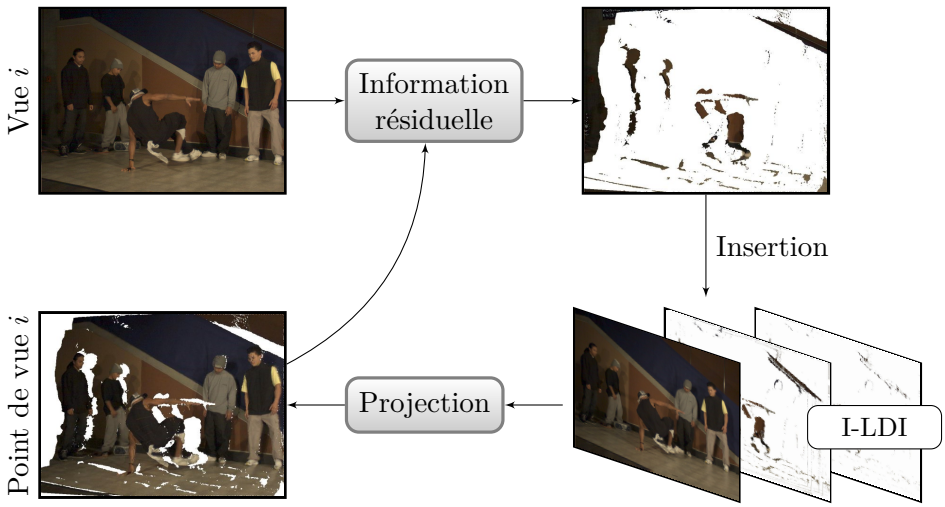


Figure 3: Extraction de l'information résiduelle de la vue i, lors de la construction incrémentale de LDI.

Les trois premières couches d'une I-LDI sont présentées dans la figure 4. La seconde couche ne contient que les textures des zones réellement occultées dans la scène.

La figure 5 présente le taux de remplissage des cinq premières couches d'une LDI et d'une I-LDI. Dans les deux cas, la première couche est complète. Dans une LDI, le nombre de pixels cumulé de toutes les couches supplémentaires représente plus de 50% du nombre de pixels présents dans la première couche. Ce pourcentage est réduit à moins de 10% par la construction

(a) 1^{ere} couche (b) 2^{eme} couche (c) 3^{eme} couche

Figure 4: Premières couches d'une I-LDI (Ref. = Vue 4), générée en utilisant les 8 caméras d'acquisition dans un ordre B-hiérarchique: (4; 2; 6; 1; 3; 5; 7; 0).

incrémentale. Dans ce cas, les couches à partir de la 3^{eme} sont pratiquement vides, et sont en pratique ignorées.

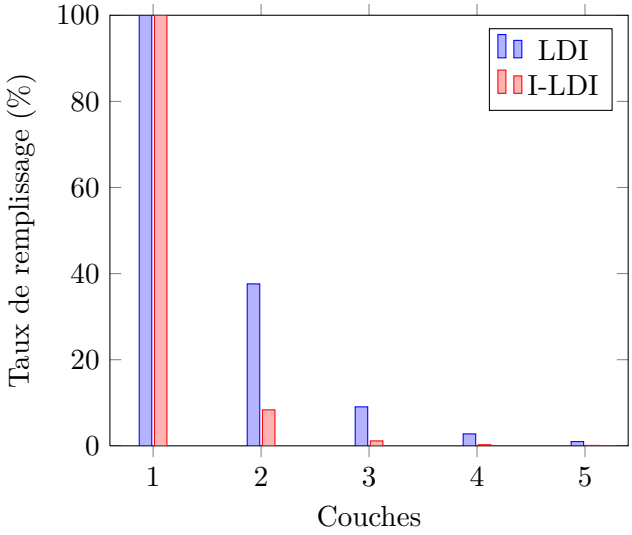


Figure 5: Taux de remplissage moyen des couches d'une LDI et d'une I-LDI

1.4 Segmentation par couche-object

Les LDI générées par les méthodes vues précédemment sont organisées par visibilité. C'est à dire que la première couche contient tous les pixels visibles depuis le point de vue de référence, et que les couches suivantes ne contiennent que les pixels restants, présents dans la scène mais masqués par un objet plus proche. Par cette organisation, les pixels d'un même objet physique peuvent être dispersés sur plusieurs couches, alors que les pixels adjacents sur une même couche peuvent représenter des objets différents aux textures différentes. La Figure 6 présente les deux premières couches d'une I-DLI, où l'on constate que les contours et la texture d'arrière-plan sont présents dans chacune des couches, et coûteux à coder.

Pour améliorer la prédiction spatiale au sein d'une couche et éliminer un grand nombre de contours inutiles, les pixels représentant un même objet sont réorganisés sur une seule couche. Une méthode de réorganisation est proposée, basée sur un algorithme de croissance de région, qui permet une bonne segmentation des objets présents dans la scène. La couche d'arrière-plan

(a) 1ère couche. (b) 2eme couche.

Figure 6: Les deux premières couches d'une LDI classique, organisée par visibilité.

est finalement complétée par une méthode d'inpainting, qui permet d'extrapoler les pixels qui ne sont visibles depuis aucune des caméras d'acquisition. Le résultat de ce traitement, présenté sur la Figure 7, montre une couche d'arrière-plan continue, et une couche d'avant-plan contenant les objets. Cette couche d'arrière-plan a l'avantage d'être pratiquement statique au cours du temps, ce qui facilite sa compression, alors que seule la couche d'avant-plan contient les objets dynamiques, aux contours détaillés.

(a) Couche d'avant-plan. (b) Couche d'arrière-plan.

Figure 7: Les deux couches d'une LDI-Object.

Dans la suite, le terme LDI sera utilisé pour désigner une image par couche, sans distinguer l'algorithme qui a permis de la synthétiser, à savoir la construction naïve, la construction incrémentale, ou la segmentation par objets.

2 Compression des LDI

Les vidéos MVD représentent un important volume de données, mais très fortement corrélées. Les corrélations inter-vues viennent s'ajouter aux corrélations temporelles, déjà présentes dans une vidéo classique. Pour compresser ces vidéos multi-vues, une des propositions mise en œuvre par MPEG [Tanimoto et al., 2008] est le codeur MVC, présenté dans la section 2.1. Ce codeur présente certaines limitations dans le cadre multi-vues, c'est pourquoi la section 2.2 présente une utilisation détournée de ce codeur, adaptée aux couches d'une LDI.

2.1 Codeur MVC

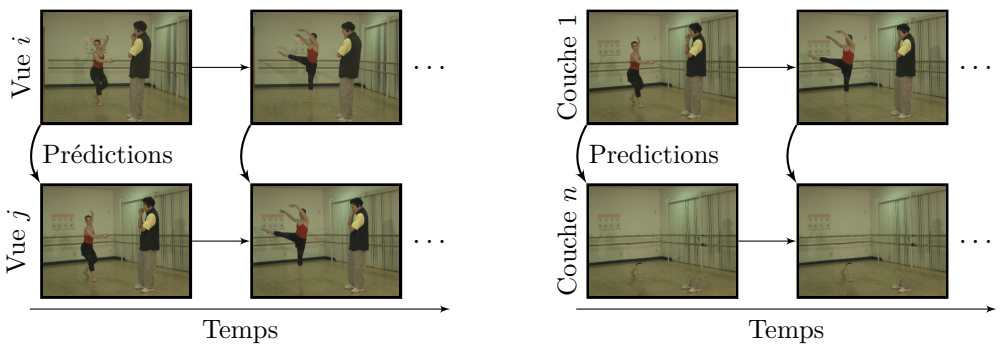
Dans un codeur vidéo 2D comme H.264/AVC, quelques images clés servent de référence, et sont encodées indépendamment des autres. Les images restantes sont prédites à partir de ces images clés, par l'utilisation d'un champ de mouvement, codé et transmis avec les images clés. L'erreur de prédiction est également encodée, pour permettre d'être corrigée par le décodeur et d'améliorer la qualité. Ce principe a été étendu aux séquences multi-vues par le codeur MVC (pour *Multi-View Coding*, une extension de H.264/AVC). L'une des vues est choisie comme vue de référence, et est encodée comme une vidéo classique. Chaque image des vues supplémentaires est alors prédite à partir des images au même instant des vues déjà encodées et des images de la même vue mais à des instants différents. La figure 8(a) présente les prédictions entre les images faites par le codeur MVC lorsqu'il est utilisé sur une sequence multi-vues.

Comparée à une compression Simulcast de huit vues, où chaque vue est encodée indépendamment par H.264/AVC, la compression MVC présente les résultats suivants [Merkle et al., 2007]:

- À débit équivalent, MVC apporte un gain en qualité entre 1 et 1.5 dB.
- À qualité équivalente, MVC apporte un gain de seulement 15% à 30% du débit.
- Le débit du flux MVC est linéairement lié au nombre de vues supplémentaires, rendant son utilisation difficile lorsque ce nombre de vues augmente.

La faible efficacité du codage MVC est expliquée par le fait que plus de 90% des corrélations dans une vidéo multi-vues sont des corrélations temporelles. Les déformations d'ordre géométriques, présentes entre les vues, sont prédites par MVC à l'aide d'un champ de mouvement par blocs, ce qui n'est pas adapté. En effet, la distance entre deux points de vue est en général beaucoup plus importante que l'amplitude du déplacement d'une camera entre deux images consécutives d'une vidéo. La section suivante propose d'éliminer ces déformations géométriques par l'utilisation des LDI comme représentation intermédiaire, puis d'utiliser le codeur MVC sur les différentes couches pour éliminer les corrélation spatiales et temporelles.

2.2 Codage des LDI par MVC



(a) Codage MVC sur données MVD

(b) Codage MVC sur données LDI

Figure 8: Schémas des prédictions du format MVC pour la compression de séquences 3D. En plus des corrélations temporelles, le codage exploite les corrélations inter-vues lorsque il est utilisé sur des données MVD 8(a), ou les corrélations inter-couches sur des données LDI 8(b).

Le codeur MVC est utilisé sur la séquence de LDI, en considérant chaque couche comme

une vidéo différente. Pour cela, les couches partiellement vides sont complétées par le contenu de la couche précédente, de sorte que le codeur MVC détecte la redondance et n'ait pas à la recoder. Le codeur MVC ne supportant que des vidéos à trois composantes, il est donc utilisé sur l'ensemble des textures, puis séparément sur l'ensemble des cartes de profondeur, considérées comme des images en niveaux de gris. La figure 8(b) présente les prédictions entre les images faites par le codeur MVC lorsqu'il est utilisé sur une sequence de LDI.

3 Synthèse de vues

Les LDI contiennent suffisamment d'informations pour permettre la synthèse de vues virtuelles.

La projection de chaque pixel d'une image selon un nouveau point de vue génère un certain nombre d'artefacts, comme les contours fantômes, les craquelures, et les découvrements (texture inconnue). Les contours fantômes apparaissent à cause des pixels proches des contours, dont la couleur est un mélange entre l'avant-plan et l'arrière-plan, mais dont la profondeur est celle de l'arrière-plan. Les craquelures sont dues à des problèmes d'échantillonnage, laissant apparaître de petits trous entre deux pixels normalement adjacents. Les découvrements sont des zones visibles selon le nouveau point de vue, mais dont la texture n'est pas inclue dans le modèle 3D. Alors que les craquelures peuvent facilement être comblées par application d'un filtre médian sur la carte de profondeur, puis re-synthèse de la texture par projection inverse [Oh et al., 2009], les découvrements sont plus difficiles à compléter. La figure 9(a) montre l'effet de ces artefacts sur une image synthétisée.

Les méthodes classiques pour éliminer les découvrements reposent sur des algorithmes de remplissage (*Inpainting*), utilisant les textures avoisinantes (et même parfois les informations de profondeur [Oh et al., 2009]) pour remplir les trous. L'utilisation d'une méthode de remplissage en post-traitement n'est pourtant pas satisfaisante, dans le sens où l'information de connexité est déjà perdue, rendant certains artefacts impossibles à détecter. La figure 9(b) montre le résultat de la méthode de remplissage implémentée dans la librairie OpenCV. En plus de l'effet d'étalement, causé par le remplissage du découvrement par la texture d'avant-plan, on constate que les craquelures sur le visage ne sont pas supprimées.

La section 3.1 propose une projection innovante, appelée JPF pour *Joint Projection Filling*. Celle ci permet d'éliminer les craquelures, les découvrements et les effets fantômes, directement lors de la projection, sans perdre les informations de connexités. Les résultats de cette projection JPF sont visibles sur la Figure 9(c).

3.1 Projection Ordonnée JPF (pour *Joint Projection Filling*)

McMillan [McMillan, 1995] propose de projeter les pixels dans un ordre déterminé, pour appliquer l'algorithme du peintre et se passer de Z-Buffer. L'ordre est défini en fonction de l'épipôle, de sorte que les derniers pixels posés soient des pixels d'avant-plan. Ce même ordre de parcours présente un second avantage, celui de détecter les découvrements pendant la projection. La différence d'abscisse, après projection, entre deux pixels voisins avant projection, permet de détecter les recouvrements et les découvrements.

La figure 10(b) présente le principe lorsque les caméras sont rectifiées, c'est-à-dire que les

(a) Artefacts (b) Inpainting OpenCV (c) Proj. Ordonnée (JPF)

Figure 9: Découvrements 9(a) et comparaison des résultats d'inpainting d'OpenCV 9(b) et de la projection JPF 9(c).

pixels peuvent être traités par ligne. Soit un pixel $P_i = (x_i, y_i)$ de l'image d'origine se projetant en $P'_i = (x'_i, y'_i)$ dans l'image finale. L'égalité $y_i = y'_i$ implique que chaque ligne est indépendante des autres. Dans les explications qui suivent, nous supposerons que l'ordonnancement de McMillan impose de parcourir chaque ligne de gauche à droite. Dans ce cas:

- Si $x'_i < x'_{i-1}$, alors il y a recouvrement, et le pixel courant est un pixel d'avant-plan.
- Si $x'_i > x'_{i-1}$, alors il y a découvrement, et le pixel courant est un pixel d'arrière-plan, pouvant servir à compléter le trou.

Le signe de ces inégalités est inversé si l'ordonnancement se fait de droite à gauche. Ce procédé est partiellement implémenté dans LDVRS, le logiciel de synthèse de vues rectifiées à partir des LDI de MPEG [Tanimoto et al., 2008].

Nous proposons d'étendre cette méthode à la synthèse de vues non rectifiées, comme le présente la figure 10(c). L'égalité de y_i et de y'_i n'est plus assurée, et les lignes doivent être parcourues simultanément (en pratique, l'image est parcourue par colonne). Pour chaque ligne j de l'image finale, on définit la limite X^j_{max} des pixels déjà remplis par l'équation (3).

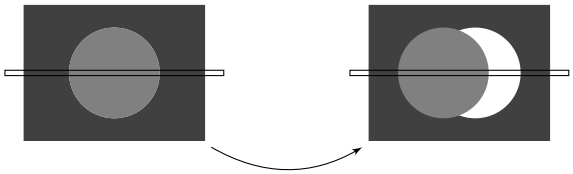
$$X^j_{max} = \max_{\{P'_i \,:\, y'=j\}} (x') \tag{3}$$

Dans ce cas:

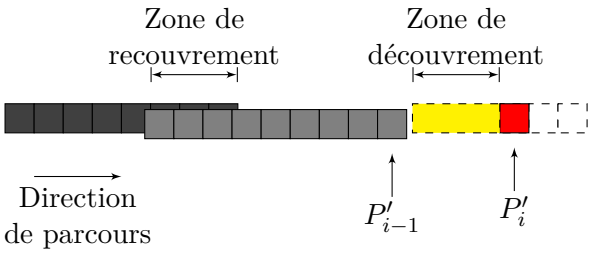
- Si $x'_i < X^j_{max}$, alors il y a recouvrement.
- Si $x'_i > X^j_{max}$, alors il y a découvrement, et le pixel courant est un pixel d'arrière-plan, pouvant servir à compléter la zone découverte.

Cette projection ne demande pas plus de calculs qu'une projection classique. Par contre, tout comme la projection de McMillan, son implémentation sur GPGPU est difficile du fait de son ordonnancement.

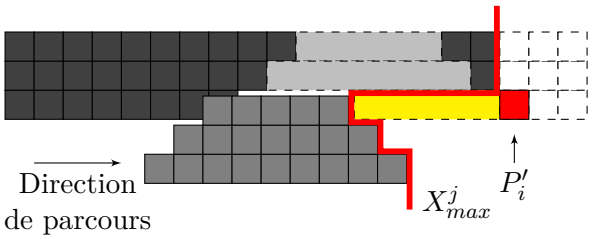
Les résultats obtenus grâce à la JPF sont acceptables pour remplir de petits découvrements, mais quelques artefacts d'étirement peuvent être observés lorsque la zone à remplir est de taille plus importante. Nous proposons d'utiliser la projection JPF pour obtenir la carte de profondeur



(a) Projection



(b) Détails d'une ligne de pixels pour une projection rectifiée.



(c) Détails de quelques lignes de pixels pour une projection non rectifiée.

Figure 10: Principe de la projection ordonnée JPF. Lors de la projection du pixel d'arrière-plan P'_i, celui ci est utilisé pour remplir le découvrement détecté.

de la vue à synthétiser, puis d'employer un algorithme de remplissage qui tient compte de cette profondeur pour combler les grands découvrements. Cet algorithme de remplissage intelligent fait l'objet de la section suivante.

3.2 Prise en compte de la profondeur lors du remplissage

Les algorithmes de remplissage les plus performants se basent sur la duplication de patchs de texture existants [Criminisi et al., 2003]. Une adaptation de cette méthode, utilisant l'information de profondeur pour sélectionner les patchs, a été proposée par Daribo [Daribo & Pesquet, 2010]. Les résultats de cette méthode ont pu être considérablement améliorés grâce à la projection JPF, nous donnant la profondeur de la zone découverte. Nous avons sensiblement modifié les équations proposées par Daribo pour accorder plus de confiance à l'information de profondeur. Les résultats de remplissage obtenus par les deux méthodes sont comparés dans la figure 11.

3.3 Qualité du rendu

Des mesures de la qualité du rendu ont été effectuées, à partir des vidéos MVD "Ballet" et "Breakdancing" (MSR) [Zitnick et al., 2004]. Les I-LDI à deux couches ont été générées selon le point de vue 4, à partir de la première image de chacune des 8 vues, utilisées dans un ordre

(a) Découvrements (b) [Daribo & Pesquet, 2010] (c) Méthode proposée

Figure 11: Méthodes de remplissage aidées par la connaissance de la profondeur.

B-hiérarchique (4; 0; 7; 2; 6; 1; 5; 3).

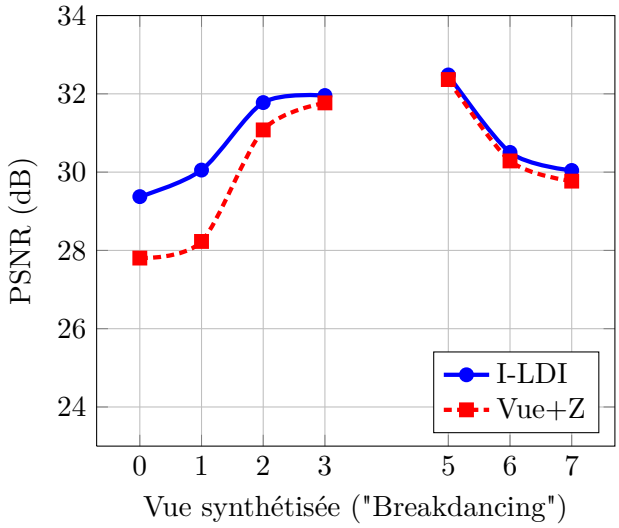


Figure 12: Qualité de rendu des vues virtuelles selon le point de vue.
(Séquence: "Breakdancing"; Référence: Vue 4)

La figure 12 présente les PSNR, mesurés entre les vues générées et les vues acquises, pour la vidéo "Breakdancing". Plus la caméra virtuelle s'éloigne du point de vue de référence, plus les erreurs de projection sont accentuées, réduisant ainsi la qualité de l'image générée.

Pour mesurer l'apport de la seconde couche de la I-LDI, les mêmes mesures ont été faites à partir uniquement de l'image 2D+Z de la vue de référence (c'est-à-dire la première couche de la I-LDI). Les zones découvertes sont alors remplies uniquement par extrapolation, en utilisant la méthode de remplissage proposée. Les PSNR obtenus sont toujours plus faibles qu'en utilisant les deux couches, mais dans des proportions très différentes selon la position de la caméra virtuelle. La géométrie de la scène fait que, pour une caméra virtuelle numérotée entre 0 et 3, les zones découvertes ont des textures uniformes, facilement estimées par la solution d'inpainting. A l'inverse, pour une caméra virtuelle numérotée entre 5 et 7, des zones fortement texturées apparaissent, rendant pertinent le contenu de la seconde couche.

Le logiciel LDVRS de MPEG [Tanimoto et al., 2008] a été utilisé pour comparer les résultats

précédents avec ce qui se fait déjà en synthèse de vues à partir de LDI. Ce logiciel utilise des LDV (semblables aux LDI à deux couches) pour générer des vues virtuelles rectifiées par rapport au point de vue de référence. VSRS [Tanimoto et al., 2008] a été utilisé sur la vidéo MVD non compressée, pour générer un ensemble de vues rectifiées pouvant servir de référence aux calculs de distorsions.

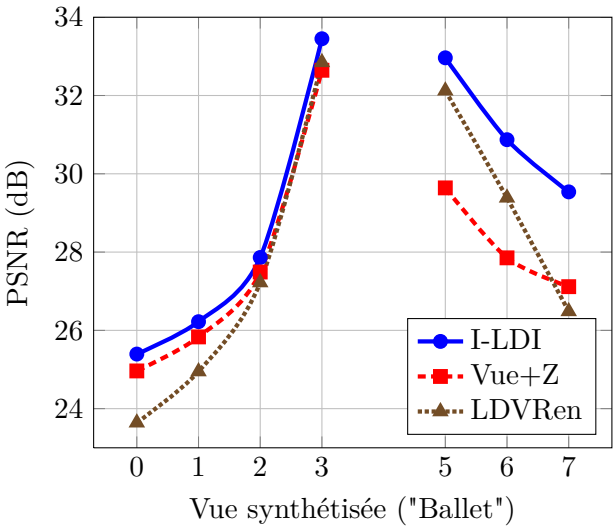


Figure 13: Qualité de rendu pour des vues virtuelles rectifiées.
(Séquence: "Ballet"; Référence: Vue 4)

La figure 13 présente les résultats obtenus sur la vidéo "Ballet". Les mêmes I-LDI sont utilisées pour un rendu par projection JPF, et pour un rendu par LDVRen, un des algorithmes de LDVRS. Quel que soit le point de vue généré, les améliorations proposées dans ce manuscrit apportent un gain en PSNR comparé au rendu de LDVRen. En effet, LDVRen est conçu pour synthétiser des vues très proches de la vue de référence. Lorsque la caméra s'éloigne, il commet des erreurs qui peuvent le rendre moins performant que des algorithmes basés sur une seule vue 2D+Z.

4 Résultats

Les figures 14 et 15 présentent plusieurs graphiques débit/distorsion, obtenus par la compression MVC des différents types de LDI d'une part, et par la compression MVC des données MVD d'autre part. Les graphiques de la figure 14 concernent la séquence "Ballet" (MSR), alors que ceux de la figure 15 concernent la séquence "Breakdancing" (MSR) [Zitnick et al., 2004]. Pour chaque séquence, le graphique de gauche utilise la métrique PSNR, alors que le graphique de droite utilise la métrique SSIM.

Dans un premier temps, les trois types de LDI précédemment cités (LDI, I-LDI et O-LDI) sont synthétisées à partir de la vue centrale 4 comme référence, et de deux vues latérales. Les vues éloignés 2 et 6 sont choisie pour la séquence "Ballet", alors que les vues plus rapprochées 3 et 5 sont préféré pour la séquence "Breakdancing". Les textures et les profondeurs de ces LDI sont encodées indépendamment par MVC, en utilisant les mêmes pas de quantification. Après

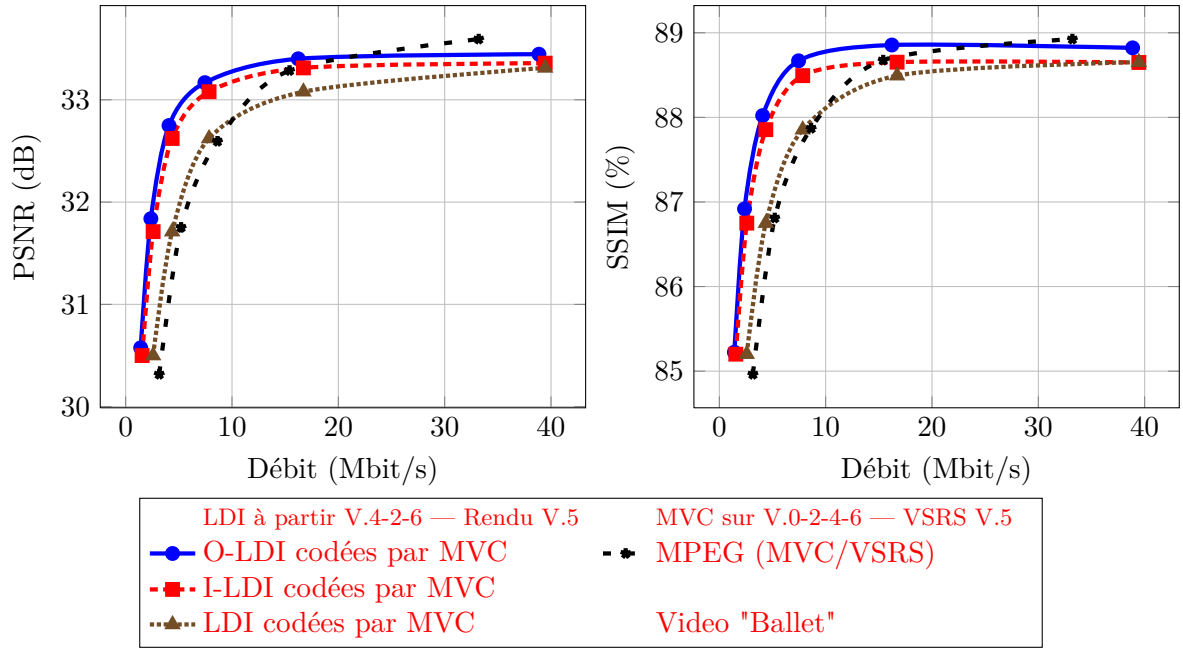


Figure 14: Comparaison débit-distorsion sur la vidéo "Ballet" (MSR), entre les formats LDI, I-LDI, O-LDI, et MVD, tous codés par MVC.

décodage, les LDI sont utilisées pour interpoler la vue intermédiaire 5 pour la séquence "Ballet", et pour extrapoler la vue latéral 6 pour la séquence "Breakdancing".

Dans un second temps, un sous ensemble des vues sont encodées directement par MVC (texture et profondeur au même pas de quantification). Les vues paires (0, 2, 4, 6) sont choisies pour la sequence "Ballet", et les impaires (1, 3, 5, 7) pour la sequence "Breakdancing". Deux des vues sont finalement décodées et utilisées par VSRS, le logiciel de synthèse de vues de MPEG [Tanimoto et al., 2008], pour interpoler la vue intermédiaire. Les vues 4 et 6 de "Ballet" sont utilisées pour synthétiser la vue 5, et les vues 5 et 7 de "Breakdancing" pour synthétiser la vue 6.

Dans tous les cas, la vue synthétisée est comparée à la vue acquise par deux métriques différentes, PSNR et SSIM. La qualité de la vue synthétisée est reportée sur les graphiques de la figure 14 pour la séquence "Ballet", et 15 pour la séquence "Breakdancing".

Comparés au standard MVD classique, les formats LDI proposés présentent un gain en qualité significatif lorsque les débits cibles sont faibles (de l'ordre de 15 Mbits/s). Pour des débits plus élevés, les formats LDI atteignent une saturation, due aux erreurs de projection et d'alignement des contours.

5 Conclusions

Ce mémoire propose une approche basée sur les LDI pour la compression et le rendu de vidéos MVD non rectifiées. La construction incrémentale permet d'éliminer les corrélations inter-vues tout en réduisant le nombre de couches produites. Ainsi, la seconde couche ne contient que 10% de pixels à coder, car différents de ceux présents dans la première couche. La segmentation des couches par objet physique permet d'augmenter la cohérence spatiale, tout en isolant les parties en mouvement. La compression MVC appliquée sur ces couches permet alors d'exploiter

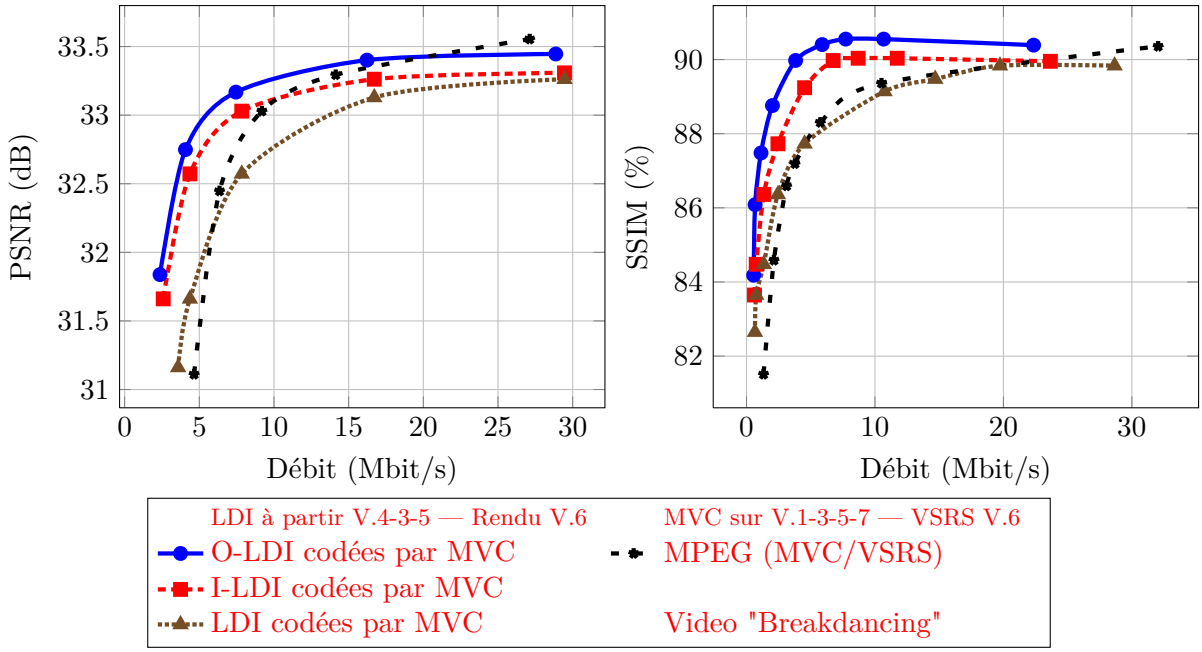


Figure 15: Comparaison débit-distorsion sur la vidéo "Breakdancing" (MSR), entre les formats LDI, I-LDI, O-LDI, et MVD, tous codés par MVC.

les corrélations temporelles et ainsi réduire le coût de codage des LDI. Les LDI produites sont presque indépendantes du nombre de vues utilisées, ce qui rend le format très compétitif comparé à la compression MVC directement appliquée sur les vues, qui lui est linéaire en fonction du nombre de vues utilisées. Enfin, l'utilisation de la projection JPF lors de la synthèse de vue permet d'éliminer les craquelures et petits découvrements, tout en produisant une carte de profondeur complète. Une méthode de remplissage des découvrements est proposée qui tient compte de cette profondeur pour extrapoler les textures d'arrière-plan sans perturber les contours des objets.

Chapter 1

General introduction

The popularity of 3D videos has exploded in recent years. The method is more than two centuries, but its recent introduction to cinema industries revived researches. Until the development of holography dynamic process, 3D effects can be restored by simulating stereoscopic vision. This effect is made possible by the combined use of multiple cameras, each positioned to acquire a certain view.

1.1 Historical context

In 1838, Charles Wheatstone discovered the principle of binocular vision. Because each eye views the world from slightly different positions, each eye's image differs from the other. Objects at different distances project images in the two eyes that differ in their horizontal positions, giving the depth cue of horizontal disparity. Wheatstone showed that this disparity was an effective depth cue by developing the stereoscope. The stereoscope allows to show a different image to each eye (with a set of mirrors), creating the illusion of depth from two flat pictures that differ only in horizontal disparity.

(a) Left image (b) Right image

Figure 1.1: One pair of stereo-images from Japan, taken in the beginning of the 20^{th} century.

In 1844, David Brewster popularized the stereoscope by combining prism and photography. Tens of thousands of stereograms were then produced, like the one shown in Figure 1.1.

In the 1960s, Bela Julesz invented random-dot stereograms. In previous stereograms, each half image showed recognizable objects. On the contrary, each half image of a random-dot stereogram shows a pattern of random dots. No recognizable objects can be seen in either half image. The two half images of a random-dot stereogram are essentially identical, except that one has an area of dots shifted horizontally, producing horizontal disparity. The gap left by the shifting is filled in with new random dots, hiding the shifted area. Nevertheless, when the two half images are viewed one to each eye, the area becomes visible by appearing closer or further than the rest of the image.

In the 1970s, Christopher Tyler invented auto-stereograms, as shown in Figure 1.2. These auto-stereograms are special stereograms that can be viewed without a stereoscope. Created from a pattern that repeats horizontally with slight distortions, an auto-stereogram reveals its secret if it is viewed correctly. The viewing technique requires that the eyes take a relative parallel angle and focus somewhere behind the image.

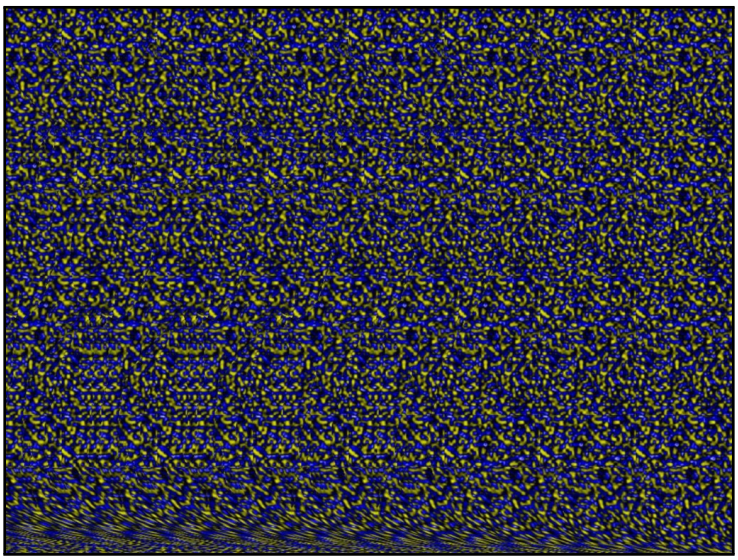


Figure 1.2: An auto-stereogram image whose secret is revealed by converging eyes at an infinite point.

The first stereoscopic film was created in 1903, but could then be seen by only one viewer at a time, appropriately positioned and equipped with a stereoscope. It was not until 1922 and the development of stereoscopic anaglyphs that a film could be seen in 3D by several viewers simultaneously. The principle then being developed for the movie industry, research effort was mainly directed towards improving visual quality, largely ignoring the issues of transport and compression. Thanks to the development of polarized glasses in 1929, and the addition of color in 1952, watching stereoscopic films became more comfortable. Given the success, the process became more widespread and as early as 50s, the production of stereoscopic films intensified. Nowadays, most films are captured in stereoscopy, and almost all cinemas provide adapted glasses to the audience.

Along with the development of 3D in the movie industry, various techniques were developed to avoid the need for the viewer to wear 3D glasses.

In 1896, Auguste Berthier introduced the parallax barrier, which is a device placed in front of an image source, to show a stereoscopic image or a multiscopic image. Nowadays, a parallax barrier consists of a layer of material with a series of precision slits. When placed in front of classical screen, the barrier allows each eye to see a different set of pixels, thus creating a sense of depth through parallax effect. The technology presents some disadvantages: the viewer must be positioned in a well-defined spot to experience the 3D effect; the effective horizontal pixel resolution viewable for each eye is reduced; and the perceived brightness of the screen is reduced by the barrier.

In 1908, Gabriel Lippmann introduced integral photography, in which a plane array of closely spaced small lenses is used to acquire a scene, recording images of the scene as it appears from many slightly different horizontal and vertical locations. When the resulting images are rectified and viewed through a similar array of lenses, a single integrated image, composed of small portions of all the images, is seen by each eye. The position of the eye determines which parts of the small images it sees. The effect is that the visual geometry of the original scene is reconstructed, so that the limits of the array seem to be the edges of a window through which the scene appears life-size and in three dimensions, realistically exhibiting parallax and perspective shift with any change in the position of the observer.

In 1912, Stephen Benton showed that a lenticular lens could be used for the same purpose. The lenticular lens is a one-dimensional array of cylindrical lenses, designed so that when viewed from slightly different angles, different images are magnified. The most common example is the lenticular printing, where the technology is used to give an illusion of depth, or to make images that seem moving as the viewing angle varies.

With the democratization of 3D display devices, some TV channels start to broadcast contents in 3D, revealing transmission and compression issues. The project ATTEST (Advanced Three-Dimensional Television System Technologies) was created to centralize all the innovations in this domain [Fehn et al., 2002]. Standards groups, such as MPEG and ISO, are defining standards, essential to the agreement of manufacturers and system interoperability.

1.2 Present challenges

The denomination "3D" refers to any visual system that attempts to recreate the illusion of 3D sphere of human viewing. In the case of video application scenarios, we will refer in this work to the pseudo-3D denomination, which includes the ability of the viewer to perceive an illusion of depth, and the illusion to navigate in a 3D space. These two functionalities are commonly called 3DTV and FTV, and can be combined [Onural et al., 2006, Kubota et al., 2007].

3D displays (3DTV) describes the functionality of the television to be viewed with a depth feeling. The effect of depth is restored by providing each viewer's eye images of the scene seen from slightly different angles. Although both views are sufficient to simulate stereoscopic vision, the use of more views allows multiple viewers simultaneously, and allows them to move slightly, thus improving comfort and immersion.

Free viewpoint TV (FTV) also known as Free Viewpoint Video (FVV), is an innovative technology that enables the viewer to select his preferred viewpoint, and to move dynamically in the video. The view observed from the desired viewpoint is computed from available data, in real time. The fluidity of the displacements and the quality of the rendered views depend on the virtual view synthesis algorithm and the data it manipulates.

The technologies that convey the 3D experience to the viewer are distributed over the entire video processing chain, from the capturing step to the rendering step. Figure 1.3 shows the complete video processing chain which is used in almost all 3D applications, including cinema industry, television, video conferencing, and virtual inspection.

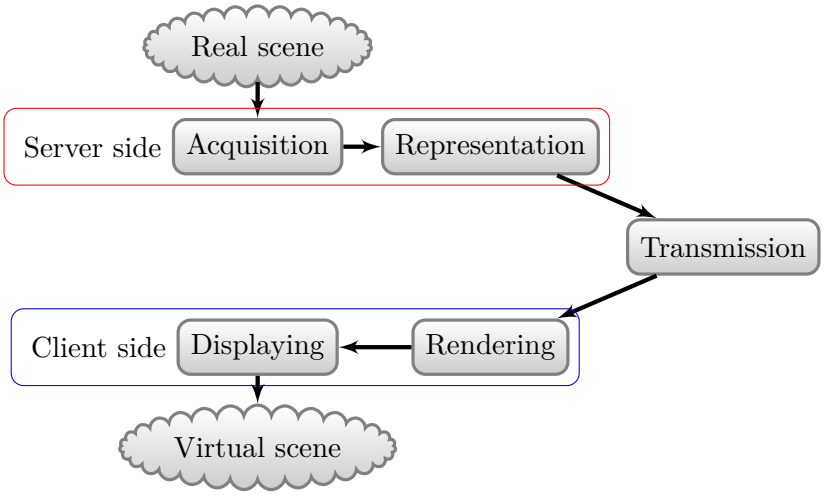


Figure 1.3: 3D video processing scheme, from real scene acquisition to virtual scene displaying.

The processing chain involves two entities, commonly named the server and the client. The server captures the scene with a dedicated device, formats the data in a specific representation, compresses the representation and sends the data to the client. The client receives and uncompresses the data, renders the required views and displays the virtual scene to an observer. In the context of 3D videos, each one of these stages raises many challenges.

Acquisition is the process of capturing information from the real world, using a variety of devices, including color cameras and depth cameras. Many technical choices have to be made, such as the characteristics of the cameras (resolution, framerate) and how they are arranged together (small baseline, sparse setup). Various methods have been explored, and specific capture solutions have been developed for each use-case.

Movie industry uses rigs of two side by side HD cameras, spaced a few centimeters, perfectly aligned and synchronized. This setup is particularly adapted for high quality stereo movies, where each camera captures exactly the content for each eye.

Recent auto-stereoscopic displays unfortunately require more than two views, arranged in a very specific disposition. Dedicated devices were thus built for each display, which can contain more than eight cameras. Increasing the number of cameras also increases the complexity of the capture system in terms of computation complexity as well as data load for transmission. Since physical rectification is not perfect, an additional image rectification algorithm is often applied

to the images during or after the acquisition process [Pollefeys et al., 1999, Fusiello et al., 2000, Fecker et al., 2008]. An auto-stereoscopic display is currently in construction, which works with no less than two hundreds views. With such a configuration, a generic solution must be designed.

An alternative to dedicated capture devices is to design a generic multi-views capture system, with a longer distance between cameras so that to find a compromise between the system complexity, the desired navigation range, and the final video quality. In this case, intermediate views between cameras have to be computed using so-called virtual view synthesis algorithms, as explained in Part I. To do so, cameras must be related to each other, i.e. exact position and orientation of the cameras must be estimated. This process is called geometric stereo camera calibration [Tsai, 1986, Zhang, 2000].

Representation is the process of converting the captured data into a convenient format, which allows easier compression and high-quality rendering. This representation may contain some geometric information about the scene in addition to color information [Benzie et al., 2007]. The choice of the representation plays a central role in a multi-view video system, as it influences the compression method and the virtual view synthesis method to be used. Section 2.1 of this manuscript describes existing representations and Part II is composed by two chapters dedicated to one of them: the Layered Depth Image representation (LDI). Chapter 5 presents the LDI representation while Chapter 6 introduces our contributions to their construction, compression, and rendering.

Transmission is the process of sending, propagating and receiving data over a point-to-point or point-to-multipoint network, either wired or wireless. The transmission should be fast enough to provide comfortable visualization at the client side, without large delays [Akar et al., 2007]. The amount of data captured by the multi-camera system is however larger than traditional single-view videos, and the network may thus be overloaded. In order to reduce the bit rate, compression methods adapted to the chosen representation are particularly important and challenging [Sayood, 2005, Schreer et al., 2005]. Section 2.2 introduces major compression methods, adapted to a wide variety of representations.

One of the main principle of compression methods is to remove correlation in the data. For video compression, both temporal and spatial correlations between pixels are exploited [Le Gall, 1992]. Compression methods could use either lossless algorithms, or lossy algorithms. Lossless compression algorithms exploit statistical redundancy in such a way as to represent the data more concisely without error. Lossy compression algorithms permit that some information is lost, involving distortions in the decoded data. The performance of lossy compression methods are evaluated by the compromise between the reached bit-rate and the observed distortions.

Special attention is required to adapt the compression method to the data, since artifacts vary depending on the properties of the data. This is particularly true for color images and depth maps: both are pixel-based images but with different characteristics that require adapted compression methods. A lot of research activities have recently been dedicated to compression methods adapted to depth maps compression [Maitre & Do, 2009, Merkle et al., 2009, Morvan, 2009, Yea & Vetro, 2009].

It is important to define compression standards ensuring that videos can be watched by

as many users as possible. Concerning video compression, two working groups of experts are developing standards: MPEG (Moving Picture Expert Group) and VCEG (Video Coding Expert Group). These two groups collaborated in the Joint Video Team (JVT) for defining video compression standards (e.g. AVC, SVC, MVC). Within MPEG organization, a sub-group called 3DV is currently studying the representation and compression of multi-view data for 3D video applications. In addition, the MPEG's group 3DGC (3D Graphics Compression) is working on 3D graphics and mesh compression standards. Existing standards are: MPEG-4 AVC for single-view video compression; MPEG-4 MVC for multi-view video compression; MPEG-4 AFX for 3D graphics tools; and MPEG-4 3DMC for 3D mesh compression.

Rendering is the process of generating an image from a 3D representation, by means of computer programs. As mentioned above in the acquisition stage, all required views could not always be captured, and some additional virtual views should thus be computed. These virtual views are generated with Image-Based rendering (IBR) algorithms, which use the transmitted images and possibly additional geometry information [Shum et al., 2007].

IBR methods are mainly based on warping techniques, which project a reference view onto a virtual viewpoint. Pixels of the original images are first modeled into the 3D space using either stereo analysis algorithms [Sourimant, 2010b], or depth camera with back-projection algorithm. The 3D model is then projected onto the image plane of the virtual view, to synthesize the virtual image.

Directly applying warping equations to each image pixel may cause some artifacts in the synthesized view, like disocclusions, cracks and ghosting artifacts. Disocclusions are areas occluded in the reference viewpoint and which become visible in the virtual viewpoint, due to parallax effect. Cracks are small disocclusions, mostly due to texture re-sampling. Ghosts are artifacts due to projection of pixels that have background depth and mixed foreground/background color.

During this rendering process, special care has to be taken about depth discontinuities and occlusions. Some additional processing of the virtual view is needed for hiding remaining artifacts, including texture blending or disocclusions filling [Bertalmío et al., 2000]. All the above computations needed for intermediate view synthesis represent a certain complexity. Since real-time visualization is usually desired for interactive 3D video experience, the view-synthesis complexity should always be compatible with the computation power of the device. Thanks to recent graphics hardware and programming improvements, some view-synthesis methods can run in real-time. This is because of the high parallel processing capabilities of such hardware. However, if no powerful graphics hardware or any other specialized parallel devices are available, e.g. on a mobile device, then only low complexity view-synthesis methods can be performed.

Part I of this manuscript is fully dedicated to virtual view synthesis algorithms and is composed by two chapters. Chapter 3 describes existing view synthesis methods and Chapter 4 introduces our contributions in this field of interest.

Displaying is the process of making visible the synthesized views for a viewer. This step can be performed by screens, projectors, holograms, auto-multi-scopic display or volumetric displays. Depending on the technology, several virtual views can be displayed simultaneously [Benzie et al., 2007].

If only free viewpoint navigation is wanted, the display may be any classical 2D screen. The navigation would then be provided by some user interface, like a remote, a mouse or even a tracking device, along with adapted processing hardware. However, if stereoscopic visualization is desired, special displays are needed here. The challenges are to provide the user with an accurate description of the scene with a natural sensation of depth. Moreover, comfort of visualization and freedom of movement are important to prevent fatigue. Finally, the cost of the device is an important criterion for acceptability into the mass market.

Today's most popular displays are based on the stereoscopy principle: two images of the scene are displayed, they correspond to the two views of the eyes. However, to ensure that each eye sees only one image, special glasses must be worn. Two different techniques are mainly used: alternate frame sequencing of the left and right images with synchronized shutter glasses and a high frame rate display, or projection with polarizing projectors and polarizing glasses. The main drawback with these solutions is that glasses are a source of discomfort and reduce the luminosity of the images.

In order to eliminate the use of special glasses, multi-view auto-stereoscopic screens have been developed [Dodgson, 2005]. Instead of glasses, a system that separates many images is placed directly in front of the screen. Two concurrent systems are currently used: parallax barrier or lenticular lenses. A drawback is that the resolution of each image is reduced proportionally to the number of views displayed simultaneously, and that stereoscopic visualization provides only horizontal parallax. These multi-view auto-stereoscopic displays are just introduced to the mass market.

Auto-stereoscopic screens do not produce real 3D images but provide a sensation of depth by simulating stereo vision. On the other hand, volumetric displays [Favalora, 2005] and holographic displays [Slinger et al., 2005] provide a more detailed description of the scene by either filling the 3D space with imagery or reproducing the light wavefront reflected by the scene. So far, such systems are under development and not yet mature enough for be introduced into the mass market.

1.3 Objectives

The objective of the thesis is to develop tools in order to use Layered Depth Images (LDI) as an intermediate representation for multi-view video compression.

The multi-view system is the most widely used acquisition method. In this system, multiple cameras capture the same scene from different viewpoints. Cameras must be placed accurately, synchronized, and their colorimetry should be equalized. In the remainder of this work, we consider that the capturing process is already performed, and the cameras parameters are provided within the videos, and do not need to be estimated.

Transmitting each captured video as is, requires an important bandwidth, as explained in Chapter 2. An intermediate representation of the scene should thus be used, allowing efficient compression and high-quality rendering for both 3DTV and FTV uses.

Depth maps are classical representation of the geometry of a scene. A depth map is created by assigning a depth value to each pixel of the captured texture image and combining those depth values into a single image. The methods used for synthesizing these depth maps are not detailed

in this manuscript, but can be found in [Hartley & Zisserman, 2004, Tanimoto et al., 2009, Morvan, 2009, Sourimant, 2010a]. In the remainder of this work, we consider that the depth value of each captured pixel is provided together with the color of this pixel.

1.4 Contributions

In this manuscript, various steps of the transmission chain are addressed. The intermediate representation is particularly studied, in order to remove redundancies in the data, and to have an efficient compression method. The rendering methods are also discussed in order to improve the final synthesized videos.

Some methods, originally designed to improve the rendering quality, are also used to produce the intermediate representation. That is why the rendering methods are described before the intermediate representation is discussed. The two parts of this manuscript are thus reversed and do not follow the ordering of the 3D processing scheme.

The first part describes methods for synthesizing virtual views from videos and depth maps (DIBR). It introduces several contributions in order to enhance the visual quality of the rendered views.

The major contribution is a novel forward projection method for DIBR, using occlusion compatible ordering [McMillan, 1995] for detecting cracks and disocclusions, for which the unknown depth values are estimated while performing the warping.

An extension of the JPF method is then proposed, relying on a depth-based pixel confidence measure for removing ghosting artifacts. Pixels along depth discontinuities contain mixed foreground and background color value, and their projection produces annoying ghosting artifacts. A refinement of the projection equations is thus proposed, which avoids ghosting artifacts by shifting these mixed pixels along new depth discontinuities.

The resulting projection method thus allows us to handle depth maps warping and disocclusion filling simultaneously. Small cracks and large disocclusions are handled gracefully, with similar computational cost as simple forward projection, avoiding the use of a filtering step as done in the classical approach.

Some stretching artifacts may appear when the filled disocclusion is too wide. A Full-Z depth-aided inpainting is thus proposed, which takes into account all information given by the depth map to fill in disocclusions with textures at the correct depth.

Finally, a method is proposed to handle inaccuracies of cameras calibration and depth map estimation when synthesizing intermediate view from multiple view plus depth. The method is based on the Floating Texture approach.

The second part provides details on the LDI representation, and describes our contributions for LDI construction, compression, and exploitation.

An incremental LDI construction method is proposed, which avoid redundancies between layers. The LDI is constructed incrementally, and at each step, only the missing information is inserted into the layers. The resulting layers thus contain fewer pixels, making the later compression easier.

An Object-based LDI representation is also proposed, which improves the quality of synthesized virtual views. The O-LDI contains only two layers, namely the background layer and the object layer, and it is particularly adapted in a rate-constrained context. Depending on the video sequence, the background layer is often static, and does not vary along time. Only the object layer contains pixels from the moving objects.

We also describe an LDI compression scheme, based on the MVC coder from MPEG. MVC is designed to exploits both temporal redundancies and inter-view redundancies. It uses a motion-based prediction to remove redundancies, which is not perfectly adapted to inter-view correlations. Instead, we propose to use it capabilities to compress LDI layers by removing inter-layer correlations.

Finally, we propose two methods for synthesizing virtual views from LDI. The first one is a mesh-based LDI rendering method, which is GPU optimized, and allows real-time rendering for an eight-views auto-stereoscopic display. The second one is based on the JPF method, which handles artifacts and improves visual quality.

1.5 Scope of the thesis

The layout of this thesis is divided in two parts. The first part addresses the view-synthesis problem, answering the question of how 3D video may be rendered on a classical display or a 3D display. The second part introduces the Layered Depth Image (LDI) representation, and explains its advantages for 3D video compression and rendering.

Chapter 2 is a global introduction to existing 3D representations and introduces the theoretical foundations of video compression methods. This chapter first presents various 3D video representations, then explains the relevant theory for understanding the block-prediction principle for video coding.

Part I: Virtual view synthesis

Chapter 3 introduces some existing DIBR techniques, which have been recognized as a promising tool for supporting advanced 3D video services. First, this chapter introduces the projective geometry which is the basis for virtual view synthesis with Depth-Image-Based Rendering (DIBR) methods. Then, it addresses the inherent problems with DIBR algorithms, which are disocclusions, cracks and ghosting artifacts. Finally, it presents existing solutions from the literature to avoid each of these artifacts.

Chapter 4 presents the contributions of this thesis for virtual view synthesis with DIBR techniques. The Joint Projection Filling method (JPF) is first introduced as a projection method which handles disocclusions problems during the projection, to make use of the scene geometry. This JPF algorithm is then used in two DIBR methods, designed for either intermediate views interpolation or virtual views extrapolation. The intermediate views interpolation method uses an optical flow estimator to reduce the blur and increases the level of details in the rendered views. The virtual views extrapolation method uses a full-Z depth-aided inpainting method to fill in large disocclusions with a coherent background texture.

Part II: Layered Depth Image (LDI) for compact 3D representation

Chapter 5 defines the LDI structure and introduces some state-of-the-art algorithms to build and manage LDI. First, methods for LDI construction are presented, either from 3D models or from captured images. Then, techniques for LDI compression are detailed, based on pixels aggregation of layers filling.

Chapter 6 presents the contributions of this thesis on LDI construction, compression, and exploitation. The proposed incremental construction scheme reduces the number of pixels in additional layers while increasing their compactness. The object-based restructuring improves the LDI compression efficiency while minimizing artifacts appearance. The proposed LDI compression scheme is based on the MVC coder from MPEG to exploit temporal correlations, spatial correlations, and inter-layer correlations. The constructed layers are finally compatible with a fast mesh-based rendering, and can also be rendered by the JPF methods presented in Chapter 4.

Chapter 7 concludes on the contributions of the dissertation and discusses some perspectives.

Chapter 2

3D content representations and compression format: a state-of-the-art

This chapter is a wide introduction to state-of-the-art representations for 3D videos. These intermediate representations aim to organize the information to improve compression efficiency, without reducing rendering quality.

Section 2.1 provides an overview of some existing 3D video data representations which could support interactive 3D video services.

Section 2.2 introduces and describes coding standards for encoding those 3D video data representations. Classical features of 2D video compression methods are first presented. The principle of block-based predictive video coding is then adapted to efficiently encode the huge amount of data required by 3D video communication services.

2.1 3D video sequences representation

After the acquisition of a multi-view video, it is important to organize the large volume of captured information in order to transmit the sequence or to synthesize virtual views. This section is devoted to the study of a number of intermediate representations described in the literature [Alatan et al., 2007, Bruls et al., 2007, Daribo, 2009, Colleu, 2010]. These representations have various structures which optimize either complexity, flexibility, or rendering quality. These representations are classified in two categories, according to their main use.

Section 2.1.1 details some rendering-optimized representations, which allow a high quality rendering, but significantly increase data size. These representations are mainly used locally, where storage space is not limited.

Section 2.1.2 presents some transmission-optimized representations, which are more compact and thus rather intended for transmission or limited storage capacity devices. These representations aim at discarding a significant portion of redundancy present in the captured data.

2.1.1 Rendering-optimized representations

The rendering quality of a multi-view video is strongly dependent on the representation of the video in memory. For a photo-realistic rendering, it is necessary to keep a large amount of acquired information. This section describes three families of representations adapted to high quality rendering, which are the multi-video representation, the plenoptic function and the microfacet billboarding.

Multi-video with additional geometry

The multi-video is the most natural representation of a multi-view video. It consists of all single-view videos acquired by different cameras. The representation allows direct access to all the pixels in each view, and every moment of the video. This representation also minimizes the conversions made to the acquired videos, and thus minimizes image degradation.

This representation usually includes some additional information about the geometry of the scene, in order to facilitate and improve the quality of the virtual view synthesis process. This geometric information may be computed by algorithms for estimating disparity between views. The access to each image of each view allows the use of advanced rendering algorithms for multi-textured synthesis of photo-realistic virtual views [Debevec et al., 1996, Debevec et al., 1998, Zitnick et al., 2004].

The plenoptic function

The plenoptic function is the 5-dimensional function representing the intensity or chromaticity of the light L observed from every position (x, y, z) and direction (θ, ϕ) in 3-dimensional space [Adelson & Bergen, 1991]. In image-based modeling, the aim is to reconstruct the plenoptic function from a set of example images. Each image defines some discrete values of the plenoptic function, which are then interpolated [Fan et al., 2004]. By interpolation between these discrete values, it is feasible to generate images by indexing the appropriate light rays.

Sampling and storing a 5-dimensional function for any useful region of space is impractical, so researchers have used both constraints on the viewpoint and coherence inherent in the function to reduce the problem's complexity.

If the plenoptic function is only constructed for a single viewpoint in space then its dimensionality is reduced from 5 to 2. This is the principle used in reflection mapping (also known as environment mapping) [Blinn & Newell, 1976], where the view of the environment from a fixed position is represented by a 2-dimensional image. This reduction in the dimensionality of the plenoptic function by constraining the viewpoint has some analogy with the mapping of image sequences onto an image plane, as in the construction of panoramas.

The Light Ray representation is derived from the plenoptic representation, when the environment is viewed from inside its convex hull. The 5-dimensional plenoptic function is thus reduced to a 4-dimensional function, because any particular ray through the convex space always intersects the same surfaces. Figure 2.1 presents this simplification, where the light rays are indexed according to the coordinates (s, t) and (u, v) of their intersection with two parallel

planes [Levoy & Hanrahan, 1996, Gortler et al., 1996]. This particular representation has been proposed for standardization [Tanimoto et al., 2008].

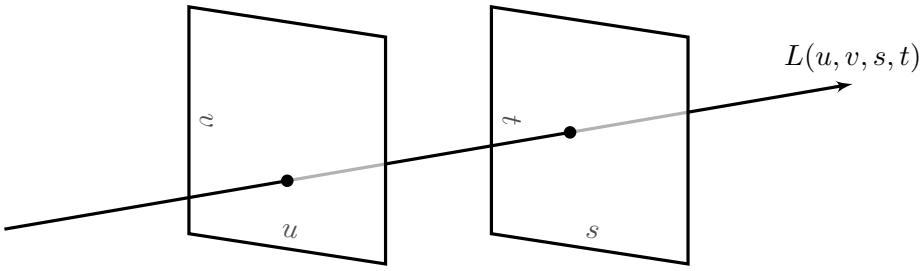


Figure 2.1: Setting of a light ray. $L(u, v, s, t) =$ color.

The synthesis of any virtual view is performed by assigning a color to each pixel of the new image. For each pixel, one characterizes the light beam passing through that pixel and arriving at the optical center of the camera. The plenoptic function is then computed for this light beam, by interpolating from light beams arriving in optical center of nearby cameras.

The plenoptic function allows scenes and objects to be rendered very efficiently for novel viewpoints. The virtual camera can rotate, translate laterally and in depth, as well as perform zooming (changing intrinsic parameters). However, it is important to have a high density of input cameras during the acquisition, in order to properly interpolate the plenoptic function. Storage of the function (even the 4-dimensional function) requires very large amounts of memory. A typical scene or simple object will require several hundreds of megabytes.

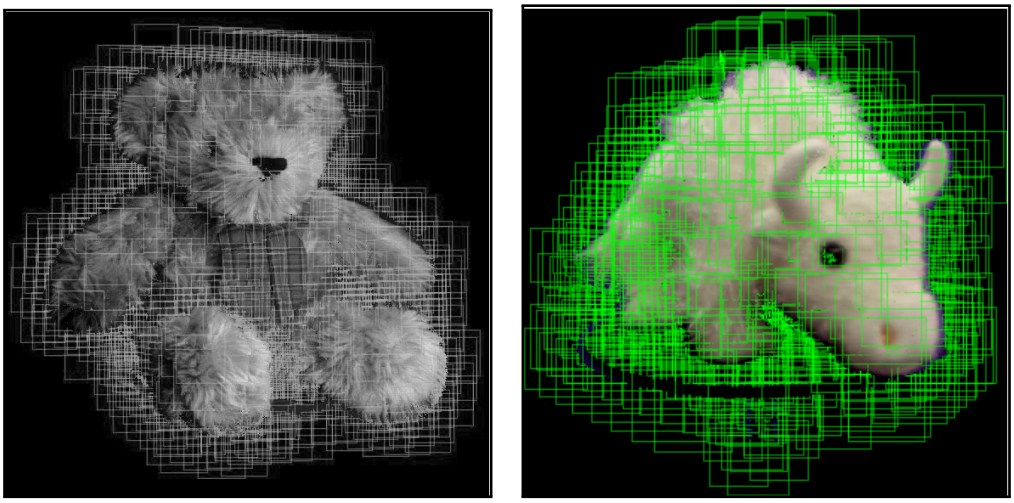
Light Field Surface is introduced in [Kitahara et al., 2007] as an alternative to index the plenoptic function. Instead of indexing the rays by their positions and orientations of observation, the Light Field Surface indexes the same rays by their positions and orientations of emission. This approach requires good knowledge of the geometry of the scene in order to correctly estimate the origin of a ray when only its direction of incidence is observable by a camera.

Microfacet billboarding

The microfacet billboarding [Yamazaki et al., 2002] is a representation of the scene by a set of many small patches of texture, named billboards. The billboards are always oriented perpendicularly to the viewing direction and discretely approximate the object geometry. The texture of each sprite is selected from the most suitable input images, according to the viewpoint. These view-dependent microfacets with view-dependent textures can render intricate geometry from various viewpoints. The partial overlap and semi-transparency of these billboards provides a photo-realistic rendering despite low density acquisition cameras.

Billboarding, combined with alpha texturing and animation, can be used to represent many phenomena that do not have solid surfaces. Smoke, fire, fog, explosions, energy shields, vapor trails, and clouds are just a few of the objects that can be represented by these techniques.

A real-time rendering method uses a cloud of microfacets for representing complex 3D objects [Yamazaki et al., 2002, Goldlucke & Magnor, 2003, Furuya et al., 2007]. These microfacets allow great freedom of movement of the virtual camera. Figure 2.2 shows two results made according to arbitrary poses, which are not among the poses acquisition.



(a) (b)

Figure 2.2: Rendering results using Microfacet billboarding. (from [Yamazaki et al., 2002])

Conclusion

These rendering-based representations provide high realistic rendering, but require a huge amount of data for storage and transfer. Next section introduces some transmission-optimized representations, which eliminate some redundant information from the input cameras. These representations are more adapted to rate-constrained context.

2.1.2 Transmission-optimized representations

This section presents some representations of 3D videos which aim at eliminating correlations between acquired views in order to reduce the bandwidth required for transmission. These representations are classified either as image-based representations or model-based representations, depending on their use of 3D models.

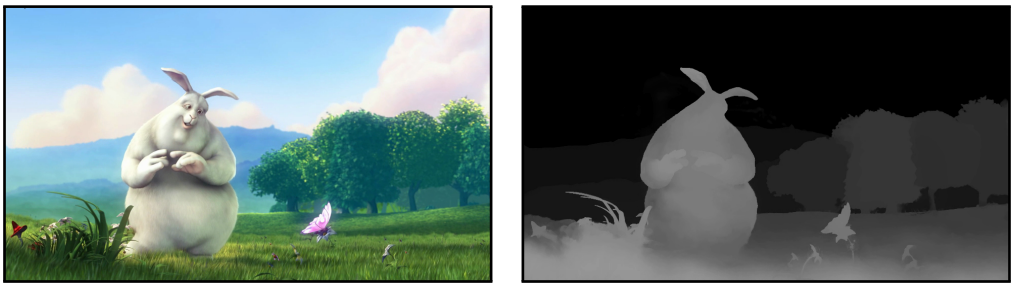
Image-based representations work directly on the captured videos. The transformation preserves the representation as matrices of pixels. The geometric model of the scene that is sometime associated with the textures is simplistic and has a reduced impact on the final rendering quality.

Model-based representations use the acquired views to build a detailed 3D model of the geometry of the scene. Their rendering quality is strongly related to the accuracy of the geometric model. This model is compressed and transmitted in addition to texture data. The compression method used on the 3D model is different from video coding techniques.

Image-based representations

Image-based representations preserve texture information as color videos, i.e. an array of pixels. In addition, they sometimes use a low-detailed model of the scene to provide geometric information. This section introduces three common image-based representations, organized in complexity order of their 3D model.

2D plus depth video representation (or 2D+Z) is a classic representation, standardized by the MPEG group [ISO/IEC, 2007]. It provides a regular color video, enriched with the associated sequence of depth maps. A depth map is an image that contains the distance of scene objects to the viewpoint. Figure 2.3 shows one frame of such a 2D plus depth video sequence [1].



(a) Texture frame. (b) Depth map frame.

Figure 2.3: One frame of a 2D plus depth video sequence. Darker is farther.

The 2D+Z representation allows the synthesis of virtual view by using Depth-Image-Based Rendering techniques (DIBR). The main limitation of the 2D+Z representation is the disocclusion problem which occurs when the virtual camera moves away from the reference viewpoint, as some hidden texture becomes visible. If disoccluded areas are small enough, they can be filled in by inpainting methods [Tauber et al., 2007, Wang et al., 2007], but annoying artifacts appear when disocclusion areas are too wide. Camera freedom is thus limited to small displacements around the reference camera position.

Layered Depth Image structure extends the 2D+Z format by transmitting hidden texture and depth values as additional layers. A picture no longer consists in a single layer of pixels, but may contain several layers, with multiple associated depth maps.

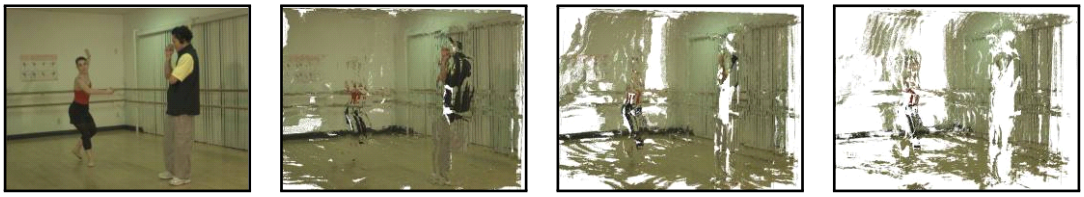
The LDI structure is introduced in [Gortler et al., 1997] and [Shade et al., 1998] as a set of billboards (or sprites). Detected with a segmentation of the depth map, each structural object is isolated on a different layer, modeled by a billboard. Each billboard may be flat, or accompanied with the corresponding depth map and rendered in 3D as displacement-mapped billboard [Waschbüsch et al., 2007]. Figure 2.4 shows the depth maps of the different layers, and an obtained rendering result.

This LDI representation had evolved, and the layers of a modern LDI no longer represent physical objects, as explained in [Yoon et al., 2004, Yoon et al., 2006b, Cheng et al., 2007a, Barenbrug, 2009]. The first layer contains the information of texture and depth of all that is visible in the scene, from a reference viewpoint. The other layers contain the texture and depth values of pixels hidden from the reference viewpoint. These additional layers are used to fill disocclusion areas in synthesized views. Figure 2.5 shows the content of the first layers of a LDI, where each pixel is associated with depth information. This representation is more extensively studied in part II of this manuscript.

[1] (c) copyright 2008, Blender Foundation / www.bigbuckbunny.org

(a) Segmented depth map. (b) Rendered view.

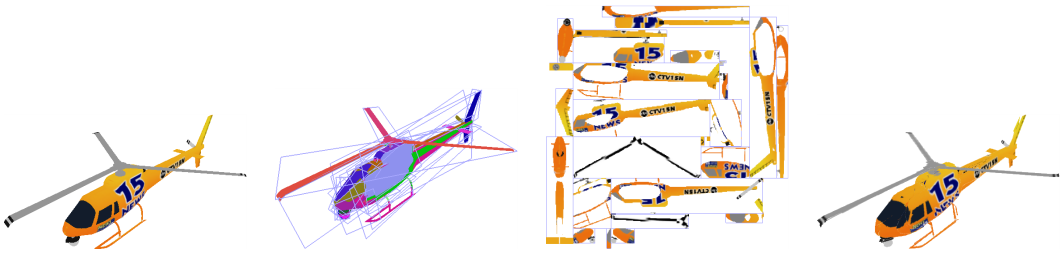
Figure 2.4: A set of displacement-mapped billboards. Each billboard is associated with a depth map. (from [Shade et al., 1998])



(a) 1st layer (b) 2nd layer (c) 3rd layer (d) 4th layer

Figure 2.5: The first layers of a LDI representation.

Billboards clouds are frequently used in virtual imaging for their suitability for fast rendering [Décoret et al., 2003]. This representation has some similitude with the microfacet billboarding representation, except that the billboards are now fixed in the 3D space, and do not rotate with the viewing direction. The geometry is represented by a complex tangle of some partially transparent 3D planes.



(a) Original model (b) Billboards (c) Textures (d) Rendering

Figure 2.6: Model simplification with billboard cloud. The original model with 5138 polygons 2.6(a). Simplified model with 32 billboards 2.6(b). Textures of billboards 2.6(c). Rendering result of the simplified model 2.6(d). (from [Décoret et al., 2003])

In [Décoret et al., 2003], the authors propose to use billboards in order to simplify 3D models. The geometry of the model is lost by the transformation, but the rendering result is correct. Figure 2.6 presents a model of more than 5000 polygons and its simplification as a cloud of only 32 billboards.

Model-based representations

Model-based representations use information from the acquisition cameras to generate a detailed 3D model of the scene [Lavoué et al., 2008, Cohen-Steiner et al., 2004, Balter, 2005, Debevec et al., 1996, Lee et al., 2002]. These representations are distinguished primarily by the type of 3D primitives generated. Three common 3D primitives are presented in this section, which are Point clouds, Voxels, and Meshes.

Point clouds model the scene as a set of colored 3D points. The points are independent of each other, and are not connected. Voxels form a uniform and regular segmentation of the scene into elementary volumes. Meshes model the scene surface as a set of connected polygons.

Point clouds is the simplest representation of a 3D object, but also the most data consuming.

The points are the most simple 3D primitives to define a surface. They are not connected together, and are usually represented by three coordinates and a color [Levoy & Whitted, 1985]. Rendering algorithms either project each point on a single pixel of the virtual image, or draw small discs, covering several pixels, which is called splatting [Zwicker et al., 2002].

A typical approach is to attach the normal vector of the surface as additional information to each point. It allows in particular to model reflections. The points are drawn as arbitrary diameter discs, tangent to the surface, the color fades away from the center to become completely transparent.

The main interest of this representation is its low complexity. It prevents the graphics engine to project complex polygons, which sometimes appear smaller than a pixel. Its implementation needs a large number of points to give realistic results. So it is a representation which is rather suitable for very detailed scenes with high bandwidth available for transfer.

Voxel representation is used to describe a complete 3D scene with a uniform level of detail. Just as images are composed by pixels, the 3D space is uniformly divided in small cubes, which represent the smallest representable element. The voxels' size defines the model's accuracy. Figure 2.7 shows a scene represented as voxels without color.

The construction of such a representation from multi-view videos is generally obtained thanks to two families of algorithms. The space-carving algorithms (or shape-from-silhouette) aim to detect voxels belonging to the scene [Kutulakos & Seitz, 1998, Müller et al., 2004]. The coloring algorithms are then used to assign a color to these voxels [Culbertson et al., 1999].

The advantage of the representation by voxels is constant cost access to the desired information. Their rendering, however, is of poor quality, since the cubes become visible if the virtual camera gets too close to objects in the scene.

Meshes modeling use 3D primitives like polygons to define geometric structures [Khodakovsky et al., 2000, Alliez & Desbrun, 2001, Han et al., 2007, Golovinskiy et al., 2009]. A mesh can be defined as a set of polygons, which are usually connected by their edges. The VRML format is very common to describe a mesh. It consists of two tables, which represent the geometry and the connectivity. The geometry of a mesh is a list of all the vertices of the mesh, with associated 3D coordinates. The connectivity of a mesh is a list of all the mesh polygonal

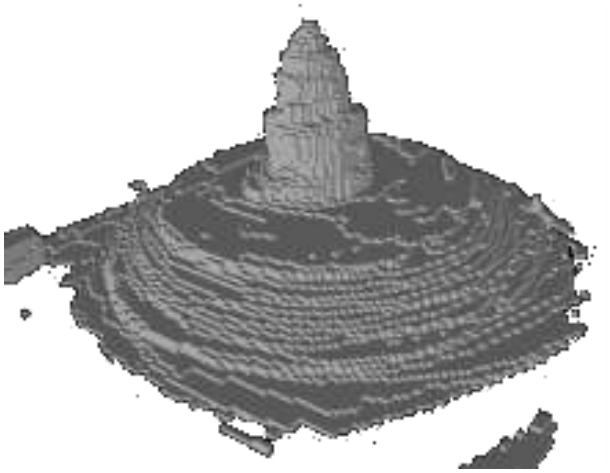
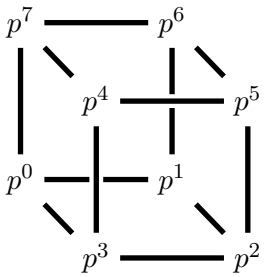


Figure 2.7: A scene modeled with monochrome voxels [Müller et al., 2004].

facets, modeled as circular lists of connected vertices. Figure 2.8 shows a simple example of a cubic mesh described in the VRML format.



(a) Meshes.

id	X	Y	Z
p^0	0	0	0
p^1	0	0	1
p^2	0	1	1
p^3	0	1	0
p^4	1	1	0
p^5	1	1	1
p^6	1	0	1
p^7	1	0	0

(b) Geometry.

id	Nodes
a^0	p^0 p^1 p^2 p^3 p^0
a^1	p^0 p^1 p^6 p^7 p^0
a^4	p^0 p^3 p^4 p^7 p^0
a^2	p^1 p^2 p^5 p^6 p^1
a^3	p^2 p^3 p^4 p^5 p^2
a^5	p^4 p^5 p^6 p^7 p^4

(c) Connectivity.

Figure 2.8: VRML representation of a simple cube mesh.

Many mesh management tools are integrated into MPEG coders, as well as hard-wired into current graphics cards.

2.2 3D video coding

Video compression refers to reducing the amount of data used to represent digital video images. This section describes methods for video coding, adapted to 2D or 3D video sequences.

2.2.1 Introduction

Compression helps reducing the consumption of expensive resources, such as hard disk space or transmission bandwidth. On the downside, compressed data must be decompressed to be used, and this extra processing may be detrimental to some applications.

A multi-view video carries much more information than a classical 2D video. Compression is thus more important for transmission and storage of multi-view data, but it should not degrade too much the rendering quality. Many compression algorithms have been developed, and are

distinguished by their type of internal representation [Lavoué et al., 2008, Smolic et al., 2007a]. All of them operate on the encoder/decoder principle. First, the encoder applies a compression algorithm on the data in order to reduce the correlations and to possibly eliminate some unnecessary information. Then, the decoder applies the appropriate decompression algorithm to the compressed data in order to recover the original sequence. When transmitting over a network, the encoder (server side) is separated from the decoder (client side) through various transmission channels. When storing on a support, the encoder exports multi-view video in a file that can be later decoded by several decoders.

The design of multi-view data compression schemes therefore involves trade-offs between various factors, including the degree of compression, the amount of distortion introduced (if using a lossy compression scheme), and the computational resources required to compress and uncompress the data. Standardized compression formats must satisfy a number of constraints:

Backward compatibility with the previous standardized MPEG format must be ensured. A client who does not have a multi-view decoder must be able to interpret the new format and to render at least one view. Additional information for 3DTV and FTV features must be transmitted as extra information, which are not interpreted by classic television decoders.

Flexibility of the format should be sufficient for rendering on different kinds of 3D devices. Stereoscopic displays require only two views, whereas auto-multiscopic displays need five, eight, or more views. Intrinsic cameras parameters of virtual views may further vary, depending on the observer preferences.

Quality of the rendered video should be comparable to existing HD video standard. If the visual quality of multi-view video is worse than HD video, no user will purchase new equipments. Compression efficiency must be adjustable and should not damage too much the rendering quality.

Ease of deployment is also an interesting criterion of new compression standards. Current infrastructures for TV transmission include cameras for acquisition, signal processing devices, various transmission channels (cables, antennas, satellites, ...) and customers television. A new format must maximize compatibility with the current infrastructures, in order to reduce the equipment cost.

Section 2.2.2 introduces some fundamental concepts of image and video compression.
Section 2.2.3 details some state-of-the-art methods for multi-view video compression.

2.2.2 Concepts of image and video compression

Blocks of images, and also frames of sequences, contain a lot of correlation. Most image and video coders thus employ a predictive coding configuration to reduce this correlation and to obtain a good compression rate.

Image prediction

Image compression schemes frequently split images into blocks of pixels. These adjacent blocks are often highly correlated, especially in homogeneous areas. The goal of the spatial prediction is

to reduce the correlations between adjacent blocks. The prediction error, also called residual data, is encoded using an entropy-coding scheme.

This process is not reversible, and a part of the information is lost. The image is thus distorted during the lossy compression. The induced distortion is measured and minimized to improve the quality of the decoded image.

Distortion measures

Lossy compression methods are evaluated by their compression rate, and by the distortion they introduce in the reconstructed images. The compression rate of major lossy compression methods can be configured to fit the desired decoded image quality.

The perceived distortion in visual content is a very difficult quantity to measure, as the characteristics of the human visual system are complex and not well understood. Various algorithms, so called "objective distortion metrics", have been introduced to approximate the human perception of reconstruction quality, as the MSE, PSNR, or SSIM. In some cases one reconstruction may appear to be closer to the original than another, even though it has a higher computed distortion. One has to be extremely careful with the range of validity of these metrics. They are only conclusively valid when they are used to compare results from the same coder (or codec type) and same content.

The Mean Squared Error (MSE) is the second moment (about the origin) of the error, and thus incorporates both the variance of the estimator and its bias. The MSE of two images I_1 and I_2 (where one of the images is considered a noisy approximation of the other) is computed as the following equation:

$$MSE(I_1, I_2) = \frac{1}{N} \cdot \sum_{p} \left(I_1(p) - I_2(p)\right)^2 \tag{2.1}$$

where N is the number of pixels within the images, and p is a sliding pixel where the squared difference is computed.

For an unbiased estimator, the MSE is the variance. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated.

For color images with three RGB values per pixel, the color image MSE is defined as the average of the MSE of each channel.

A lower MSE would normally indicate that the reconstruction is of higher quality. When the two images are identical, the MSE will be zero.

Peak Signal-to-Noise Ratio (PSNR) is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. The signal represents the original image, and the noise is the error introduced by compression. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.

PSNR is defined via the MSE as the following equation, for two monochrome images I_1 and

I_2:

$$PSNR(I_1, I_2) = 10 \cdot \log_{10} \left(\frac{\omega^2}{MSE(I_1, I_2)} \right) \tag{2.2}$$

where ω is the maximum possible pixel value of the image (typically $\omega = 255$).

Color images are often converted to a different color space (such that YUV) and PSNR is reported against each channel of that color space.

A higher PSNR would normally indicate that the reconstruction is of higher quality. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB. When the two images are identical, the PSNR is infinity.

The structural similarity index (SSIM) is designed to improve on traditional distortion metrics, like PSNR and MSE, which have been proved not to reflect well human eye perception [Wang et al., 2004]. The SSIM metric is calculated on various windows of an image. The distance between two windows W_1 and W_2 of common size $N \times N$ is computed as follows:

$$SSIM(W_1, W_2) = \frac{(2\mu_1\mu_2 + c_1)(2\sigma_{1,2} + c_2)}{(\mu_1^2 + \mu_2^2 + c_1)(\sigma_1^2 + \sigma_2^2 + c_2)} \tag{2.3}$$

where μ_i and σ_i^2 are respectively the average and the variance on window W_i, $\sigma_{i,j}$ is the covariance on windows W_i and W_j, and c_1, c_2 are two variables used to stabilize the division with weak denominator. By default, $c_i = (255k_i)^2$ where $k_1 = 0.01$ and $k_2 = 0.03$.

In order to evaluate image quality, this formula is applied only on luma. The resulting SSIM index is a decimal value between -1 and 1. A higher SSIM would normally indicate that the reconstruction is of higher quality. When the two images are identical, the SSIM will be one. Typically, SSIM is calculated on window sizes of 8×8. The window can be displaced pixel-by-pixel on the image but the authors propose to use only a subgroup of the possible windows to reduce the complexity of the calculation.

A recent work [Bosc et al., 2011b] aims at comparing these metrics with many others and subjective testing. Against all odds, the results of this study show that among all tested objective metrics, PSNR is one of the most correlated with perceptual evaluation but its correlation factor is nevertheless below 40%. One thus has to be extremely careful with the range of validity of these metrics. In this dissertation, only PSNR and SSIM metrics are used.

2D Video compression

Classical video compression methods work with similar principles as image compression methods, but exploit both spatial and temporal correlation. Some parts of the sequence are static, or are moving slowly in the sequence, resulting in a high correlation between consecutive frames. These slow movements are predicted with a motion estimation and compensation technique.

A compressed video is represented as a sequence of compressed frames. Each frame is compressed in one out of three modes, which are Intra mode (I-Frame), Predicted mode (P-Frame), or Bilateral predicted mode (B-Frame)

I-Frame is compressed using only the frame content itself. The frame is divided into small blocks of pixels, and blocks are compressed using spacial prediction.

P-Frame uses one earlier frame in a sequence to predict, and thus compress, the current frame. If the frame contains blocks similar to blocks already decoded, the system simply issues a short command that copies that part of the previous frame, bit-for-bit, into the new one.

B-Frame is predicted from more than one earlier and/or later frames. The principle is similar than for P-Frames except than the comparison is performed with several other reference frames, some of which are later in the video sequence. This compression mode provides higher compression rate, but requires to previously decode any reference frame.

A video compression scheme should adapt the sequencing of each kind of compressed frame, to fit the main use of the video. Predicted compression mode works well for programs that will simply be played back by the viewer, but can cause problems if the video sequence needs to be edited. For example, the decoding of a single B-frame may require the previous decoding of 2 other frames, each requiring the decoding of 2 other frames, etc... If one of these frames is poorly decoded, due to errors during transitions, all dependent frames will also contain artifacts.

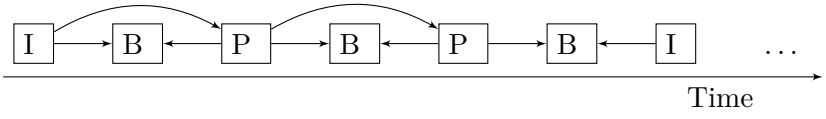
A classical video compression scheme divides the video sequence into small Groups Of successive Pictures (GOP), which are compressed independently of the others GOP. A GOP always begins with an I-frame. Afterward several P-frames follow, separated with several B frames (see figure 2.9). A few video coders allow for more than one I-frame in a GOP.

The I-frames contain the full image and do not require any additional information to be reconstructed. Therefore any error within the GOP structure is corrected by the next I-frame. The more I-frames the video stream has, the more editable it is. However, having more I-frames increases the stream size. In order to save bandwidth and disk space, videos often have only one I-frame per GOP of 12 pictures.

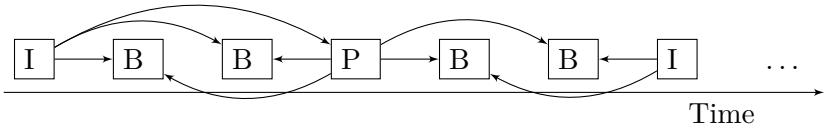
The structure of a GOP may be characterized by two numbers N and M. The first one tells the distance between two anchor frames (I or P). The second one tells the distance between two full images (I-frames): it is the GOP length. Instead of the M parameter one can use the maximal count of B-frames between two consecutive anchor frames. Some classical GOP structures are presented in Figure 2.9. A classical GOP structure is characterized by $N = 3$ and $M = 12$.

The Moving Picture Experts Group (MPEG) was established in 1988 by the International Organization for Standardization (ISO). It is responsible for developing international standards for compression, decompression, processing and coding of video, audio and multimedia, for a wide range of applications.

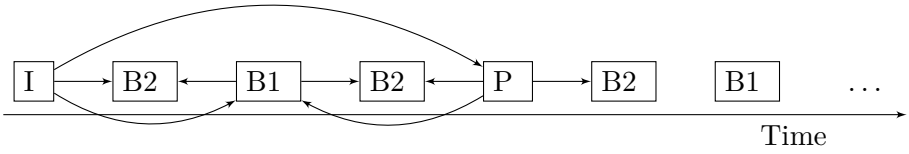
H.264/MPEG-4 AVC is the current standard, proposed in 2002 by the Joint Video Team (JVT) as a collaborative work of MPEG and VCEG (for Video Coding Experts Group). AVC (for Advanced Video Coding) is the MPEG Part-10 extension, described in ISO/IEC 14496-10, which supports various macro-blocks sizes, and multiple reference frames for temporal prediction. MPEG and VCEG are now working jointly on the HEVC (High Efficiency Video Codec) which is planned to be standardized in 2013.



(a) IBP scheme (N=2 M=6)



(b) IBBP Scheme (N=3 M=6)



(c) IB^{2}P scheme (N=4 M=8 Hierarchical B)

Figure 2.9: Classical GOP structures. Each arrow represents a prediction. Anchor frames can be separated by one 2.9(a), two 2.9(b), or more B-frames. B-Frames can be organized in a hierarchical structure 2.9(c).

2.2.3 Multi-view video compression

When many cameras are observing the same scene, captured images at a same time are highly correlated. These inter-views correlations can also be exploited, in addition to spatial and temporal correlations. In 2001, the MPEG group has created the 3D Audio-Visual subsidiary (3DAV), to specify new coding standard for 3DTV and FTV. One must distinguish three families of multi-view video compression algorithms, sorted by increasing efficiency, but also by increasing complexity.

The Simulcast Coding scheme is the most simple multi-view video compression scheme. Each captured view is compressed independently of the others, as a classical 2D video. Temporal correlations are thus eliminated, but inter-view correlations are not exploited at all. This coding scheme serves as a reference to evaluate other compression algorithms.

The Multi-View Coding scheme (MVC) is an improvement on the previous scheme, which was designed to reduce both inter-view correlations and temporal correlations. This coding method reuses techniques presented in Section 2.2.2 by considering the differences between views as moving objects. Inter-view correlations are thus coded with motion compensation techniques, by using motion vectors. Specificities of these motion vectors are not exploited.

The Multi-View plus Depth coding scheme (MVD) exploits the consistency between views by encoding the scene geometry as a depth map. In addition to a reference view, this depth map is used to predict each additional views.

These three multi-view video compression families are detailed in next sections.

Simulcast Coding scheme

The Simulcast Coding scheme consists in a independent compression of each input view. Figure 2.10 presents the simulcast scheme for three input views, where each view is coded with the IBBP GOP structure.

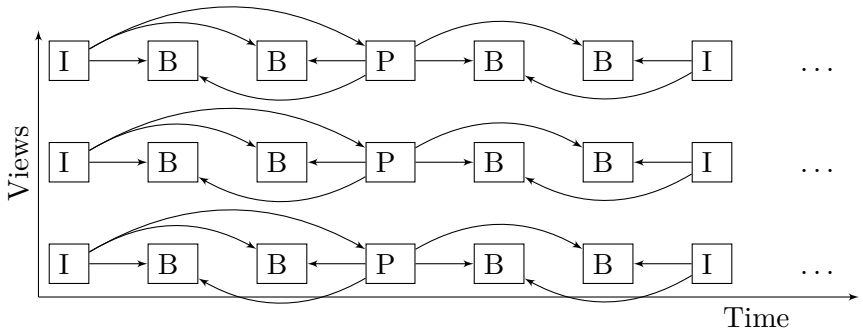


Figure 2.10: Simulcast Coding scheme. Each view is coded independently of the others.

This Simulcast Coding scheme has many capabilities. It is fully compatible with current network architecture and displays; The desired view can be decoded easily, without the need for decoding any other view; The coding process and decoding process can be parallelized on as many processors as views; The number of transmitted views can be adapted to the channel capacity.

The main limitation of the Simulcast scheme is its low compression rate. Each compressed view has the same size, and thus, the total size of a multi-view video increases linearly with the number of views. The bit-rate required for the transmission of an N-views video is N-times the bit-rate required for the transmission of a single view.

One of the first experience of real-time acquisition, transmission and rendering of multi-view videos was conceived in 2004 into the Mitsubishi Electric Research Laboratories (MERL) [Matusik & Pfister, 2004]. The acquisition is performed by 16 cameras, aligned horizontally. Each camera have a resolution of 1300 px. Each color video is encoded independently with the MPEG-2 standard.

The encoding step requires the parallelization of 8 servers, each one dealing with only 2 views. The transmission step is performed by 8 direct gigabyte connections, which connect the 8 servers to 8 rendering computers. 3D rendering is obtained by the calibration of 16 projectors, whose images are framed on a large lenticular screen. A direct matching between cameras and projectors would have been possible, but it would have been sensitive to misalignments, and it would have required the same number of projectors as the number of cameras. Therefore, the Unstructured Lumigraph rendering method was used [Buehler et al., 2001].

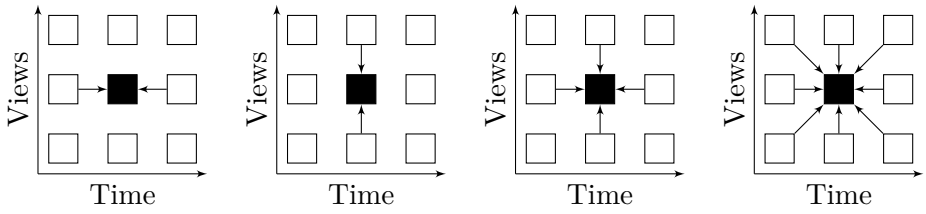
This first demonstration of the full 3DTV chain enlightens some of the encountered difficulties. The transmission step, performed by 8 direct connections, requires rate capacities well beyond the current Internet connection speed.

Multi-View Coding (MVC)

The Multi-View Coding scheme (MVC) is an extension of H264 which improves the compression rate of multi-view videos, by including inter-view prediction.

In classical video compression, some frames are predicted by some earlier and/or later frames, by the use of a motion field. In the same way, the MVC scheme uses a disparity field to estimate a frame of a view, from some frames of the other views [Kurutepe et al., 2006, Kurutepe et al., 2007, Flierl et al., 2007, He et al., 2007, Smolic et al., 2004, Zwicker et al., 2007].

Figure 2.11 presents some of the various prediction schemes possible for any frame. In Figure 2.11(a), the black frame is only predicted from previous and next temporal frames of the same view, as for Simulcast coding scheme. In Figure 2.11(b), the black frame is only predicted from the frame of the left and right views, captured at the same instant. Depending on the frame content, one prediction scheme may be more efficient than another. Both temporal prediction and inter-view prediction can thus be used simultaneously (see Figure 2.11(c) and 2.11(d)), which improves the compression ratio, but also increases the coding and decoding complexity.



(a) Temporal (b) Inter-view (c) Quadruple (d) Eightfold

Figure 2.11: Various prediction schemes for MVC frames [Merkle et al., 2007]. Both temporal predictions 2.11(a) and inter-view predictions 2.11(b) can be associated in quadruple 2.11(c) or eightfold 2.11(d) prediction scheme.

Compression methods based on MVC have limited compression ratio, for an increased decoding complexity (in terms of time and memory usage). Compared to a Simulcast compression scheme, where each view is encoded independently by H.264/AVC method, the MVC compression method presents the following results [Merkle et al., 2007]: At equivalent bitrate, MVC provides gains of about 1 to 1.5 dB in PSNR. For the same visual quality, MVC reduces the required bitrate of about 15% to 30%. Decoding a single frame may require the prior decoding of many adjacent views and memory storage of many other frames.

In order to reduce the coding and decoding complexity, while preserving an efficient compression rate, inter-view prediction can be used only for I-frame prediction. In a classical sequence coded in simulcast, one can observe that I-Frames represent only 8% of the number of frames, but their data represents more than 20% of the total size of the video. Using an inter-view prediction scheme to predict only these I-Frames, reduces by nearly 20% the size of the video, without increasing too much the decoding complexity. Each view can be decoded with the only knowledge of I-Frames of other views.

Figure 2.12 presents some of the various prediction schemes possible for I-frames. These inter-view prediction schemes are similar to temporal prediction schemes, presented in Figure 2.9

In [Merkle et al., 2007], the authors present their results for multi-view video compression,

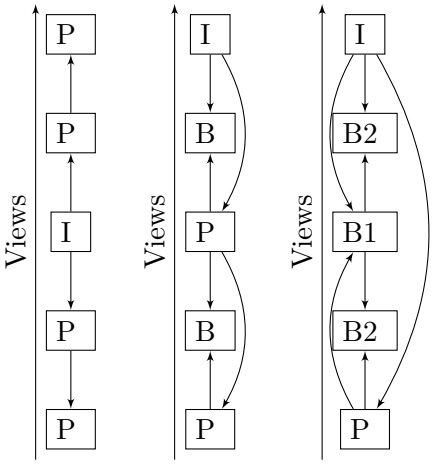


Figure 2.12: Various inter-view prediction schemes for I-frames [Merkle et al., 2007].

using the MVC scheme. Two main families were compared: AS family refers to MVC scheme where All frames are predicted, using inter-view and temporal predictions. KS family refers to MVC scheme where only Key frames of additional views are predicted, using inter-view predictions.

Figure 2.13 shows an example of a compression scheme of both KS and AS families, where temporal prediction follows a B hierarchical prediction scheme. Figure 2.13(a) shows an AS compression scheme, where all frames are predicted using an inter-views IPP prediction scheme. Figure 2.13(b) shows a KS compression scheme, where GOP key frames are predicted using an inter-views IBP prediction scheme.

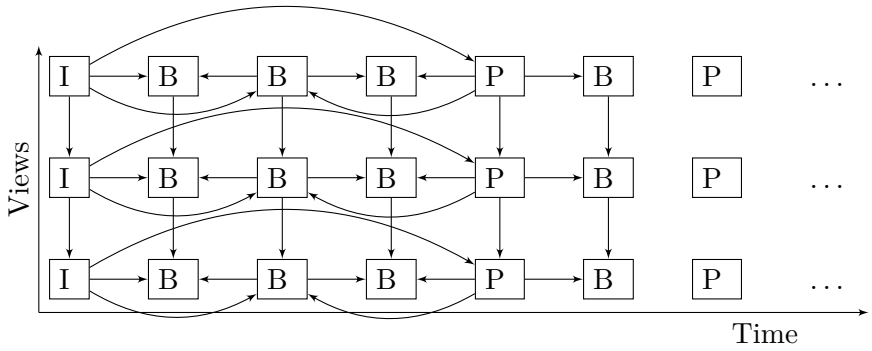
Results presented in [Merkle et al., 2007] show that a KS scheme improves the compression rate by 17%, compared to Simulcast compression at same quality. Some AS schemes provide a slightly better compression rate, but the significant increase of decoding complexity contrasts with the small gain in compression. These results are partially reproduced in Figure 2.14.

The main limitation of the MVC coding scheme is the prediction of additional views, based on a motion flow. This motion flow does not exploit the scene geometry, and should thus be computed and coded for each additional view and for each frame.

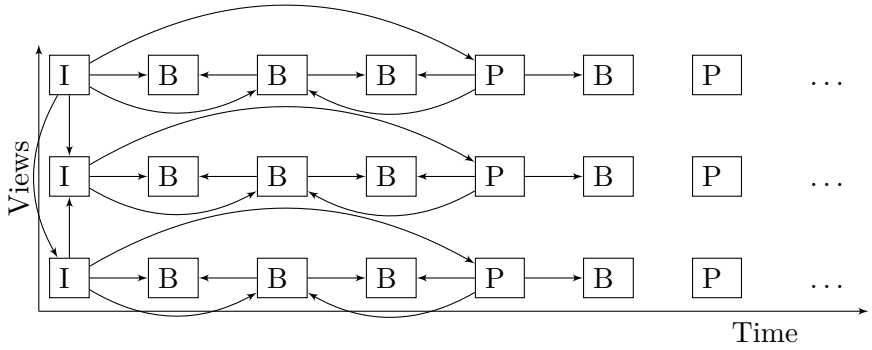
Multi-View plus Depth coding (MVD)

Inter-view correlations are due to the geometry of the scene, and are thus consistent along time. Instead of using a motion field to predict each additional view, MVD methods encode a geometric model of the scene, often consisting of a depth map. This geometric model, in addition to a reference view, allows the decoder to efficiently predict every additional view by using Depth-Image-Based Rendering methods (DIBR). For additional views, only the prediction error (named residual information) is thus coded [Shimizu et al., 2007, Yamamoto et al., 2007].

Figure 2.15 presents the classical MVD encoding scheme, as proposed in [Shimizu et al., 2007]. One view is considered as the reference view (or base view), and is coded like classical 2D video, with only temporal and spatial predictions. Each other view is then used jointly to the reference view to estimate a view-dependent geometric model of the scene. This views-dependent geometric



(a) AS-IPP format



(b) KS-IBP format

Figure 2.13: Various MVC schemes [Merkle et al., 2007]. Inter-view prediction may be used to predict all frames 2.13(a), or only key frames 2.13(b).

model is coded, and it is also used with the reference view to predict the corresponding additional views. The prediction error is finally computed for each additional view and coded within the data flow.

Backward compatibility with classical 2D video is ensured by separating the reference view, coded as classical 2D video, and the additional data. A classical 2D decoder is thus able to recover the reference view by ignoring the extra data. The view dependent geometry is almost always represented as a sequence of depth maps, also coded as classical monochromatic videos.

In [Smolic et al., 2007b], the proposed experiments show that the geometric model of the scene can be greatly simplified without compromising the visual quality of the final rendered video. The authors thus propose that the bit-rate allocated to the depth map transmission should be between 10% and 20% of the bit-rate allocated to the reference view. In some others experiments [Bosc et al., 2010, Bosc et al., 2011a], the test protocol is slightly different and the observed optimal bit-rate reserved to depth information should be near 50% of the total bit-rate. These results show that the optimal compression ratio of the geometric information highly depend on the final application and the rendering method.

MVD without depth transmission

A MVD alternative solution is to not directly encode the geometry of the scene. This geometry is estimated by the encoder to calculate and encode the residual errors of certain views, but it is

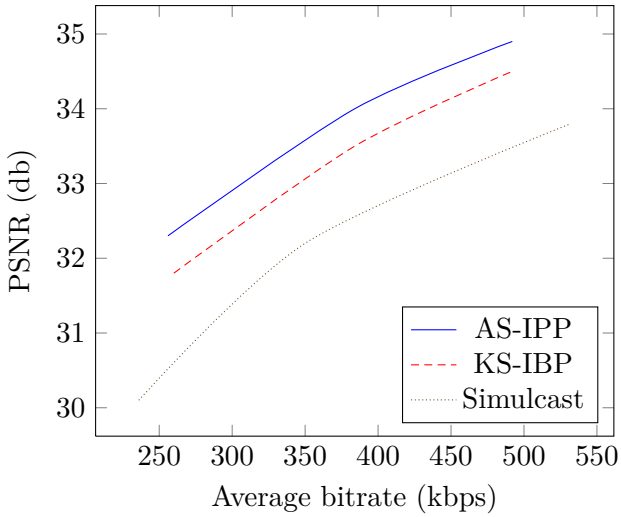


Figure 2.14: Rate-distortion curve for some MVC prediction scheme. (Results from [Merkle et al., 2007] on "Ballet" test sequence)

not transferred to the decoder. Instead, this geometry is re-estimated by the decoder from the views already received. To guide the estimation of disparity made by the decoder, the encoder transmits at least two views encoded with MVC, together with any information correction. Other views can be encoded using the geometric information that will be re-estimated by the decoder. This coding scheme is shown in Figure 2.16.

This alternative scheme allows the decoder to easily decode two views, while preserving the ability to decode more of them, if necessary. The on line estimation of the geometry is very approximate, which may result in annoying artifacts and inconsistencies on predicted views. This method is thus particularly suitable for stereo-vision, without prohibiting multi-scopic displays or FTV use.

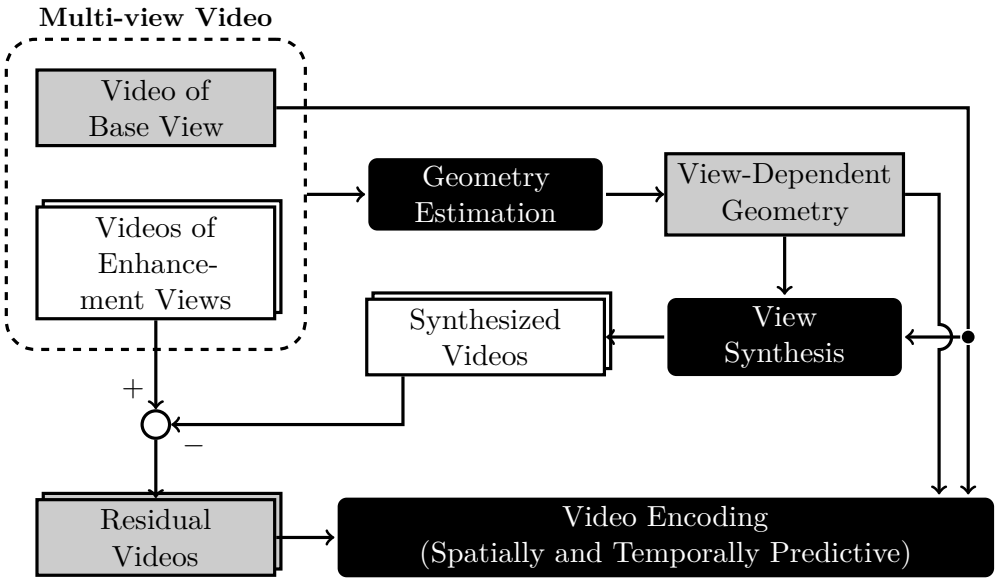


Figure 2.15: MVD coding scheme [Flierl et al., 2007].

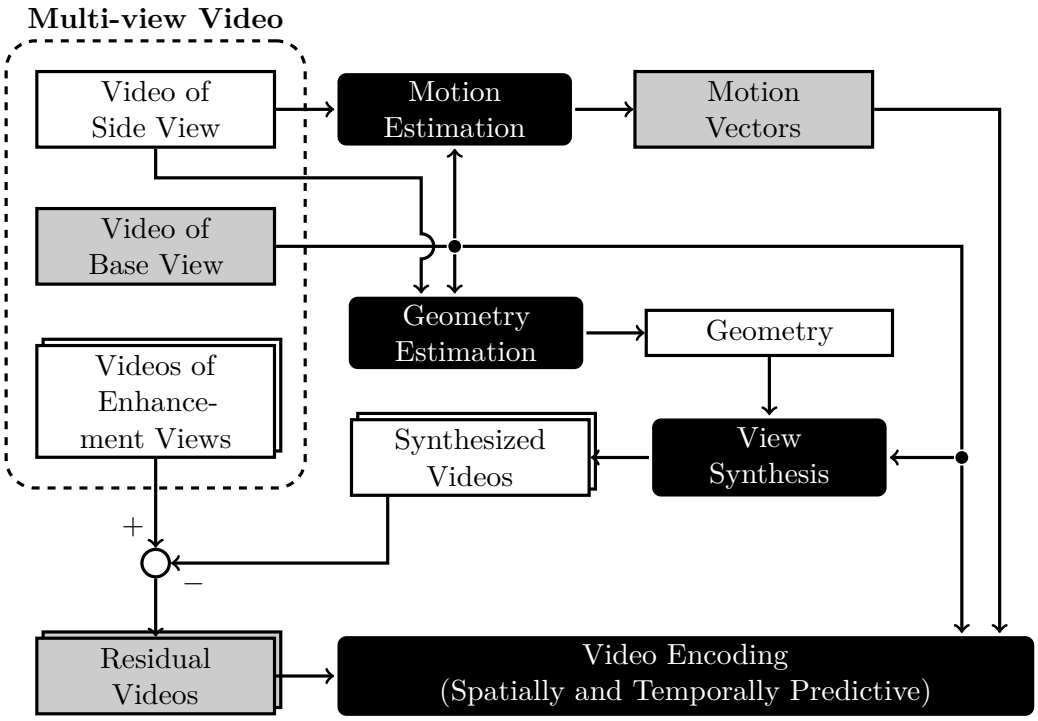


Figure 2.16: MVD coding scheme without explicit transmission of depth information. The base view is coded as classical 2D video. The side view is coded with motion compensation from the base view. These two views are used to approximate the geometry of the scene, which helps to predict the additional views.

Part I

Virtual view synthesis

As explained in the previous chapter, a multi-view video carries much more information than a classical 2D video. In order to reduce the size of the data to be transmitted, one classical solution is to not transmit all captured views, but just enough information to allow the client to re-synthesize its desired views. The efficiency of the view synthesis method is thus an important parameter for multi-view video compression.

The synthesis of virtual views is a classical problem in computer vision applications. This problem is encountered in applications such as robot navigation, object recognition, free-viewpoint navigation, or stereoscopic visualization. Many rendering algorithms have been developed and are classified rather as Image-Based Rendering (IBR) techniques, Geometry-Based Rendering (GBR) techniques, or hybrid Depth-Image-Based Rendering (DIBR) techniques.

DIBR techniques are a key technology in advanced three dimensional television system (3DTV System). These techniques are based on one video stream and the corresponding depth stream which assigns to every pixel of the video stream a distance value to the camera. DIBR takes the pixels of the original view and builds a simple 3D model of the scene by projecting them into the 3D world according to their depth values specified by the depth stream. This 3D model is then projected onto the image plane of a virtual camera. This process is called 3D image warping.

Any nearby view can be synthesized by mapping pixel coordinates one by one according to its depth value. From this information, DIBR is able to render (almost) arbitrary virtual views, like a left-eye and a right-eye video stream for stereoscopic display. The main advantage of view synthesis at decoder side is that the relevant parameters of the stereo video stream are not predefined at recording time, but can be set on the receiver-side. The user can watch more comfortable 3D video by adjusting depth perception and parallax to his individual preference (like color and contrast in nowadays TVs).

This part describes DIBR techniques for synthesizing virtual views from video plus depth input sequences. Both intermediate view synthesis and virtual view extrapolation problems are addressed. Intermediate view synthesis refers to the synthesis of a virtual view from multiple video plus depth sequences. Information from input sequences is combined to synthesize the desired view targeting the best possible rendering quality. Virtual view extrapolation refers to the synthesis of a virtual view from a single input view plus depth sequence.

Chapter 3 is a review of state-of-the-art methods for DIBR, and introduces the classical warping algorithm. The three types of artifacts it can generate are discussed, which are disocclusions, cracks, and ghosting artifacts. Some algorithms are then detailed to avoid each of these artifacts. Finally, a full DIBR method is presented, using the previous presented algorithms to synthesize virtual views with a satisfying visual quality.

Chapter 4 presents our contributions for virtual view synthesis with DIBR techniques. The Joint Projection Filling method (JPF) is first introduced as a projection method, which handles disocclusions problems during the projection to make use of the scene geometry. This JPF algorithm is then used in two DIBR methods, designed for either intermediate view interpolation or virtual view extrapolation. The intermediate view interpolation method uses an optical flow estimator to reduce the blur and increase the level of details in the rendered views. The virtual view extrapolation method uses a depth-aided inpainting method to fill in large disocclusions while reproducing texture patterns.

Chapter 3

Background work on DIBR: State-of-the-art

Many rendering algorithms have been developed and are classified rather as Image-Based Rendering (IBR) techniques or Geometry-Based Rendering (GBR) techniques, according to the amount of 3D information they use.

IBR techniques use multi-view video sequences and some limited geometric information to synthesize intermediate views. These methods allow the generation of photo-realistic virtual views at the expense of limited virtual camera freedom [Shum et al., 2007, Debevec et al., 1998, Fitzgibbon et al., 2005, Chan et al., 2007, Shum & Kang, 2000].

GBR techniques require detailed 3D models of the scene to synthesize arbitrary viewpoints. They are sensitive to the accuracy of the 3D model, which is difficult to estimate from real multi-view videos. GBR techniques are thus more suitable for rendering synthetic data.

Depth-Image-Based Rendering (DIBR) techniques are IBR methods which include hybrid rendering techniques between IBR and GBR algorithms [Shum & Kang, 2000, Zhang & Chen, 2004, Furihata et al., 2010, Ndjiki-Nya et al., 2011, Nguyen et al., 2009]. DIBR methods are based on warping equations, which project a reference view onto a virtual viewpoint. Each input view is defined by a "color" (or "texture") map and a "depth" map, which associates a depth value to each image pixel. These depth maps can be estimated from multi-view video sequences by using a disparity estimation algorithm [Hartley & Zisserman, 2004, Morvan, 2009, Sourimant, 2010a, Tanimoto et al., 2009]. In this manuscript, the input depth maps are assumed to be known and provided as input data.

This chapter presents the projective geometry equations and some state-of-the-art DIBR methods. The advantages and limitations of each algorithm are detailed.

Section 3.1 presents some background information on projective geometry, which is necessary to understand the methods developed in this work. These equations are the theoretical foundations of the warping algorithm, which is able to change the point of view of an image whose depth is known. Warping equations are focused on the geometric relation between two adjacent cameras.

Section 3.2 introduces the use of warping equations to synthesize virtual views, and exhibits the artifacts generated by this method, which are cracks, ghosting and disocclusions.

Section 3.3 presents the backward projection which avoids cracks and others sampling artifacts.

Section 3.4 details some specific treatments on object boundaries for avoiding ghosting artifacts.

Section 3.5 shows how to handle disocclusion issues by using either additional views, or inpainting methods.

3.1 Perspectives on projective geometry

Projective geometry is an efficient mathematical framework used in computer vision, which nicely models perspective projection. It is used in scene reconstruction, robotics, space positioning, object recognition, mosaicking, image synthesis, analysis of shadows, etc.

Consider $\mathcal{R}_0$ the coordinate system of the real world, and $M_0 = (X, Y, Z)$ the three coordinates of a point in this coordinate system. If a camera C is directed to this 3D point M_0, the captured image will contain a representation of this point as a 2D pixel $p_C = (u, v)$ in the image coordinate system $\mathcal{I}_C$.

A real acquisition process is often modeled by the equations of the pinhole camera [Hartley & Zisserman, 2004]. The projection equations link 3D coordinates in the real world coordinate system, and 2D coordinates in the image coordinate system. This section presents the computational steps required to convert 3D points into 2D pixels, then explains how to reverse these equations and recover the 3D coordinates from a 2D pixel and its depth value.

Section 3.1.1 introduces the homogeneous coordinates, a mathematical tool used to linearize the equations of the perspective projection.

Section 3.1.2 describes the perspective projection and introduces the equations of the pinhole camera model.

Section 3.1.3 explains the method to reverse the equations of perspective projection, and thus recover the 3D coordinates of a point from its pixel representation in an image plus an additional depth information.

Section 3.1.4 presents a well known optimization where some coefficients are precomputed.

3.1.1 Homogeneous coordinates

Homogeneous coordinates are used in projective geometry such as Cartesian coordinates are used in Euclidean geometry [Richter-Gebert & Richter-Gebert, 2011]. In homogeneous coordinates, the equations of perspective projection become linear and can thus be described as matrix operations. All points, including points at infinity, can be represented using finite homogeneous coordinates. Four homogeneous coordinates are required to specify a point in the projective 3D space $\mathbb{R}^3$. The homogeneous coordinates have the advantage to handle both rotations and translations, with simple matrix multiplications. If the homogeneous coordinates of a 3D point are multiplied by a non-zero scalar then the resulting coordinates represent the same 3D point.

The 3D coordinates M_{3D} of a point M can be converted into homogeneous coordinates

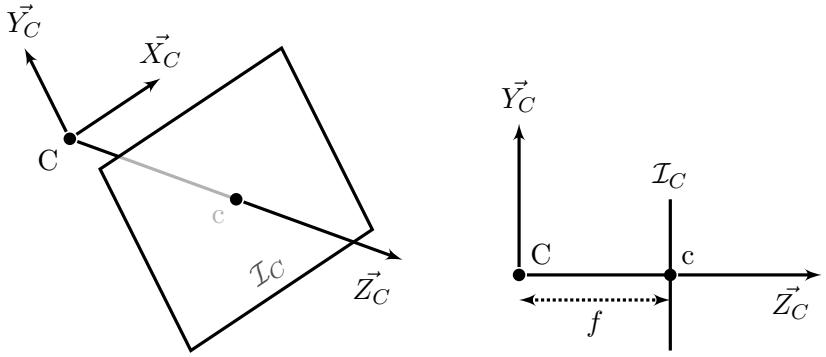
M_{Homo}.

$$M_{3D} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \Longrightarrow \quad M_{Homo} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

$$M_{Homo} = \begin{pmatrix} X' \\ Y' \\ Z' \\ W' \end{pmatrix} \quad \Longrightarrow_{\text{If } W' \neq 0} \quad M_{3D} = \begin{pmatrix} X'/W' \\ Y'/W' \\ Z'/W' \end{pmatrix} \tag{3.2}$$

3.1.2 Perspective projection

We describe the image acquisition process known as the pinhole camera model, which is regularly employed as a basis in this thesis. A pinhole camera is described by its optical center C (also known as the camera projection center) and the image plane $\mathcal{I}_C$. The distance, from the optical center C to the image plane is called the focal length f. The line Z_C passing through the optical center and perpendicular to the image plane is called the optical axis or principal ray of the camera.



(a) 3D view (b) Side view

Figure 3.1: Pinhole camera coordinate system $\mathcal{R}_C$ with its image plane coordinates system $\mathcal{I}_C$.

The pinhole camera model denotes a simple transformation from the 3D-world coordinates system $\mathcal{R}_0$ onto a 2D-image plane coordinates system $\mathcal{I}_C$. This transformation can be decomposed in three steps:

- Changing the 3D coordinate system from the 3D-world coordinate system $\mathcal{R}_0$ to the 3D-camera coordinate system $\mathcal{R}_C$.
- Projecting the 3D point expressed in $\mathcal{R}_C$ onto the image plane.
- Shifting the origin of the image plane to the optical center c and transforming the coordinate units of $\mathcal{R}_C$ to pixel units of $\mathcal{I}_C$.

The next sections detail each of these three steps.

From $\mathcal{R}_0$ to $\mathcal{R}_C$

The 3D coordinate transformation from the 3D-world coordinate system $\mathcal{R}_0$ to the 3D-camera coordinate system $\mathcal{R}_C$ requires knowledge of the extrinsic parameters of the camera C. The

extrinsic parameters describe the position and orientation of the camera with respect to the world coordinate system $\mathcal{R}_0$.

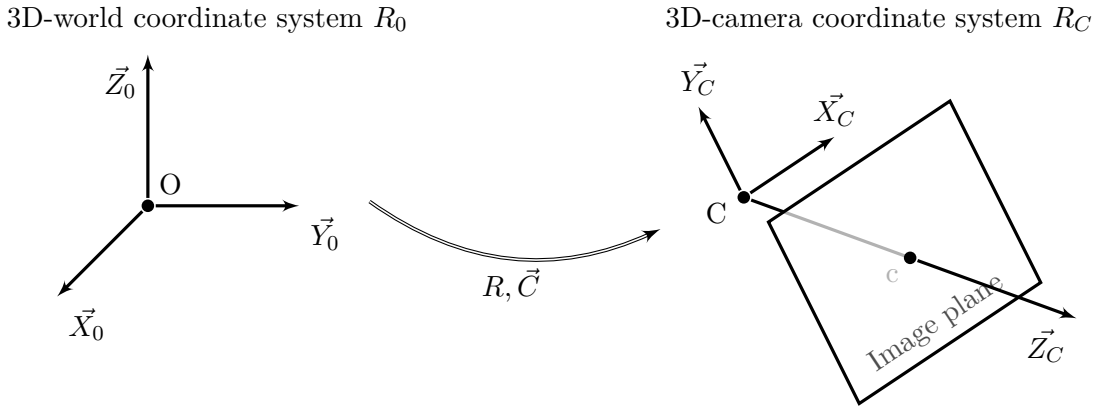


Figure 3.2: World to camera coordinates transformation.

These extrinsic parameters include a 3×1 position vector $\vec{C} = (C_X, C_Y, C_Z)$ and the orthogonal 3×3 rotation matrix R. The relation between coordinates $M_0 = (X, Y, Z)$ of a 3D point in R_0 and coordinates $M_C = (X', Y', Z')$ of the same 3D point in R_C, can be written as:

$$\mathbf{M_C} = R \cdot (M_0 - C) \tag{3.3}$$

When defining the 3×1 translation vector $t = -R \cdot C$, this equation is re-written in homogeneous coordinates as:

$$\mathbf{M_C} = \left(\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot M_0 \tag{3.4}$$

From $\mathcal{R}_C$ to image plane

The perspective projection is the projection of the 3D space on the plane $Z = 1$, according to the family of lines that pass through 0. This perspective projection is used to model a camera, where light rays are captured by a light sensor.

Let M_C be a point in the camera coordinate system $\mathcal{R}_C$, its perspective projection point $m = (x, y)$ onto the plane $Z = 1$, satisfies the following equation in homogeneous coordinates:

$$m \equiv \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \equiv \begin{pmatrix} z * x \\ z * y \\ z \end{pmatrix} = I_{3,4} M_c \tag{3.5}$$

where $I_{3,4}$ is the 3×4 eye matrix.

This perspective projection, as all other projections, is not invertible without additional information.

From image plane to pixel coordinate system $\mathcal{I}_C$

The pinhole camera model assumes that the origin of the image coordinate system, corresponding to the principal point offset (c_x, c_y), is located at the center of the image, which means $(c_x, c_y) = (0, 0)$. However, in general, most of the current imaging systems define the origin at the top-left pixel of the image. A coordinate system conversion is thus necessary.

The size of pixels (the distance units in the image coordinate system $\mathcal{I}_C$) generally does not correspond to the metric of coordinate units $\mathcal{R}_0$. The distance units should also be converted.

Performing those two operations requires knowledge of the intrinsic parameters of the camera.

By opposition to the extrinsic parameters that describe the external position and orientation of the camera, the intrinsic parameters indicate the internal camera parameters. The intrinsic parameters describe the properties of the lenses and the charged-coupled device (CCD) chips within the camera, such as focal length f, pixel size (k_x, k_y) and center of projection (u_0, v_0).

In order to simplify the equations, these parameters are organized as the following matrix K:

$$K = \begin{pmatrix} f \cdot k_x & 0 & u_0 \\ 0 & f \cdot k_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

By defining this matrix K, the origin shifting of the image plane and the distance metric conversion are performed by a simple matrix product. The pixel $p = (u, v, 1)$ corresponding to the point $m = (x, y, 1)$ in the image plane is thus defined as follows:

$$p = K \cdot m \tag{3.7}$$

Summary

Consider $M_0 = (X, Y, Z, 1)$ the homogeneous coordinates of a point in the real world coordinate system $\mathcal{R}_0$ and the pixel p representing this 3D point in the captured image coordinate system $\mathcal{I}_C$.

The perspective projection of the point M_0 is computed as follows:

$$\begin{aligned} p &\equiv I_{3,4} \cdot \left(\begin{array}{ccc|c} & K \cdot R & & K \cdot t \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot M \\ &\equiv I_{3,4} \cdot P \cdot M \end{aligned} \tag{3.8}$$

where $I_{3,4}$ is the 3×4 eye matrix. P is named the 4×4 projection matrix.

The parameters of this model can be estimated through a process called camera calibration, which is based on the analysis of image features (lines, points, corners, etc). In more complex models, errors resulting from many properties and artifacts of cameras (misaligned lenses, mispositioned CCD chip, lens distortion, etc) are taken into account.

3.1.3 Reverse equations

The reverse problem is to recover the coordinates of a 3D point M_0 in the world coordinate system, from its pixel representation p in the image coordinate system.

The projection matrix P, presented is equation (3.8), is invertible, and its inverse matrix is P^{-1}:

$$P^{-1} = \left(\begin{array}{ccc|c} & R^\top K^{-1} & & -R^\top t \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{3.9}$$

Because of the $I_{3,4}$ matrix, the projection equations are however not directly reversible, and the inversion requires an additional value, which is the depth of each pixel. This depth information can either be given as Z_C, the third coordinate of the 3D point in R_C, or as R_0, the third coordinate of the 3D point in R_0.

When the depth Z_C of a pixel in the camera coordinate system in known, the 3D point coordinate can be computed as follows:

$$M_0 \equiv \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{pmatrix} = P^{-1} \begin{pmatrix} Z_C \cdot u \\ Z_C \cdot v \\ Z_C \cdot 1 \\ 1 \end{pmatrix} \tag{3.10}$$

More often, only the depth Z_0 of the pixel in the world coordinate system is known. The depth Z_C of the pixel in the camera coordinate system should then be computed, using the third line of the equation (3.10).

$$Z_0 = Z_C \left(P_{3,1}^{-1} u + P_{3,2}^{-1} v + P_{3,3}^{-1}\right) + P_{3,4}^{-1} \tag{3.11}$$

This equation could also be written as follows:

$$\begin{pmatrix} Z_C \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_p & \beta_p \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} Z_0 \\ 1 \end{pmatrix} \tag{3.12}$$

with α_p and β_p the correct scalar numbers, depending on the pixel position p.

3.1.4 Optimization

When using these inverse equations with image sequences, one can observe that for a given pixel, its position does not vary along time. Only its depth is variable. It is thus interesting to isolate constants from varying values, in order to pre-compute some terms. Reverse projection equations can thus be written as follows:

$$
\begin{aligned}
M_0 &= \begin{pmatrix} R^\top K^{-1} \cdot p & -R^\top t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} Z_C \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} R^\top K^{-1} \cdot p & -R^\top t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha_p & \beta_p \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} Z_0 \\ 1 \end{pmatrix}
\end{aligned}
\tag{3.13}
$$

3.2 Forward projection

DIBR methods are based on warping, which is the projection of a reference view onto a virtual viewpoint. 3D image warping maps two views, pixel by pixel, according to each pixel depth value. In other words, 3D image warping transforms pixel locations according to depth values.

In this section, we introduce the concept of forward projection, a DIBR technique which enables the generation of virtual views.

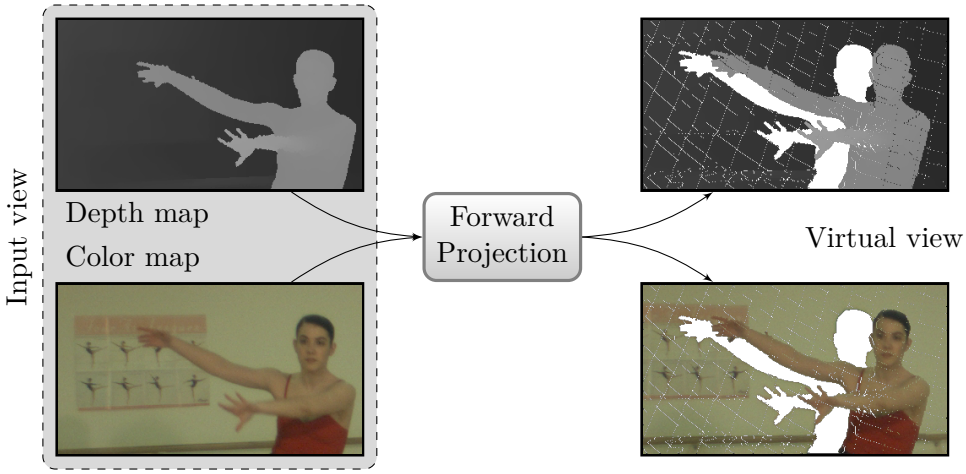


Figure 3.3: Forward projection scheme. Each depth-pixel is warped from the reference viewpoint to the virtual viewpoint.

Applying the warping algorithm on each pixel of a captured view results in a novel image, as observed from a virtual viewpoint. In order to synthesize a virtual viewpoint, the forward projection algorithm thus requires an input view, its depth map, and the input and desired viewpoint cameras parameters. Figure 3.3 shows the forward projection scheme and its rendering results.

(a) Disocclusions. (b) Cracks. (c) Ghosts.

Figure 3.4: Visual artifacts in virtual views synthesized with forward projection method.

Directly applying warping equations to each image pixel may cause some artifacts in the synthesized view, like disocclusions, cracks and ghosting artifacts, as shown in Figure 3.4. Disocclusions are areas occluded in the reference viewpoint and which become visible in the virtual viewpoint, due to parallax effect. Cracks are small disocclusions, mostly due to texture re-sampling. Ghosts are artifacts due to projection of pixels that have background depth and mixed foreground/background color.

In the next sections, we describes methods introduced in the literature which cope with these artifacts. Cracks and others sampling artifacts are handled by the backward projection. Ghosting artifacts are removed through a boundaries detection. Disocclusions are filled in with image inpainting techniques.

3.3 Backward projection for cracks handling

The backward projection method is the commonly used technique to avoid cracks and other sampling artifacts in the synthesized view [Mori et al., 2009, Ndjiki-Nya et al., 2011, Tsung et al., 2009]. Unlike the forward projection scheme, where each pixel from the reference image are warped onto the virtual viewpoint, the backward projection scheme warps each pixel from the virtual view onto the reference viewpoint in order to compute its color.

In order to warp pixels from the virtual view onto the reference viewpoint, the method requires the knowledge of the depth value of every pixel of the virtual viewpoint. The depth map of the virtual view is thus commonly computed with a prior-forward projection of the reference depth map.

The backward projection scheme is thus composed of three distinct steps, as shown in Figure 3.5. The reference depth map is first projected onto the virtual viewpoint in order to compute the depth map of the virtual image view. This forward projection generates many artifacts like cracks and disocclusions. The virtual depth map is then filtered to avoid sampling artifacts and to assign a depth value to all pixels, including pixels inside cracks. Finally, every pixel of the virtual view is warped back onto the reference viewpoint, thanks to the computed virtual depth map. Backward warping of each pixel results in a floating coordinate point in the reference image plan, whose color is computed as bi-linear interpolation of the reference view.

The virtual depth map filtering is the critical step of the backward projection method. Various filters have been proposed to avoid visual artifacts.

The projected depth map contains cracks where the depth value is unknown or wrong. The common solution is to perform a median filtering on the projected depth map to fill these cracks.

The projected depth map is sometimes bilateral filtered to reduce sampling artifacts by smoothing, while preserving edges and depth discontinuities.

In [Do et al., 2009], the authors propose to reduce computational complexity by performing backward projection only for pixels labeled as cracks, i.e. pixels whose depth values are significantly modified by the filtering step. In [Nguyen et al., 2009], the authors propose an occlusion removal algorithm based on the epipolar direction, followed by a depth-color bilateral filtering, in order to handle disocclusions in the depth map.

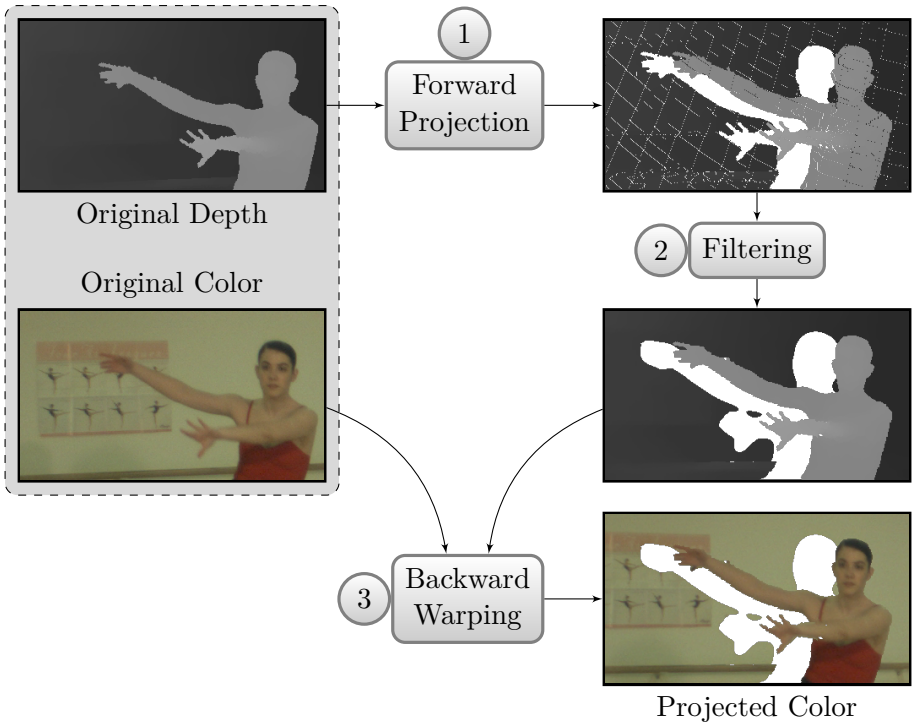


Figure 3.5: Backward projection scheme. First, the input depth map is projected onto the virtual viewpoint. Second, the resulting depth map is filtered to avoid cracks and ghosting artifacts. Third, the filtered depth map is projected back onto the reference viewpoint to find the color of each pixel.

3.4 Ghosting artifacts removal through boundaries detection

In natural images, pixels along object boundaries have mixed foreground/background color. After the projection of the reference image into a new viewpoint, these mixed pixels can become far from the new object boundary, resulting in an annoying ghosting artifact.

Ghosting artifacts are often avoided by detecting depth discontinuities on the depth map, in order to separate the boundary layer (containing pixels near a boundary) from the main layer (containing pixels far from a boundary) [Zitnick et al., 2004]. The main layer is first projected into the virtual viewpoint, then the boundary layer is added everywhere it is visible (i.e. where its depth value is smaller than the main layer's one). In [Müller et al., 2008b], the authors propose to split again the boundary layer into foreground boundary and background boundary layers. The main layer is first projected, the foreground boundary layer is then added everywhere it is visible, and the background boundary layer is finally used to fill in remaining holes.

Ghosting artifacts can be further avoided by estimating the background and foreground contributions in the rendered view with the help of advanced matting techniques [Chuang et al., 2002, Hasinoff et al., 2006, Wang & Cohen, 2007, Sarim et al., 2009].

3.5 Disocclusion filling

One issue that occurs when rendering virtual views is called disocclusion. It is based on the fact that one single video stream does not contain the color values of all pixels of the 3D scene.

When using DIBR for 3D video, the virtual views are usually shifted horizontally with respect to the original view. This allows the virtual view to "look behind" some foreground objects and to see background that is occluded in the original view. The original video stream does not provide any information about the color values of those background areas, so they come as holes in the rendered view. The location and size of holes may differ from frame to frame.

There are two classical techniques to fill these disoccluded regions: multi-view blending methods and image inpainting methods. Note that specific representations such as Layered Depth Videos (LDV) can also be helpful for addressing the problem of occlusion handling since they allow storing texture information seen by other cameras [Shade et al., 1998, Yoon et al., 2007, Müller et al., 2008a].

The multi-view blending scheme consists in synthesizing the desired virtual view from each input view independently, then to blend the resulting synthesized views. Disocclusions are thus filled in with information captured by some extra views, when they are available. This technique is used to perform intermediate view synthesis, as proposed in Section 4.3.

When extra views are not available, the frequent solution for disocclusion handling is image interpolation with inpainting techniques. Classical inpainting algorithms are detailed in the following section, and an advanced depth-aided inpainting solution is proposed in Section 4.2.

3.5.1 Image inpainting

Image inpainting refers to methods which consist in filling-in missing regions (holes) in an image [Bertalmío et al., 2000]. Inpainting techniques find applications in a number of other image processing problems: image editing (e.g. object removal), image restoration, image coding, loss concealment after impaired transmission...

A good review on the use of inpainting for image-based rendering can be found in [Tauber et al., 2007]. Existing methods can be classified into two main categories: diffusion-based approaches and exemplar-based approaches.

The first category concerns diffusion-based approaches which propagate level lines or linear structures (so-calles isophotes) via diffusion based on partial differential equations [Bertalmío et al., 2001, Telea, 2004, Tschumperlé, 2006]. These methods tend to extend isophotes intersecting the border of the region to be filled. The drawback of diffusion based methods is thus to introduce some blur when the hole to be filled is large.

The second category concerns exemplar-based approaches which sample and copy best match texture patches from the known image neighborhood [Harrison, 2001, Bornard et al., 2002, Criminisi et al., 2003, Drori et al., 2003, Jia & Tang, 2003, Sun et al., 2005, Barnes et al., 2009, Barnes et al., 2010]. These methods have been inspired by texture synthesis techniques [Efros & Leung, 1999] and are known to work well in case of repeating textures. The first attempt to use exemplar-based techniques for hole filling has been reported in [Harrison, 2001]. The authors in [Drori et al., 2003] improve the search for similar patches by introducing a priori rough estimate of the inpainted values using a multi-scale approach which then results in an iterative approximation of the missing regions from coarse to fine level. In addition, the candidate patches for the match also include rotated, scaled, and mirrored versions of texture patches taken from the image.

The two types of methods (diffusion and exemplar-based) can be efficiently combined so that Partial Differential Equations (PDE) diffusion techniques are applied on patches with structured features and exemplar-based methods are used for synthesizing patches containing texture with finer details. In [Criminisi et al., 2003], the authors introduce an exemplar-based method in which the filling order is defined by a priority function which depends on the angle between the isophote direction and the normal direction of the local filling front. The goal of this priority function is to ensure that linear structures (ie structural patches) are propagated before texture filling (ie texture patches).

Another approach based on the original method of Efros and Leung [Efros & Leung, 1999] has been introduced in [Ashikhmin, 2001] which searches among candidate patches which have already been inpainted. A sparse approximation method is considered in [Xu & Sun, 2010] together with a sparsity-based priority function. The results obtained with this method show sharp inpainting results with efficient and consistent inferring of structures and textures.

The next section presents in more details the exemplar-based inpainting approach, introduced in [Criminisi et al., 2003].

Exemplar-based inpainting approach

The inpainting algorithm introduced in [Criminisi et al., 2003] is an exemplar-based inpainting technique. The authors noted that exemplar-based inpainting techniques can replicate both texture and structure, and they demonstrate that the quality of the output image synthesis is highly influenced by the order in which inpainting is processed. Based on these observations, they describe an inpainting algorithm which iterates the following two steps. First, a priority term is computed, in order to determine the next patch of the image to be filled in. Second, this selected patch is filled in by copying a patch chosen as the best match for the known pixels of the patch to be filled in. These two steps are iterated until all missing pixels have been filled in.

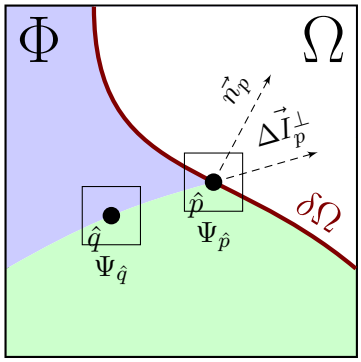


Figure 3.6: Notation diagram, introduced in [Criminisi et al., 2003]. Given the block Ψ_p, n_p is the normal to the contour $\delta\Omega$ separating the hole region Ω from the non-hole region Φ. $\Delta I_p^{\perp}$ is the isophote at point p.

Considering an input image I, and a missing region Ω, the source region Φ is defined as $\Phi = I - \Omega$ (see Figure 3.6). For each pixel p along the frontier $\delta\Omega$, let be the patch Ψ_p centered in point p.

The priority $P(p)$ of the patch Ψ_p is computed as the product of the Confidence term $C(p)$

and the Data term $D(p)$.

$$P(p) = C(p) \cdot D(p) \tag{3.14}$$

The Confidence term $C(p)$ indicates the reliability of the current patch. It is related to the ratio between the number of known pixels in the patch, compared to the size of the patch. $C(p)$ is initialized to 0 if $p \in \Omega$ or 1 if $p \in \Phi$, and then it is recomputed for pixels $p \in \delta\Omega$ as follows:

$$C(p) = \frac{1}{|\Psi_p|} \sum_{q \in \Psi_p \cap \Phi} C(q) \tag{3.15}$$

where $|\Psi_p|$ is the number of pixels within the patch Ψ_p. This Confidence term increases the priority of pixels whose neighborhood is already known.

The Data term $D(p)$ is computed as the scalar product of the isophote direction $\Delta I_p^{\perp}$, and the unit vector n_p, orthogonal to $\delta\Omega$ at point p.

$$D(p) = \frac{\left\langle \Delta I_p^{\perp}, n_p \right\rangle}{\alpha} \tag{3.16}$$

where α is a normalization factor (e.g. $\alpha = 255$ for a typical gray-level image). This Data term increases the priority of pixels on linear structures, in order to extend them.

Once all priorities on $\delta\Omega$ are computed, the patch $\Psi_{\hat{p}}$ with the highest priority is selected to be filled in. Then, a template matching algorithm searches for the best exemplar $\Psi_{\hat{q}}$ to fill in missing pixels under $\Psi_{\hat{p}}$, as follows:

$$\Psi_{\hat{q}} = \arg\min_{\Psi_q \in \Phi} \{\mathrm{SSD}_{\Phi}(\Psi_{\hat{p}}, \Psi_q)\} \tag{3.17}$$

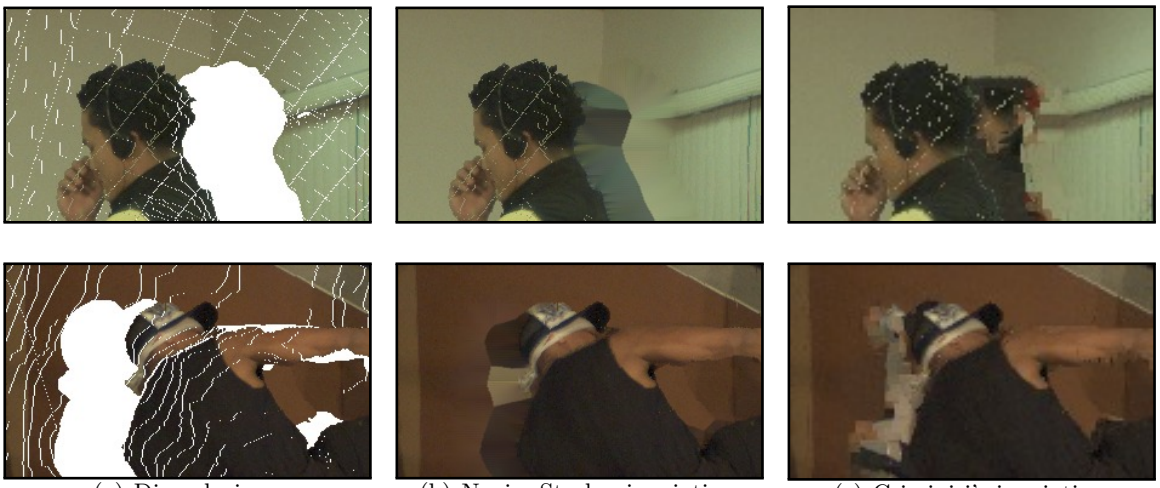
where $\mathrm{SSD}_{\Phi}(\cdot, \cdot)$ is the distance between two patches, defined as the Sum of Squared Differences and only computed on pixels from the non-hole region Φ.

The priority computation step and the best match duplication step are iterated until all missing pixels have been filled in. The exemplar-based inpainting method produces good results for object removal and image restoration, but is not well suited to address the problem of disocclusions handling, because disocclusions should be filled in with background texture only.

Most inpainting techniques use neighboring pixels solely based upon colorimetric distance, while a disocclusion hole should be filled in with background pixels, rather than foreground ones. Figure 3.7 presents the results of two classical inpainting methods. One can observe that the disocclusions are filled in with both foreground and background texture, resulting in an annoying artifact. Inpainting techniques specially designed for disocclusions filling are described in next sections.

Inpainting methods for disocclusions filling

Some classical inpainting methods have been adapted to specifically handle disocclusions in a DIBR algorithm. In [Do et al., 2009], the authors estimate each pixel value inside a disocclusion area from nearest known pixels along the eight cardinal directions, after nullifying the weight of foreground pixels. In [Oh et al., 2009], the authors temporarily replace foreground textures by background texture before inpainting, so that disocclusions are filled in only with background



(a) Disocclusions (b) Navier-Strokes inpainting (c) Criminisi's inpainting

Figure 3.7: Disocclusions filling with inpainting method. The disocclusions 3.7(a) are filled in with either a diffusion-based inpainting approach based on Navier-Strokes equations [Bertalmío et al., 2000] 3.7(b), or an PDE-based inpainting approach [Criminisi et al., 2003] 3.7(c)

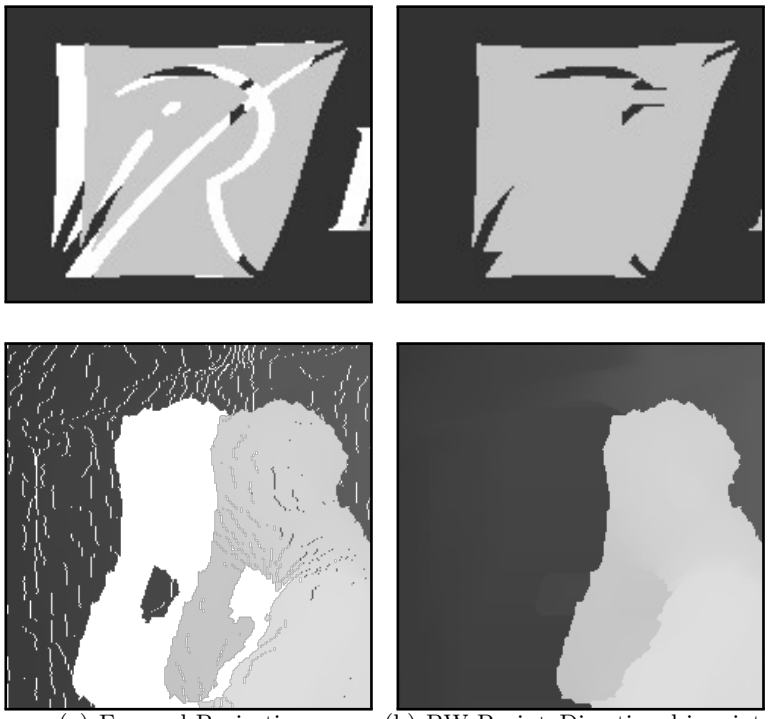
texture.

Advanced depth-aided inpainting methods assume that the depth map of the virtual viewpoint to be rendered is available. In [Daribo & Pesquet, 2010], the authors enhance the inpainting method in [Criminisi et al., 2003] by reducing the priority of patches containing a depth discontinuity, and by adding a depth comparison in the search for best matches. In [Gautier et al., 2011], the authors use a similar approach but estimate isophotes directions with a more robust tensor computation and constrain the propagation in the direction of the epipole.

The full depth map for the virtual view is not available most of the time, and it must be estimated from the input depth map. In [Daribo & Pesquet, 2010], the authors perform a diffusion-based inpainting [Bertalmío et al., 2001] on the projected depth map, but both foreground and background are diffused to fill disocclusions. In [Nguyen et al., 2009], the authors constrain the depth map inpainting in the direction of the epipole, in order that only the background is diffused, but this method fails when a disocclusion is surrounded by foreground depth.

Figure 3.8 shows the result of such an inpainting method. In Figure 3.8(a), the depth map is synthesized by the point-based forward projection method, which generates cracks and disocclusions. In Figure 3.8(b), the depth map is synthesized by the advanced projection method proposed in [Nguyen et al., 2009]. This method is composed of three steps: a point-based forward projection; a median filtering of the synthesized depth map (as presented in Section 3.3); and a directional inpainting method to fill in disocclusions. Unfortunatly, the directional inpainting, processed after the projection, fills some holes with foreground depth value, because the connectivity is not preserved.

Connectivity information is lost during the projection step. Without this connectivity information, every inpainting method fails to fill in background disocclusions if the disoccluded area is surrounded by foreground objects. This case may happen each time a foreground object



(a) Forward Projection. (b) BW Proj + Directional inpaint.

Figure 3.8: Results of virtual depth map synthesis. The point-based projection method generates cracks and disocclusions 3.8(a). Median filtering and directional inpainting [Nguyen et al., 2009] fill some holes with foreground depth 3.8(b).

is not convex, and contains holes. As a result, the structure of the foreground object is modified, and some parts of the background are filled in with foreground values.

The next section presents in more details the extension to the Criminisi's inpainting methods, proposed in [Daribo & Pesquet, 2010], which takes into account the depth map during the inpainting process.

Depth-aided inpainting method

The Criminisi's inpainting method has been adapted in [Daribo & Pesquet, 2010], to address the specific problem of disocclusion handling. The authors propose two major modifications, which are the addition of a new term $L(p)$ into the priority function evaluation, and the consideration of depth into the SSD computation.

The first modification consists in introducing the Level regularity term $L(p)$ as a third term into the priority function $P(p)$.

$$P(p) = C(p) \cdot D(p) \cdot L(p) \tag{3.18}$$

The Level regularity term $L(p)$ is defined as the inverse variance of the depth patch Z_p:

$$L(p) = \frac{|Z_p|}{|Z_p| + \sum_{q \in Z_p \cap \Phi} \left(Z_p(q) - \bar{Z}_p \right)^2} \tag{3.19}$$

where $|Z_p|$ is the area (in terms of number of pixels) of depth patch Z_p centered in p, $Z_p(q)$ is

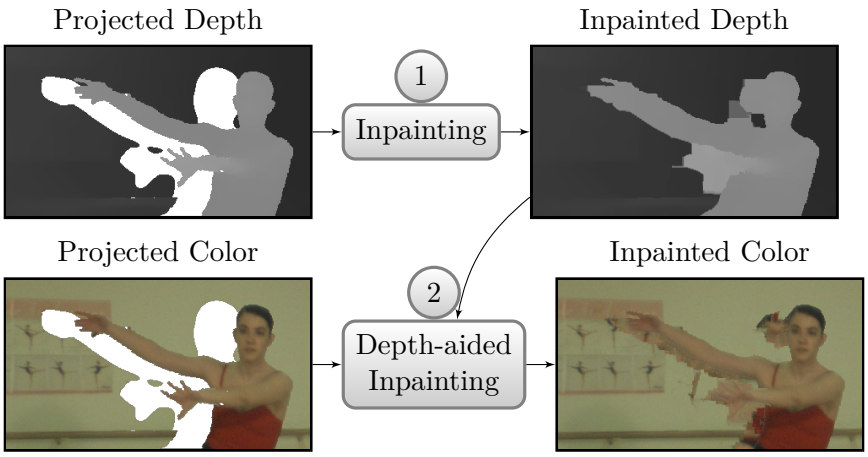


Figure 3.9: Depth-aided inpainting scheme [Daribo & Pesquet, 2010]. Firstly, the depth map is inpainted with a Navier-Stokes based algorithm [Bertalmío et al., 2001]. Secondly, the texture is inpainted, aided with the resulting depth map.

the depth value at pixel location q under Z_p, and $\bar{Z}_p$ the mean value.

The second modification consists in adding depth into SSD computation, computed during the best match search. Equation 3.17 is thus modified as follows:

$$\Psi_{\hat{q}} = \arg\min_{\Psi_q \in \Phi} \{\mathrm{SSD}_\Phi(\Psi_{\hat{p}}, \Psi_q) + \alpha\, \mathrm{SSD}_\Phi(Z_{\hat{p}}, Z_q)\} \tag{3.20}$$

where the parameter α controls the importance given to the depth distance minimization.

The Level regularity term $L(p)$ gives more priority to patches with a constant depth level, which is expected to favor background pixels over foreground ones. Considering depth into the SSD computation favors patches which are at the same depth as the patch to be copied.

3.6 Conclusion

The classical DIBR scheme for virtual view extrapolation from single input view plus depth video sequence is the merging of all these described methods. The complete process, shown in Figure 3.10, is divided in several distinct steps, each one designed to solve a specific problem. First, the input depth map is warped onto the virtual viewpoint. The obtained warped depth map contains disocclusions, cracks and ghosting artifacts. Second, this virtual depth map is filtered a first time with a median filter, in order to remove the cracks, then a second time to dilate disocclusion areas on the background side, in order to avoid ghosting artifacts during view synthesis. Third, the filtered depth map is involved in a backward warping step to compute the color of each pixel of the virtual view being outside of disocclusion areas. Fourth, this resulting depth map is inpainted, to fill in disocclusion areas. Finally, this complete depth map is used by a depth-aided inpainting algorithm to fill in disocclusions in the color map.

All these steps are inter-dependent, and errors introduced by each one are amplified by the following one. Connectivity information is lost during the first projection step, as explained in Section 3.5.1. Without this connectivity information, every inpainting method fails to fill in background disocclusions if the disoccluded area is surrounded by foreground objects. As a

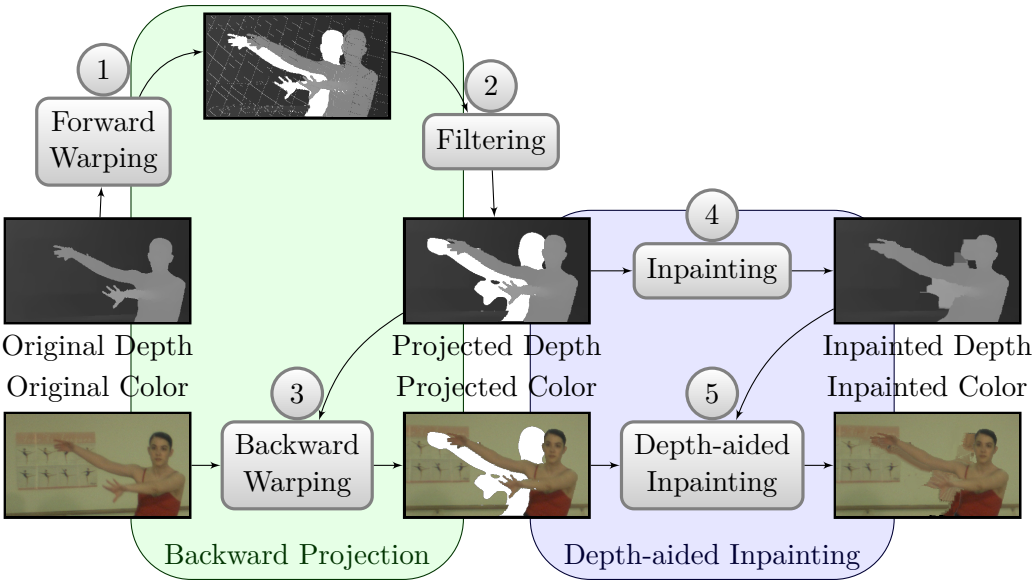


Figure 3.10: Classical scheme for virtual view extrapolation from a single input view plus depth video sequence. First, the input depth map is projected onto the virtual viewpoint. Second, the resulting depth map is filtered to avoid cracks and ghosting artifacts. Third, the filtered depth map is projected back onto the reference viewpoint to find the color of each pixel. Fourth, the depth map is inpainted to fill in disocclusions. Finally, the inpainted depth map is used to conduct disocclusions filling of the color map (synthesized view).

result, depth-aided inpainting uses wrong foreground patches to fill in background disocclusions, producing annoying artifacts.

As a conclusion, state-of-the-art DIBR methods need a complete depth map at the rendered viewpoint (for backward projection and depth-aided inpainting). However, no fully satisfying method yet exists to obtain a complete and correct depth map, which would avoid artifacts generation when used for DIBR. Moreover, most disocclusion handling methods proposed in the literature work as a post processing on the projected view. Connectivity information is not preserved during the projection, and inpainting methods fail to fill in background disocclusions when they are surrounded by foreground objects. Next chapter introduces an advanced DIBR method which aims at suppressing such drawbacks.

Chapter 4

Joint Projection Filling method for new DIBR

As explained in the previous chapter, state-of-the-art DIBR methods need a complete depth map at the rendered viewpoint, but do not provide satisfying solution to compute these depth map. This chapter introduces thus two new DIBR techniques, based on a novel forward projection algorithm, called the Joint Projection Filling method (JPF). The JPF method performs forward projection, using connectivity information to fill in disocclusions in a single step.

The first proposed DIBR method is designed to extrapolate virtual views from a single input view plus depth video sequence. The method differs from the classical scheme, presented in the previous section, by two points: the virtual depth map is synthesized by the JPF method, avoiding the use of dedicated filtering and inpainting processes; the depth-aided texture inpainting method is revised to take into account the high quality of the synthesized depth map.

The second proposed DIBR method is designed to interpolate intermediate views from multiple input view plus depth sequences. The method uses the Floating Texture approach to register multiple input view plus depth sequences before blending.

This chapter is organized in three sections.

Section 4.1 introduces the JPF method as a novel forward projection algorithm. The JPF method simultaneously handles warping and disocclusion filling, in order to preserve connectivity and fill in disocclusions with background textures. The JPF method is designed to synthesize virtual views from one or many input view plus depth video sequences, depending on the final application.

Section 4.2 describes a DIBR method to synthesize a virtual view from a single color plus depth video. This *extrapolation* algorithm uses the JPF method to synthesize the virtual depth map, as well as advanced depth-aided inpainting method to synthesize the virtual color map.

Section 4.3 presents a DIBR method to synthesize an intermediate view from a pair of color plus depth videos. This *interpolation* algorithm uses the JPF method to rapidly synthesize the desired view from each input view, and then uses an advanced texture blending algorithm to obtain the final view.

4.1 Joint Projection Filling method for disocclusion handling

This section introduces the Joint Projection Filling (JPF) method, which uses an occlusion compatible ordering [McMillan, 1995] for detecting cracks and disocclusions, for which unknown depth values are estimated while performing warping.

The JPF method is a forward projection algorithm, similar to the method described in Section 3.2. The algorithm uses the warping equations to synthesize a virtual view from an input view plus depth video.

During warping, overlapping (several pixels projected at the same position) or disocclusion (no pixels projected at a position) may happen. In [McMillan, 1995], a pixel scanning order is introduced to perform the painter's algorithm during the projection. In case of overlapping, this pixel scanning order ensures the pixel just projected at a position to be the foreground pixel so that the z-buffer is not needed. A second property, resulting from the first one, is more helpful to handle disocclusions. If the pixel just warped in the virtual image is far from the previous one, the hole between them is a disocclusion, and the pixel just warped is the background pixel. This second property is exploited to ensure that only background pixels are used to fill in disocclusion areas.

Section 4.1.1 describes the JPF algorithm. It is first presented for rectified cameras and then generalized for non-rectified cameras.

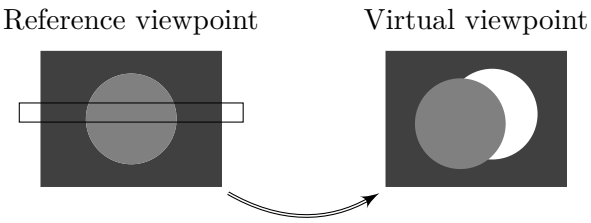
Section 4.1.2 introduces a pixel's confidence measure into the JPF method to remove ghosting artifacts.

Section 4.1.3 presents some results of synthesized depth maps, obtained by the JPF method.

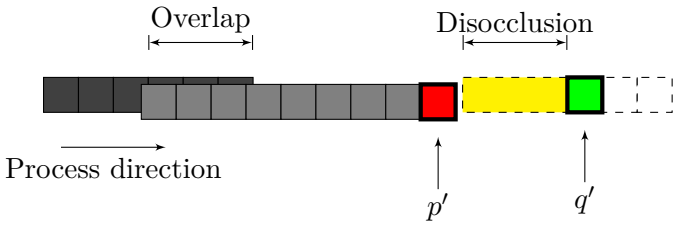
4.1.1 Disocclusion detection

The JPF method fills in disocclusion areas during the projection, to ensure that geometrical structures are well preserved. The method uses the occlusion-compatible ordering presented by McMillan in [McMillan, 1995], which uses epipolar geometry to select a pixel scanning order. The algorithm was initially introduced to perform the painter's algorithm during the projection without the need of a Z-buffer. Here, not using a Z-buffer is not our purpose (by the way, the constructed depth map can be regarded as a Z-buffer). The occlusion-compatible ordering is instead used to handle disocclusions gracefully. Cracks are filled in by interpolation of neighboring pixels, whereas disocclusions are only filled in by background pixels. This technique can be used with non-rectified views, avoiding prior creation of parallax maps as done in [Kauff et al., 2007].

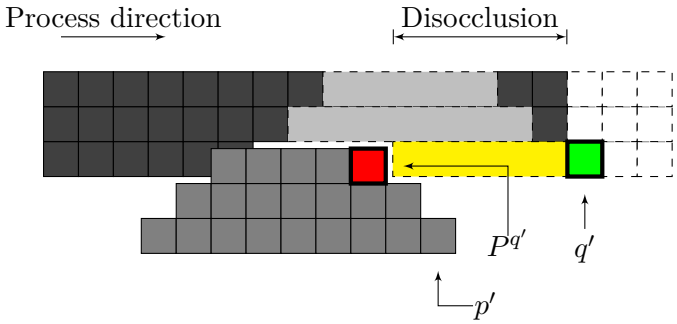
The scanning order introduced by McMillan in [McMillan, 1995] is based on the epipolar geometry. This ordering is determined by examining a single point, the position of the projection of the virtual viewpoint. The 3D camera center of the virtual viewpoint is first projected into the input image coordinated system. This point, named the epipolar center, defines a partition of the original image pixels into four groups (potentially empty): pixels above left, above right, below left and below right of the epipolar center. A different scanning order in defined for any of these groups. For sake of explanations, we assume in the following that pixels from the reference image are processed sequentially, from top-left to bottom-right, according to McMillan scanning order. Otherwise, the inequalities should be inverted.



(a) Projection.



(b) Details of a single row of pixels in the synthesized view for a rectified projection.



(c) Details of some rows of pixels in the synthesized view for a non rectified projection.

Figure 4.1: JPF method scheme for rectified 4.1(b) and non rectified 4.1(c) cameras. q' is a background pixel which is used to fill in the highlighted disocclusion.

Figure 4.1(b) presents the principle of the Joint Projection Filling (JPF) method, in the particular case of rectified cameras. Each row is thus independent of the others, reducing the problem to one dimension. Consider a row of pixels from the reference view, and a pixel $p = (p_x, p_y)$ on that row. The pixel p is projected on position p' in the synthesized view. After having processed pixel p, the next pixel to be processed is $q = (p_x + 1, p_y)$. Its projected position $q' = (q'_x, p_y)$ verifies one out of the three following equations:

$$\begin{cases} q'_x = p'_x + 1 & \text{Pixels } p' \text{ and } q' \text{ are adjacent.} \\ q'_x < p'_x + 1 & \text{There is an overlap.} \\ q'_x > p'_x + 1 & \text{There is a crack or a disocclusion.} \end{cases} \tag{4.1}$$

The first and the second cases do not generate artifacts. In the last case, p' and q' are in same order as p and q, but there is a gap between them. In the proposed method, contrary to classical point-based projection, this gap is filled in immediately, before projecting of rest of the image. The method to fill the gap is adapted to its size. If the gap is small enough, it is considered as a

crack. p' and q' are thus assumed to be on same layer, and the gap is filled in by interpolating the two pixels p' and q'. If the gap is too large, it is considered as a disocclusion. p' and q' are thus assumed to be on two distinct depth layers and the gap is filled in only by background pixel. The McMillan pixel ordering ensures that q' is the background pixel, which is stretched from position p' to q'. The value of each pixel m between p' and q' is thus estimated as follows:

$$m = \begin{cases} (1-\alpha)p' + \alpha q' & \text{if } d \leq K \\ q' & \text{if } d > K \end{cases} \qquad \text{where} \begin{cases} d = q'_x - p'_x \\ \alpha = \frac{1}{d}(m_x - p'_x) \end{cases} \tag{4.2}$$

In the simulation results reported in Section 4.1.3, the threshold K has been fixed to 5 pixels, to handle cracks and small disocclusions.

The algorithm is generalized for non-rectified cameras, as illustrated in Figure 4.1(c). Pixels p' and q' may no longer be on the same row, thus we define the pixel $P^{q'}$ as the last pixel projected on row q'_y. Equation (4.1) is revised, replacing p' with $P^{q'}$, thus q' and $P^{q'}$ are on the same row.

$$\begin{cases} q'_x \leq P^{q'}_x + 1 & \text{There is no artifact.} \\ q'_x > P^{q'}_x + 1 & \text{There is a disocclusion.} \end{cases} \tag{4.3}$$

As previously, the disocclusion handling method depends on the distance between q'_x and $P^{q'}_x$. The value of each pixel m between $P^{q'}$ and q' is thus estimated as follows:

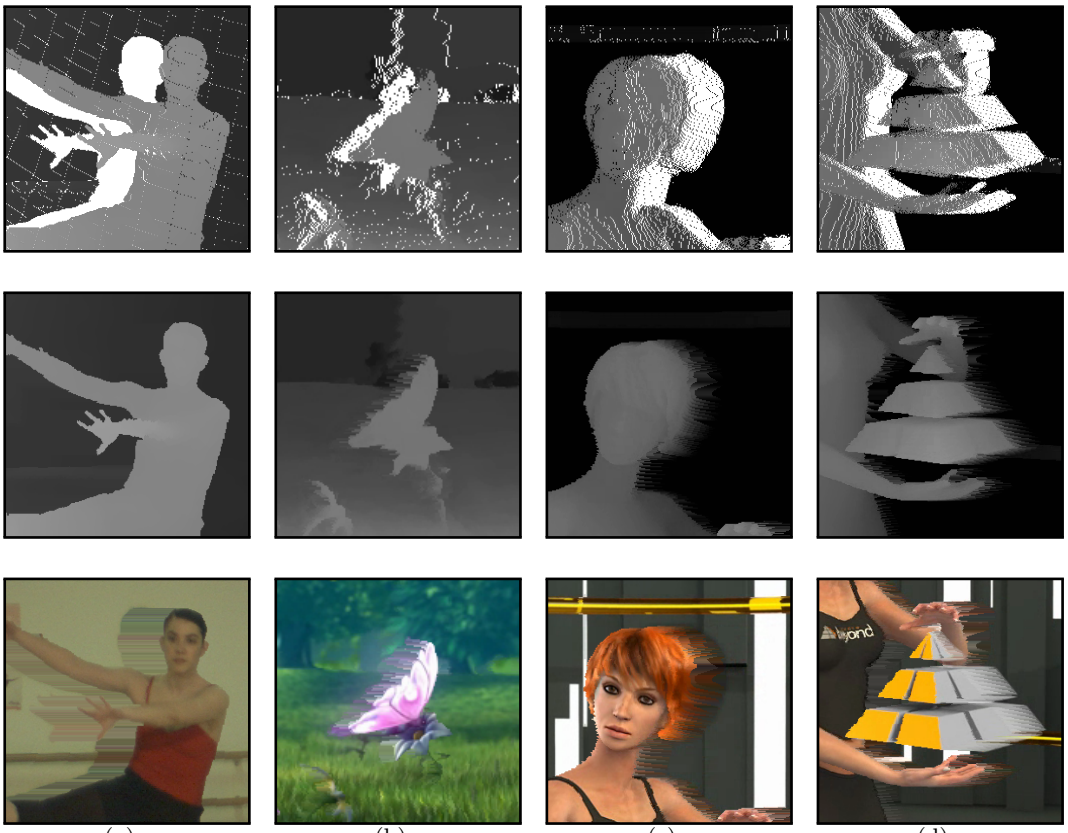
$$m = \begin{cases} (1-\alpha)P^{q'} + \alpha q' & \text{if } d \leq K \\ q' & \text{if } d > K \end{cases} \qquad \text{where} \begin{cases} d = q'_x - P^{q'}_x \\ \alpha = \frac{1}{d}(m_x - P^{q'}_x) \end{cases} \tag{4.4}$$

Figure 4.2 presents the synthesized depth maps obtained with the JPF method, without any ghosting removal technique. Our JPF method has removed all cracks and has filled in the disocclusions with background pixels. Depth maps from the "Ballet" sequence contain sharp discontinuities, which are preserved by the JPF method (Figure 4.2(a)). Depth maps from other sequences (Figure 4.2(b) 4.2(c) 4.2(d)) contain some blur along depth discontinuities, due to DCT-based compression. This blur produces some sparse pixels inside the disocclusion area, which are stretched to fill the disocclusion, resulting in an annoying artifact. The next section explains how to recover edges sharpness and reduce stretching artifacts.

4.1.2 Disocclusion filling

Pixels along objects boundaries are considered unreliable, because they often contain mixed foreground/background information for texture and depth value. Their projection may thus create artifacts in the synthesized views.

The JPF method, as described in Section 4.1.1, fills in each row of a disoccluded region using a single pixel. When applied on such a "blended" boundary pixel, this method may result in annoying pixel stretching artifacts, as can be seen in the third row of Figure 4.2. However, these artifacts can be minimized by adapting the pixel stretching length, according to a pixel confidence measure. The algorithm used to avoid stretching artifacts thus proceeds with the following two steps:



(a) (b) (c) (d)

Figure 4.2: Comparison between synthesized depth maps from a forward point-based projection (first row) and from the JPF method (second row). Blurred depth discontinuities in the original depth map produces stretching effects on the synthesized depth maps, particularly annoying on the synthesized color map (third row). Note that McMillan scanning order is from right to left in figures 4.2(a) and 4.2(b), whereas it is from left to right in figures 4.2(c) and 4.2(d), due to epipole location for input image pairs.

In a first step, a confidence measure $\lambda_q \in [0;1]$ is computed for each pixel q by convolving the depth map (Z) with a Difference-Of-Gaussians (DOG) operator as follows:

$$\lambda_q = 1 - (\mathrm{DOG} * \mathrm{Z})(q) \tag{4.5}$$

The DOG operator is built as the difference of two gaussians: the gaussian G of variance σ^2, and the 2D Dirac delta function δ_2.

$$\begin{aligned} \mathrm{DOG} &= G - \delta_2 \\ G(u,v) &= \frac{1}{\sigma^2} \cdot \phi\left(\frac{u}{\sigma}\right) \cdot \phi\left(\frac{v}{\sigma}\right) \end{aligned} \tag{4.6}$$

where ϕ is the standard normal distribution. The value of σ, in the experiments described below, has been fixed to 3. This function is chosen to compute a high confidence score for foreground pixels, and background pixels far from a depth discontinuities. The computed confidence score for background pixels decreases accordingly to their proximity to a depth discontinuity.

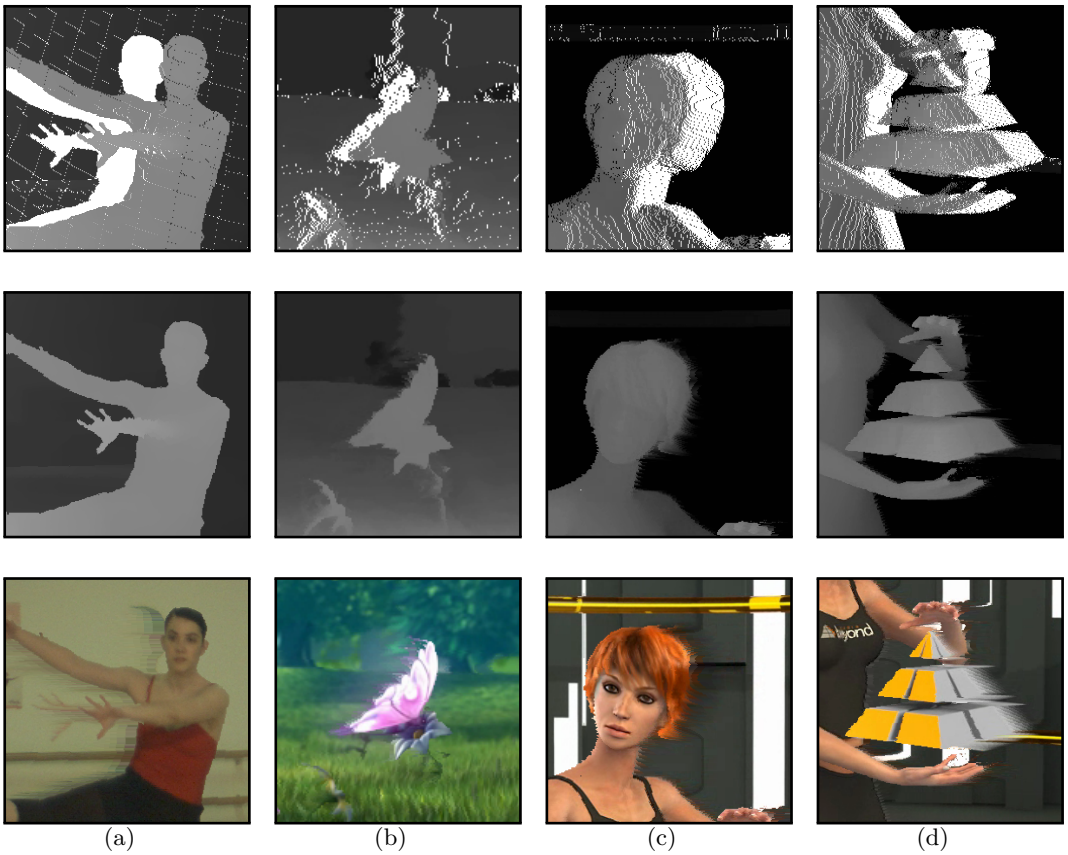
In a second step, the confidence measure is used during the JPF method, to confine pixel

stretching. Reusing the notations introduced in Section 4.1.1, suppose that a wide disocclusion is discovered during the projection of pixel q. Instead of filling the whole gap between $P^{q'}$ and q', with color and depth values of q', only a part of the gap is filled in. The rest will be filled with the next pixel which will be projected on that same row j.

Assume M, the point between $P^{q'}$ and q' defined with the following equation:

$$M = (1 - \lambda_q)P^{q'} + \lambda_q q' \tag{4.7}$$

The gap between $P^{q'}$ and M is filled in by pixel q', thus pixels on foreground/background boundaries which have low confidence measures are used to fill the disocclusion only for a couple of pixels next to the foreground, where blended pixels are expected to be in the synthesized view.

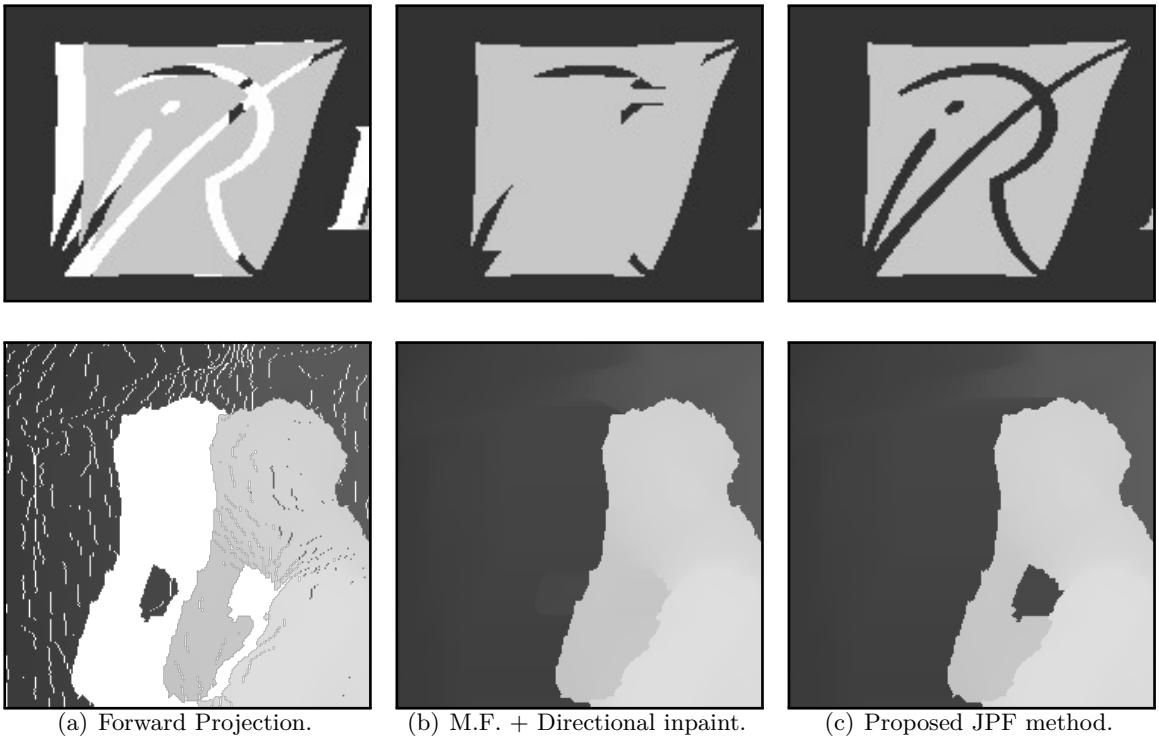


(a) (b) (c) (d)

Figure 4.3: Comparison between synthesized depth maps from a forward point-based projection (first row) and from the JPF method with confidence-based interpolation (second row). Note that McMillan scanning order is from right to left in figures 4.3(a) and 4.3(b), whereas it is from left to right in figures 4.3(c) and 4.3(d), due to epipole location for input image pairs.

The confidence-based interpolation method shifts back unreliable pixels near the discontinuities and uses reliable pixels to fill in disocclusions. Figure 4.3 presents the rendering results of the JPF method with confidence-based interpolation. These results are to be compared with those presented in Figure 4.2. One can see that depth discontinuities are sharpened, producing more realistic depth maps. The third row presents the results obtained with the same algorithm applied on texture. Disocclusions are gracefully filled in when the background is uniform, but stretching artifacts still remain in case of textured background.

4.1.3 Results



(a) Forward Projection. (b) M.F. + Directional inpaint. (c) Proposed JPF method.

Figure 4.4: Virtual depth map synthesized by three forward projection methods. The point-based projection method generates cracks and disocclusions 4.4(a). Median filtering and directional inpainting [Nguyen et al., 2009] fills some holes with foreground depth 4.4(b). The proposed JPF method fills cracks and disocclusions with realistic background 4.4(c).

Figure 4.4 shows virtual depth maps synthesized by the JPF method, and compares them to virtual depth maps synthesized by state-of-the-art forward projection. On can observe that geometrical structures are well preserved by our JPF projection, even for complex scene geometry.

The JPF method simultaneously handles warping and disocclusion filling, in order to preserve connectivity and fill in disocclusions with background texture and depth. Unfortunately, the method could introduce texture stretching. It is thus more suitable to use it as a step of a virtual view synthesis algorithm. Two use cases are addressed in next sections, either for virtual view extrapolation when only one input view is available, or for intermediate view interpolation when multiple input views are available.

4.2 View extrapolation with full-Z depth-aided inpainting

In order to synthesize a virtual view from only one input view plus depth sequence, the classical rendering scheme, introduced in Figure 3.10, is replaced by the one presented in Figure 4.5. First, the depth map for the virtual view is synthesized by our JPF method, handling ghosting, cracks and disocclusions. Then, the texture of the virtual view is obtained by a classical backward warping followed by an enhanced full-Z depth-aided inpainting algorithm.

Our proposed full-Z depth-aided inpainting algorithm is a modification of the depth-aided inpainting method described in [Daribo & Pesquet, 2010], itself based on the exemplar-based

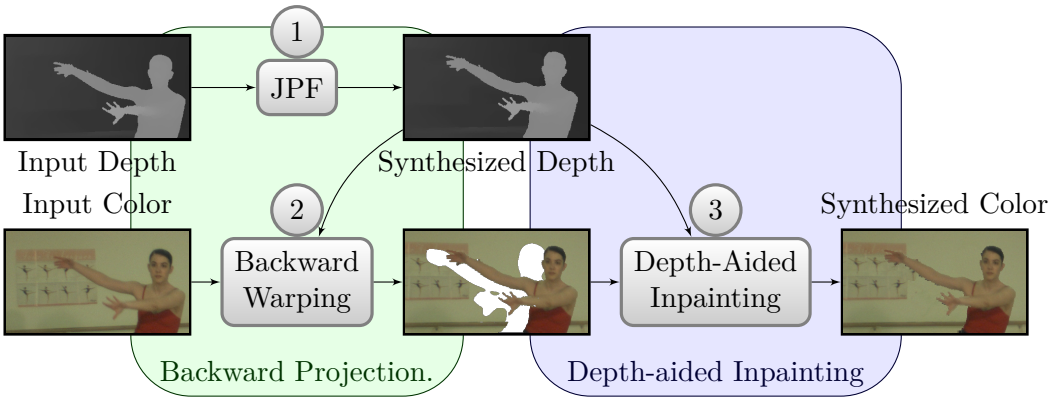


Figure 4.5: Proposed scheme for virtual view extrapolation from single input view plus depth sequence. First, the Joint Projection Filling (JPF) method handles cracks and disocclusions during the depth map warping. Then, the backward projection method synthesizes the virtual view. Finally, the depth-aided inpainting takes into account the high quality of the computed depth map to fill disoccluded areas in the synthesized view.

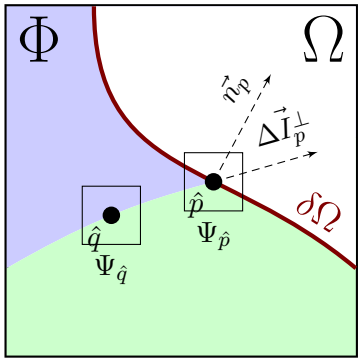
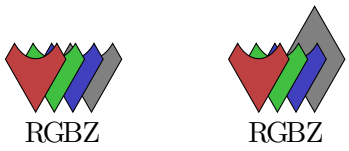


Figure 4.6: Notation diagram, introduced in [Criminisi et al., 2003]. Given the block Ψ_p, n_p is the normal to the contour $\delta\Omega$ separating the hole region Ω from the non-hole region Φ. $\Delta I_p^{\perp}$ is the isophote at point p.

inpainting approach, introduced in [Criminisi et al., 2003]. The exemplar-based inpainting approach and the depth-aided overlay are both detailed in Section 3.5.1. They use the notations reintroduced in Figure 4.6. The depth-aided overlay, proposed by Daribo et al., uses the depth map to drive the inpainting process. This section describes our proposed modification which takes into account the high quality of the virtual depth map. The importance of the synthesized depth map quality is discussed in Section 4.2.2, for three different depth-aided inpainting methods.

4.2.1 Full-Z depth-aided inpainting method

The synthesized depth map does not contain holes, thanks to the JPF method which projects the input depth map onto the virtual viewpoint while filling cracks and disocclusions. The patch $\Psi\hat{p}$ to be filled in thus contains a depth value for each pixel, even for pixels in the hole region Ω. These depth values are close to the ground truth, because disocclusions are only filled in with background depth. The proposed modification to the depth-aided inpainting is to use the depth value of all pixels in the patch, including those whose color is not known. Equation 3.20 is thus



(a) (b)

Figure 4.7: Part of the patch $\Psi_{\hat{p}}$ which is involved in SSD computation. In [Daribo & Pesquet, 2010], the authors compute SSD with color (RGB) and depth (Z) information only from the known part of the patch $\Psi_{\hat{p}} \cap \phi$, as shown in Figure 4.7(a). Instead, we use the depth of the full patch to compute SSD, as shown in Figure 4.7(b).

modified as follows:

$$\Psi_{\hat{q}} = \arg\min_{\Psi_q \in \Phi} \{\mathrm{SSD}_{\Phi}(\Psi_{\hat{p}}, \Psi_q) + \alpha\, \mathrm{SSD}_{\underline{\Phi \cup \Omega}}(Z_{\hat{p}}, Z_q)\} \tag{4.8}$$

Figure 4.7 shows the part of patch $\Psi\hat{p}$ which is involved in SSD computation.

Results of the proposed full-Z depth-aided inpainting method are analyzed in the next section, and compared with results from two other depth-aided inpainting methods.

4.2.2 Inpainting Results

This section compares virtual view synthesis results obtained when using three depth-aided inpainting techniques for occlusion handling.

- Results of the first method, denoted Daribo DAI [Daribo & Pesquet, 2010], are presented in Figure 4.8.
- Results of the second method, denoted Gautier DAI [Gautier et al., 2011], are presented in Figure 4.9.
- Results of the proposed method, denoted Full-Z DAI, are presented in Figure 4.10.

For each inpainting technique, virtual depth map is synthesized either by the classical scheme shown in Section 3.6, or by the JPF method. Each figure contains five columns:

- The first column shows a virtual view synthesized by the backward projection. This column is helpful to identify disocclusion areas marked in white.
- The second column shows the virtual depth map where disocclusions are filled in with a Navier-strokes inpainting algorithm [Bertalmío et al., 2001].
- The third column show the results of the evaluated depth-aided inpainting method, led by the depth map presented in column 2.
- The fourth column shows the virtual depth map synthesized with our JPF method.
- The fifth column show the results of the evaluated depth-aided inpainting method, led by the depth map presented in column 4.

One can observe that the depth maps shown in column 2 are not realistic because depth discontinuities do not fit with object boundaries. This is due to the depth map inpainting method, which fills disocclusions with both background and foreground values. On the contrary, depth maps presented in column 4 are closer to the ground truth, thanks to the JPF method.

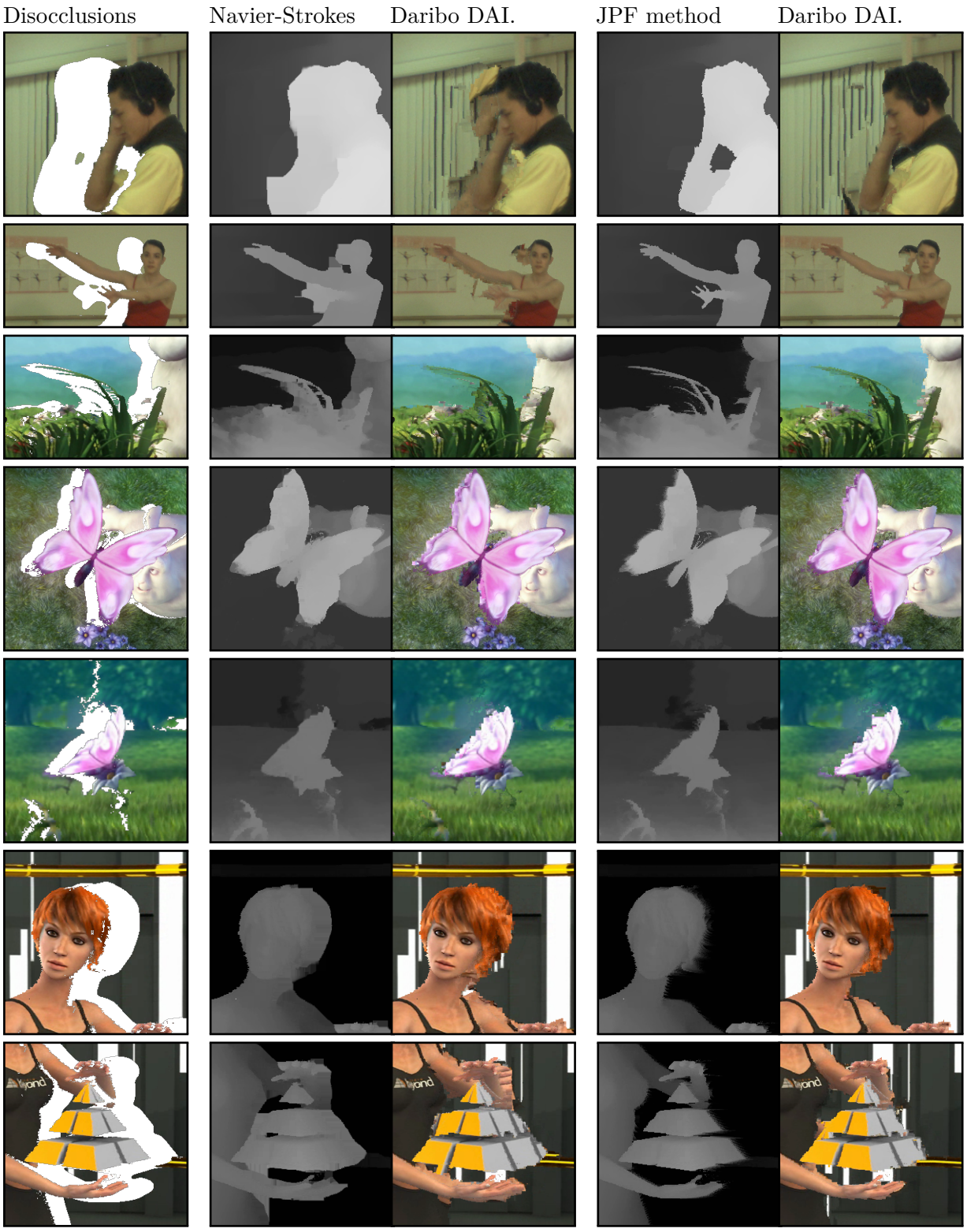


Figure 4.8: Results for Daribo depth-aided inpainting [Daribo & Pesquet, 2010]. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-Strokes inpainting algorithm [Bertalmío et al., 2001] and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

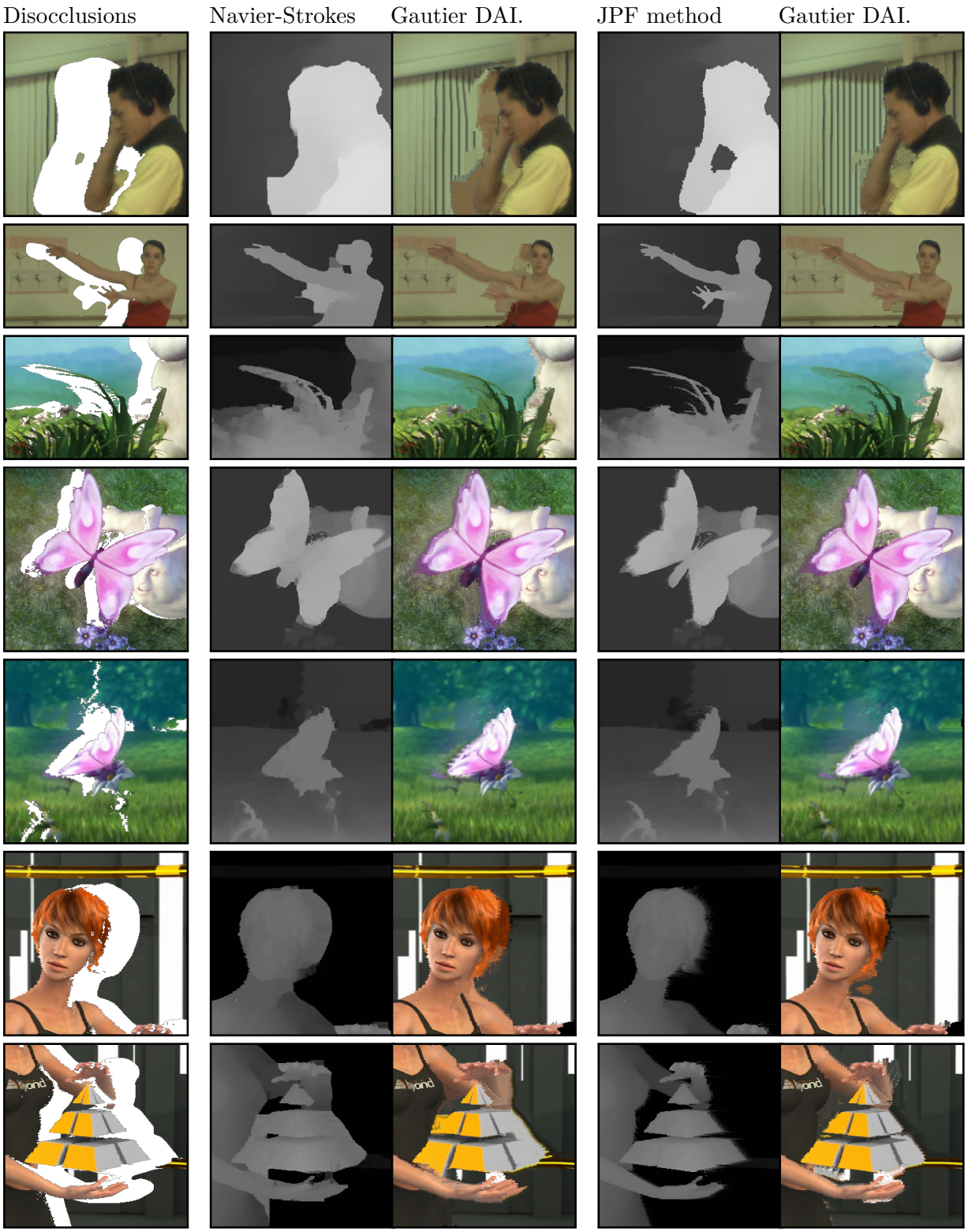


Figure 4.9: Results for Gautier depth-aided inpainting [Gautier et al., 2011]. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-Strokes inpainting algorithm [Bertalmío et al., 2001] and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

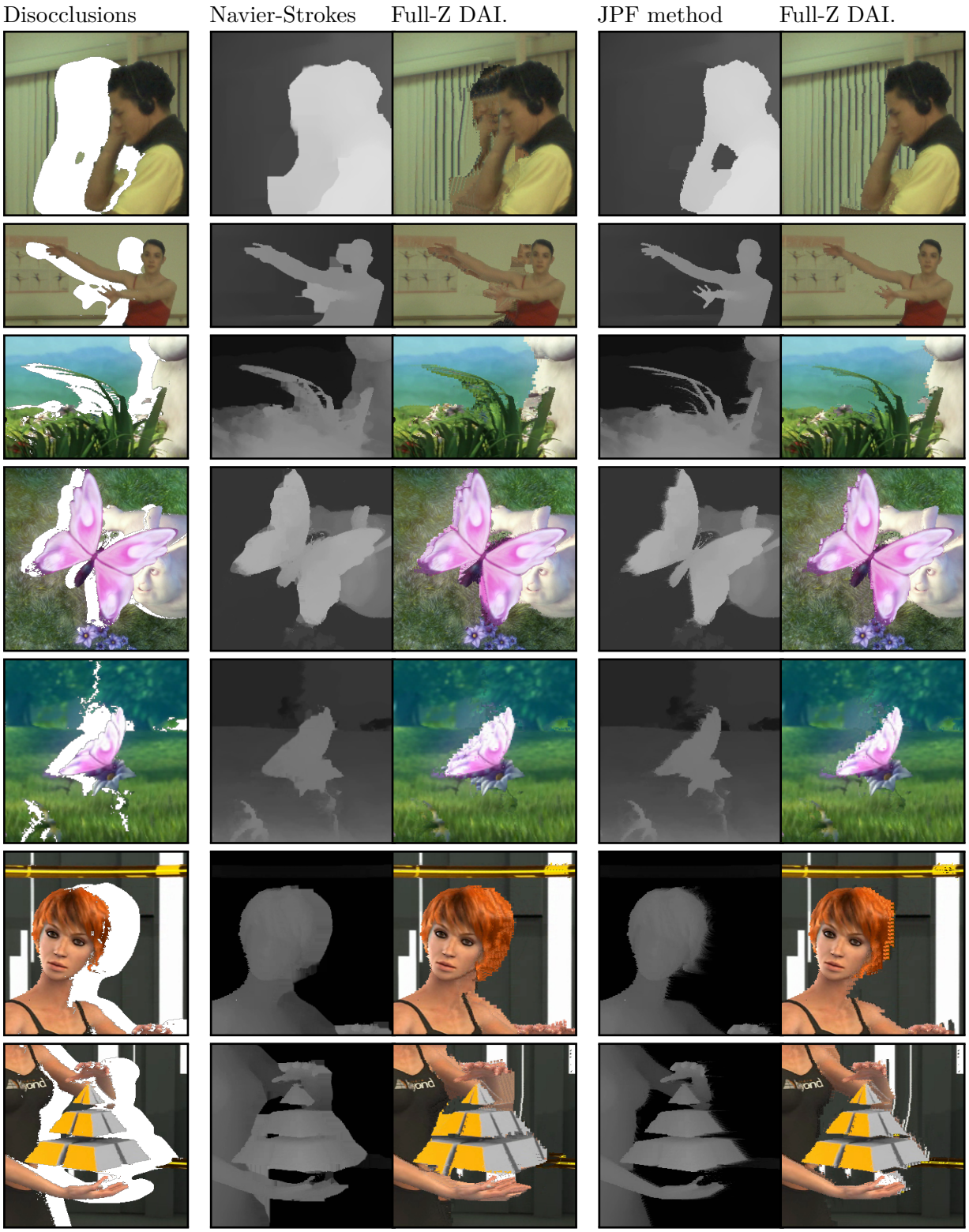


Figure 4.10: Results for proposed full-Z depth-aided inpainting. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-Strokes inpainting algorithm [Bertalmío et al., 2001] and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

The influence of the virtual depth map can be observed by comparing column 3 and 5 of each figure. Errors in depth map from column 2 are amplified by every depth-aided inpainting methods, because some foreground patches are selected to fill in disocclusions. The resulting images, shown in column 3, contain more artifacts than the ones obtained with a correct depth map.

Depth-aided inpainting methods can be compared with each other by analyzing the fifth column of each figure. Rendering results shown in Figures 4.8 and 4.9 still contains blur artifacts along most boundaries, even if the correct depth map is used to conduct the inpainting process. The proposed full-Z depth-aided inpainting method preserves small details, as fingers on row 2 or blades of grass on row 3. Unfortunately, if the depth map is too blurred, some artifacts may persist, as shown in the two last rows.

As a conclusion, the quality of the rendered view is strongly dependent on the quality of the virtual depth map, no matter the depth-aided inpainting method. Synthesizing high quality virtual depth map is thus an interesting challenge for DIBR techniques. The JPF method is well suited for this purpose, because connectivity information is used during the forward projection. Moreover, the proposed full-Z depth-aided inpainting method improves upon state-of-the-art methods by taking into account the correctness of the synthesized depth map.

4.3 Intermediate view interpolation with Floating Texture

Intermediate view rendering methods fill in disocclusions with texture information from many input views [Manning & Dyer, 1996, Müller et al., 2009]. The classical scheme works as follows: intermediate views are first synthesized by projecting each input view onto the virtual viewpoint, using the backward projection described in Figure 3.10; the backward projection removes cracks and sampling artifacts with three time-consuming steps, which are a forward warping step, a filtering step and a backward warping step; the final rendered view is then computed by blending intermediate views together. Disocclusions are thus filled in with corresponding textures from side views. Depending on the correctness of the estimated depth maps, and the accuracy of the cameras' calibration, depth information coming from each view may be inconsistent, resulting in blurring artifacts after blending, as shown in Figure 4.12(d).

The proposed solution uses the Floating Texture approach, introduced in [Eisemann et al., 2008], which is based on an optical-flow warping to correct for local texture misalignment. Other image registration algorithms can be found in [Krutz et al., 2006]. Figure 4.11 presents each step of the proposed intermediate view interpolation with the Floating Texture registration. Each input view plus depth sequence is first forward projected onto the virtual viewpoint. The JPF method is used to warp both color and depth maps. This projection method allows handling cracks without the need for a backward projection. Each intermediate view is then registered using an optical-flow-based warping technique. Optical flow is computed with the algorithm described in [Zach et al., 2007], which provides the best results for Floating Texture registration, compared to other methods [Horn & Schunck, 1981]. Finally, the registered views are blended together using a weighting scheme based on the confidence score, computed in Section 4.1.2.

Figure 4.12 presents an intermediate view, rendered by our proposed method. Blending

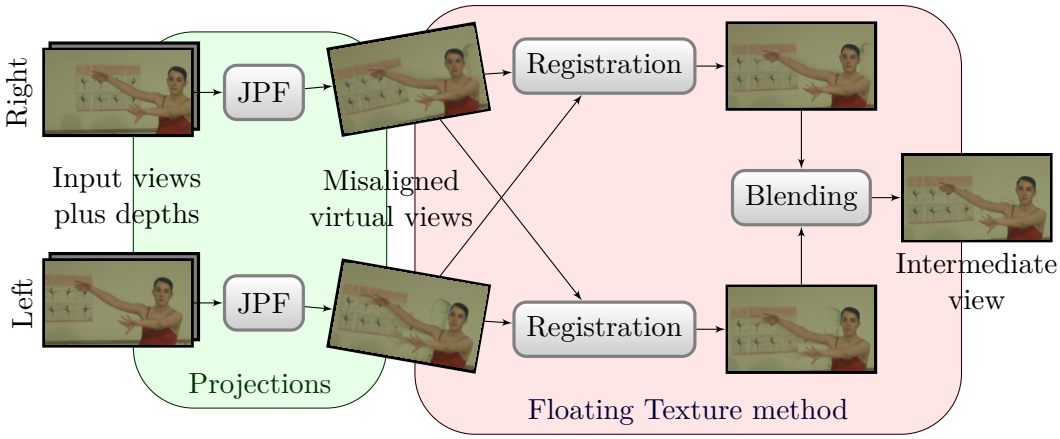


Figure 4.11: Floating Texture scheme for intermediate view synthesis. Each input view is projected onto the virtual viewpoint, using the Joint Projection Filling (JPF) method to fill in disocclusions and preserve contours. Projected views are realigned with the Floating Texture algorithm, then blended together with a weighting based on the virtual view position and confidence score.

intermediate views together, weighted by the confidence score, allows us to fill in disocclusions with real texture. Blurring artifacts may appear if intermediate views are misaligned, as shown in Figure 4.12(d). Applying the Floating Texture approach removes blur on contours and texture details are enhanced, as shown in Figure 4.12(e). One of the flow fields estimated with the method proposed in [Zach et al., 2007] is shown in Figure 4.12(c). Colors represent directions, and luminosities represent norms.

4.4 Conclusion

This chapter describes two DIBR techniques, for virtual view extrapolation and intermediate view interpolation, relying on the Joint Projection Filling (JPF) method to improve the rendering quality.

The JPF method is based on McMillan's occlusion-compatible ordering [McMillan, 1995]. It allows cracks handling and disocclusions filling by ensuring that only background pixels are used during interpolation. The confidence-based pixels shifting avoids ghosting artifacts and texture stretching while sharpening discontinuities. In terms of computational complexity, this JPF method is equivalent to classical point-based projection and can be used with non-rectified views. Synthesized depth maps are very similar to ground truth depth maps.

The first proposed DIBR technique is a virtual view extrapolation, based on a depth-aided inpainting technique. The JPF method is used here to synthesize the depth map of the virtual view with less errors, in order to conduct depth-aided inpainting. Depth-aided inpainting takes into account the high quality of the generated depth map to select correct patches to be duplicated.

The second proposed DIBR technique is a virtual view interpolation method, which uses the Floating Texture algorithm [Eisemann et al., 2008] to sharpen the final view. The JPF method is used here to produce intermediate views without cracks nor disocclusions, in order to enable optical flow estimation.

(a) Projected left view (b) Projected right view (c) Optical flow [Zach et al., 2007]

(d) Blending whithout Floating Texture (e) Blending with Floating Texture

Figure 4.12: Results of Floating Texture algorithm for virtual view synthesis from multiple video plus depth sequence. Figures 4.12(a) and 4.12(b) show the same intermediate view, synthesized from the two side views, and Figure 4.12(c) shows the optical flow computed on that two synthesized views. Figures 4.12(d) and 4.12(e) show the blending result of that two synthesized views respectively without rectification and with Floating Texture rectification.

Part II

Layered Depth Image (LDI) for compact 3D representation

A multi-view video is a collection of video sequences captured for the same scene, synchronously by many cameras at different locations. Associated with a view-synthesis method, a multi-view video allows the generation of virtual views of the scene from any viewpoint [Buehler et al., 2001, Zitnick et al., 2004]. This property can be used in a large diversity of applications [Smolic et al., 2007b], including Three-Dimensional TV (3DTV), Free Viewpoint Video (FTV), security monitoring, tracking and 3D reconstruction. The transmission or storage of huge amount of data contained in a multi-view video sequence needs a compact representation [Merkle et al., 2007].

The compression algorithm strongly depends on the intermediate representation and has a strong impact on the view synthesis methods. Intermediate representations can thus be classified in two classes: geometry-based representations and image-based representations.

Geometry-based representations use a detailed 3D model of the scene. These methods are useful with synthetic video data but they become inadequate with real multi-view videos, where 3D models are difficult to estimate.

Image-based representations are commonly used for reducing the geometric complexity of a scene description. These methods use the acquired videos accompanied by some low-detailed geometric information to synthesize photo-realistic virtual views.

Layer Depth Images (LDI) are an image-based representation where pixels are no more composed by a single color and a single depth value, but can contain several colors and associated depth values. The farther depth pixels, which are occluded from the reference viewpoint, will act to fill in the disocclusions that occur as the viewpoint moves away from the center. This representation is introduced in [Shade et al., 1998] to reduces efficiently the multi-view video size, and offers a fast photo-realistic rendering, even with complex scene geometry.

This part is an overview of the LDI as an intermediate representation of a multi-view video.

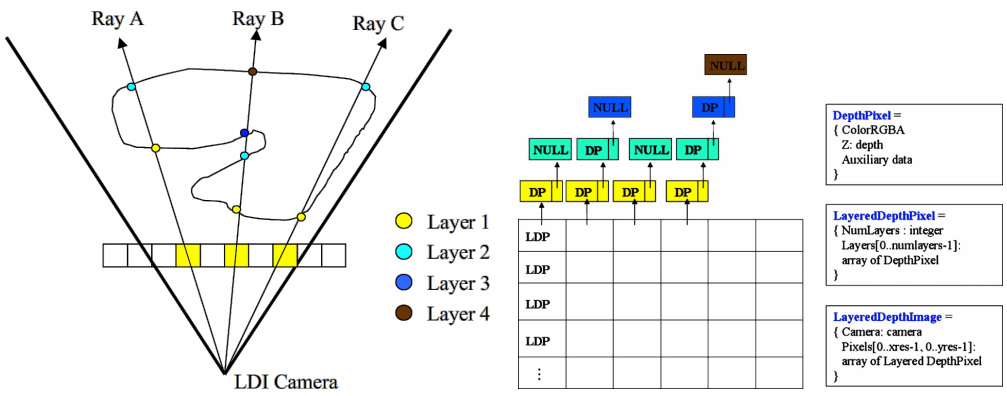
Chapter 5 defines the LDI structure and introduces some state-of-the-art algorithms to build and treat LDI. First, methods for LDI construction are presented, either from 3D models or from captured images. Then, techniques for LDI compression are detailed, based on pixels aggregation or layers filling.

Chapter 6 covers the thesis work by presenting our contributions on LDI compression and construction. The LDI compression method is based on the MVC algorithm, to exploit temporal and inter-layer correlations. Two construction schemes are then proposed. The incremental construction scheme reduces the number of pixels in additional layers while increasing their compactness. The object-based restructuring improves the LDI compression efficiency while minimizing artifacts appearance. The constructed layers are finally compatible with a fast mesh-based rendering, and can also be rendered by the JPF methods presented in Section 4.1.

Chapter 5

Background work on LDI: State-of-the-art

The concept of Layered Depth Image (LDI) was first introduced in [Shade et al., 1998], for complex geometries. A LDI potentially contains multiple depth pixels at each discrete location in the image. Instead of a 2D array of depth pixels (a pixel with associated depth information), a LDI is a 2D array of layered depth pixels. A layered depth pixel stores a set of depth pixels along one line of sight sorted from front to back order. The front element in the layered depth pixel samples the first surface seen along that line of sight; the next pixel in the layered depth pixel samples the next surface seen along that line of sight, etc. The temporal extension of LDI is called LDV, for Layered Depth Video. Figure 5.1(a) presents the LDI representation as defined in [Shade et al., 1998].

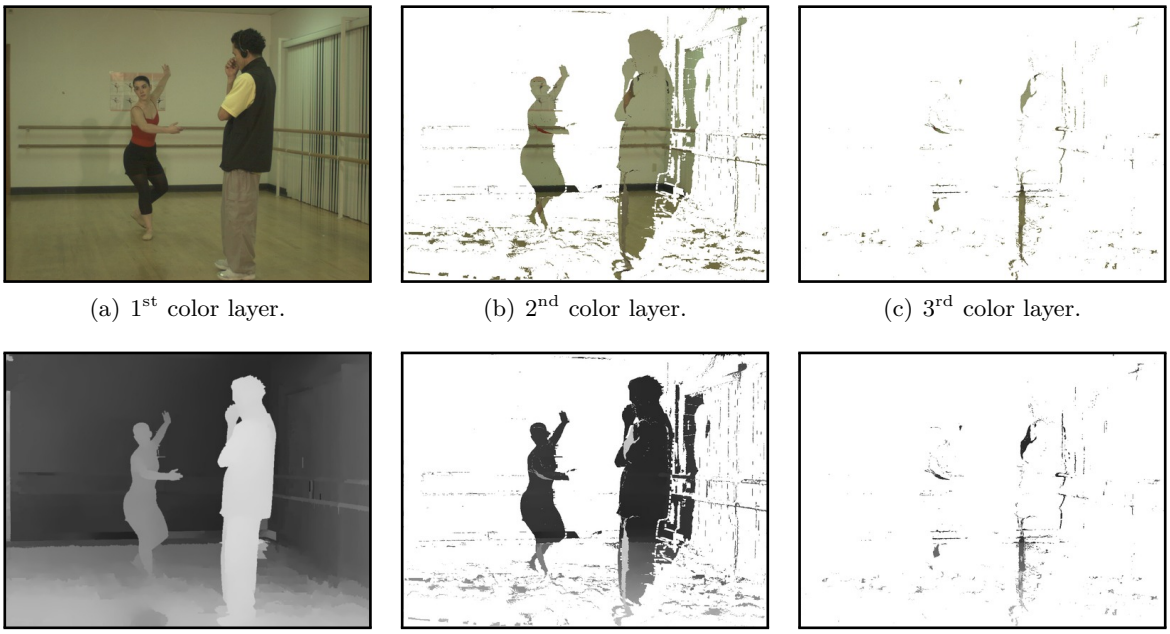


(a) (b)

Figure 5.1: Structure of a Layer Depth Image (LDI) (from [Yoon et al., 2007]).

Figure 5.2 shows the first layers of such a LDI, where pixels from a same layer are grouped in a separate image. One can observe that the first layer contains all pixels which are visible from the reference viewpoint, it is the classical 2D image. The other layers contain pixels in the camera scope, but hidden by objects in previous layers.

The number of layers is not limited by the definition, and depends on the complexity of the scene. The size of the representation grows only linearly with the observed depth complexity in the scene, and not with the number of input views. In practice, a LDI is often limited to a few



(a) 1st color layer. (b) 2nd color layer. (c) 3rd color layer.

(d) 1st depth layer. (e) 2nd depth layer. (f) 3rd depth layer.

Figure 5.2: First layers of a LDI frame.

number of layers.

Section 5.1 introduces existing techniques for constructing such a LDI structure, from either 3D models or real captured images.

Section 5.2 presents common methods for LDI compression, based on either pixels aggregation or layers filling.

5.1 Classical LDI construction

Various methods have been proposed in the literature to construct LDI representations. They principally differ on the type of input data they require to operate. The first detailed method works with a 3D model, whereas the two other presented methods work on real acquired video sequences.

5.1.1 LDI from 3D models

A LDI is easier to construct if the 3D model of the scene is already known. In [Shade et al., 1998], the authors propose to use an adapted ray-tracing method.

At first, a virtual light-ray is emitted from the reference camera position, in direction of each pixel of the LDI. All intersections between this light-ray and the 3D surface define a new depth pixel which is then introduced in the LDI structure.

The quality of the reconstruction from another viewpoint will depend on how closely the distribution of depth pixels in the LDI, when warped to the new viewpoint, corresponds to the pixel density in the new image. To ensure a high rendering quality of the virtual view, the authors propose to increase the density of the light-ray used to construct the LDI. Unfortunately, a high light-ray density also increases the size of the LDI, which is an important feature for storage and

transmission. A compromise between rendering quality and representation compactness thus has to be found by adapting the light-ray density.

5.1.2 LDI from real images

Several methods similar to space-carving [Kutulakos & Seitz, 1998, Shade et al., 1998, Müller et al., 2004] and plane sweeping [Collins, 1996] have been proposed to construct a LDI from a set of real captured pictures. The regular voxelization used in space-carving techniques is replaced by a view-centered voxelization similar to the LDI structure.

This approach enables the construction of LDI's from images that do not contain depth information for each pixel. It requires the knowledge of extrinsic and intrinsic camera parameters.

Other methods use a disparity estimation algorithm [Hartley & Zisserman, 2004, Kim & Sikora, 2007, Sourimant, 2010a] to estimate the depth value of each pixel from each image. Methods to generate LDI based on multi-video plus depth sequences are detailed in the next section.

5.1.3 LDI from multiple video plus depth sequences

Generating LDI from multi-video plus depth sequences is the most studied construction scheme. Given a set of viewpoints and one depth map per view, the classical algorithm for LDI construction [Cheng et al., 2007a, Yoon et al., 2007] proceeds in three steps, summarized in Figure 5.3. First, an arbitrary viewpoint is chosen as the reference viewpoint. This reference viewpoint is usually chosen among input viewpoints, but this is not an obligation. Then, each input view is warped onto this reference viewpoint using a DIBR method. Finally, all these warped views are merged into a single LDI model, where each pixel position may contain many depth pixels.

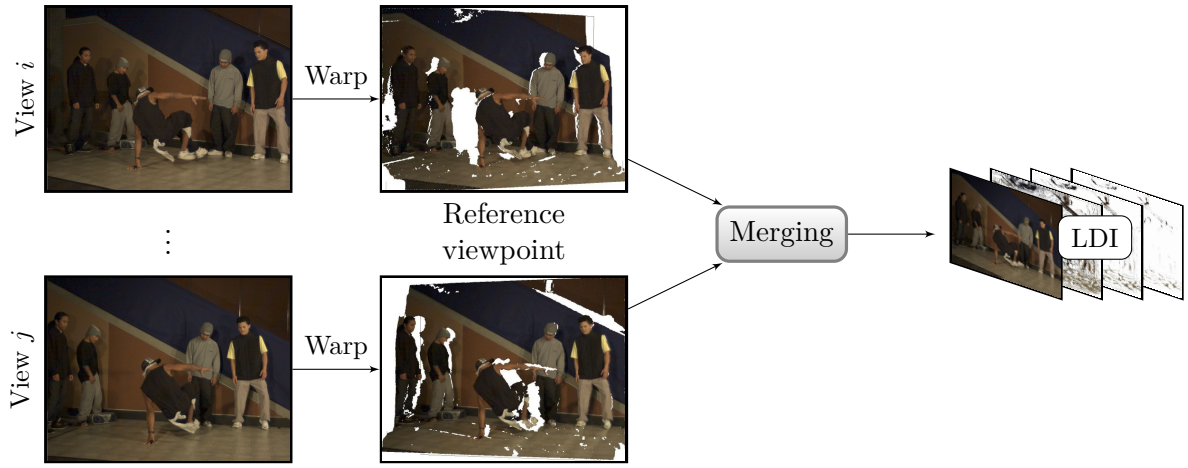


Figure 5.3: Classical LDI construction scheme.

There are many merging policies depending on the application. Keeping all depth pixels results in unnecessary highly redundant layers, due to the inter-view correlation. It is preferable to keep at each pixel location, only pixels whose depth value significantly differs from that of the others. A threshold Δ_d is often used on the depth value to eliminate pixels with very similar depth values.

The first three layers of such a LDI are presented in Figure 5.4. Layered pixels are ordered according to their depth value. The first layer is composed of pixels with smallest depth, the second layer contains pixels with second smallest depth, and so on. Except for the first layer which is the reference view, one can observe that layers are partially empty, and non-empty pixels are sparsely distributed all over the layer. Furthermore, many pixels are redundant between the layers. These characteristics make it difficult to efficiently compress the LDI.

(a) 1st layer (b) 2nd layer (c) 3rd layer

Figure 5.4: First layers of a LDI frame. 8 inputs views are used for the generation. ($\Delta_d = 0.1$)

The temporal extension of LDI is called LDV, for Layered Depth Video. MVD2LDV is the reference software, developed by MPEG, which generates LDV from MVD [Tanimoto et al., 2008]. This software has some limitations. The software requires exactly three input views to generate a LDI with exactly two layers. The implemented algorithm only memorizes the nearest foreground layer, and the furthest background layer. The software requires a set of rectified views, i.e. cameras should be perfectly aligned and have identical intrinsic parameters. This constraint simplifies projection equations, because all epipolar lines are horizontal. In practice, it is hard to obtain rectified views with a real capture system. A rectification step is thus needed on the acquired views as a preprocessing step.

5.2 LDI compression schemes

Because of the special data structure of the LDI, existing image compression methods cannot be applied directly or are not very efficient. There are three key characteristics of the LDI data. It consists in multiple layers; the content in the back layer is sparse; and each pixel consists of multiple property values, including the color, depth, and possibly alpha value. The key to develop an efficient LDI compression algorithm is thus to deal with these three characteristics.

After generating LDI frames from the natural multi-view video with depth, each LDI frame is separated into three components: color, depth, and the number of layers (NOL) [Yoon et al., 2006a]. NOL could be considered as an image containing the number of layers at each pixel location. Since the NOL information is very important to restore or reconstruct multi-view images from the decoded LDI, it is encoded by using the H.264/AVC intra mode. Color and depth components consist in layer images, respectively. These components are processed by either data aggregation or layer filling to apply classic video coding methods.

In this section, we present two kinds of encoding algorithms for color and depth components of LDI, which are data aggregation and layer filling [Duan & Li, 2003, Yoon et al., 2006a,

Cheng et al., 2007b, Yoon & Ho, 2007].

5.2.1 Aggregation

The first method aggregates scattered pixels into the horizontal or vertical direction [Duan & Li, 2003]. The LDI layers are sparse and not of rectangular support. The deeper the layer is, the fewer number of pixels there is. A preprocessing step is thus proposed to aggregate the data in each layer. The aggregation operation is similar to the data push operation used in MPEG-4 Shape Adaptive DCT (SA-DCT).

The aggregation proposed by the authors is simple horizontal aggregation: all the pixels in the same line are pushed to the left of the line. Figure 5.5 shows an example of the aggregation operation.

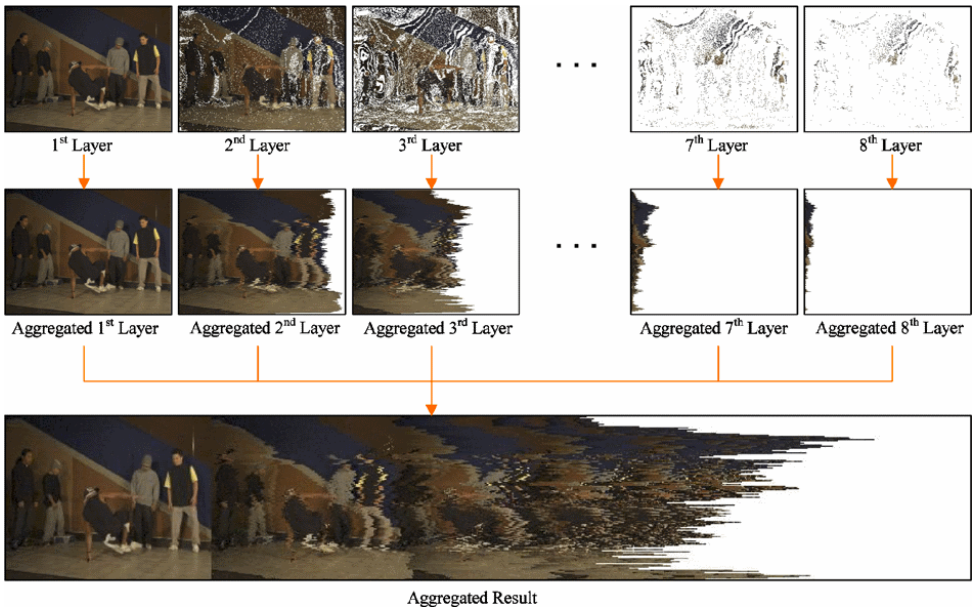


Figure 5.5: Data aggregation with the horizontal direction (from [Yoon et al., 2007]).

Since the layer mask derived from the NOL is available at the decoder, the aggregation can be easily inverted. The aggregation algorithm lengthens the horizontal object segments, and improves the transform efficiency. The authors have also tried to further aggregate the data along the vertical direction, i.e., to push the data again along the vertical axis. However, experiments show that such further aggregation does not provide additional performance improvement, in term of rate-distortion measures.

We may use existing coding tools to compress the component images. One possible approach is the MPEG-4 SA-DCT mode, where all component images are treated as a video sequence and compressed by MPEG-4. An alternative approach is first to pad the component image to a rectangular image, and then to compress it with a rectangular still image coder such as the JPEG-2000.

One problem of the data aggregation is that the resultant images have severely different color distributions. It leads to poor coding efficiency because the prediction among aggregated images is difficult. Blocks of pixels contain high frequencies and can not be predicted from their neighbors. Aggregation also produces many flickering artifacts which prevent the exploitation

of temporal prediction. In [Duan & Li, 2003], the authors show that the advantage of the data aggregation outweighs its disadvantage.

5.2.2 Filling

The second method to avoid empty areas in additional layers is to fill them by copying pixels at the same position, but from another layer. Since the first layer has no empty pixels, we can use pixels in the first layer to fill the other layers. This method increases the prediction accuracy of H.264/AVC, therefore data size can be reduced further. The newly filled pixels can be eliminate in the decoding process by using the information of NOL if it is sent to the decoder, or by comparing layers and detecting duplicated pixels. Figure 5.6 presents the first three layers of a LDI after filing each additional layer with pixels from the previous layer.

(a) 1st layer (b) 2nd layer (c) 3rd layer

Figure 5.6: First layers of a LDI after filling each layer with pixels from previous layers.

5.3 Conclusion

As a conclusion, a LDI is an efficient representation of a 3D scene. Texture information from multiple input views are grouped into a single structure as observed from a single viewpoint. The additional layers provide enough information to handle disocclusions generated by DIBR techniques. Textures seen from many cameras are described just once in the LDI structure, reducing inter-view correlation.

State-of-the-art LDI construction methods aim to construct LDI from 3D model or real scene. However, no fully satisfying methods yet exist to obtain a compact LDI from multi-view videos. Existing methods produce correlated layers with high redundancy between layers.

State-of-the-art LDI compression methods aim to reduce the size of such a LDI but have various defects. The aggregation method introduces high frequencies and is thus not adapted to temporal extension. The filling method introduces inter-layer correlation which should later be exploited.

An alternative construction algorithm, that we call I-LDI (for Incremental LDI), is described in Section 6.2. This incremental algorithm reduces the number of pixels inside a LDI while enhancing the final rendering quality. A layer reorganization algorithm is proposed in Section 6.3 to enhance depth discontinuity. This pixel reorganization improves the quality of synthesized views, in a rate-constrained context. Pixels from each layer are reorganized to enhance depth

continuity. A compression method is also proposed which exploits temporal correlations and inter-layer correlations.

Chapter 6

Layer Depth Image representation

This chapter presents our contributions to LDI compression efficiency, by addressing both LDI construction, compression, and rendering. The compression of the layers is processed by the MVC coder, which exploits temporal correlations, and cancels the undesirable copying effect of the filling process (see Section 5.2.2). The proposed construction scheme uses an incremental process to generate a LDI from a multiple video plus depth input sequence. The generated layers contain only occluded pixels, which reduces their completion and correlation. Pixels in these LDI are then reorganized to enhance depth continuity in each layer. This continuity improves the compression efficiency while avoiding some compression artifacts. Finally, the LDI rendering is done either by a fast mesh-based rendering method, or by the advanced JPF method as described in Section 4.1. The first method is fast enough for real time rendering in a height-view autostereoscopic display, whereas the second method is slower but produces high quality virtual views.

This chapter is composed of four sections.

Section 6.1 provides a LDI compression scheme, based on MVC techniques, which exploit both temporal and inter-layer correlations.

Section 6.2 introduces a novel I-LDI construction scheme, which uses an Incremental process to generate LDI layers from multiple video plus depth sequences.

Section 6.3 describes a novel Object-based LDI representation, improving the compression efficiency and the synthesized virtual views quality in a rate-constrained context.

Section 6.4 presents two rendering methods to synthesize virtual views from LDI layers, focusing respectively on efficiency and rendering quality.

6.1 Compression of LDI sequences with AVC/MVC

Although the LDI sequences can be encoded as collections of separate LDI, it is not efficient due to the temporal redundancy. In [Yoon et al., 2007], the authors proposed a framework for multi-view video coding using layer depth images. In this framework, components of layers in same level are treated as video sequences and encoded by traditional codecs such as H.264/AVC. To achieve better motion estimation efficiency, they use filling methods to interpolate the holes (empty pixels) in back layers by pixels at same position from first layer (see Section 5.2.2).

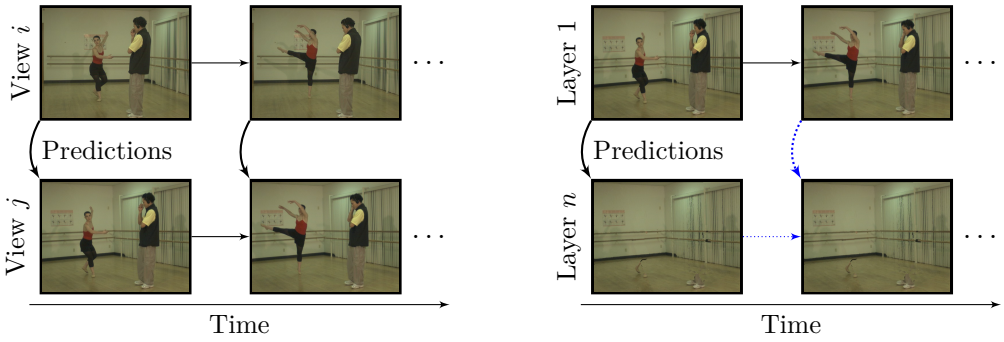
This compression method exploits temporal correlations by the use of traditional video codecs, but introduces duplicated data during the layer filling process, which increases the size of the data. A proposition is to use the Multi-View Compression codec (MVC) to exploit both temporal correlation, and inter-layers correlations.

This section first explains how to adapt the MVC technique to compress LDI data, then it presents some preliminary results on compression efficiency.

6.1.1 LDI coding with AVC/MVC

The principle of image prediction has been extended to multi-view sequence by the AVC/MVC technique. The MVC codec, widely described in Section 2.2.3, is represented in Figure 6.1(a).

As explained is Section 2.2.3, the main limitation of the MVC coding scheme is the prediction of additional view, based on a motion flow. This motion flow does not exploit the scene geometry, and should thus be computed and coded for each additional view and for each frame.



(a) MVC on multi-view data

(b) MVC on multi-layer data

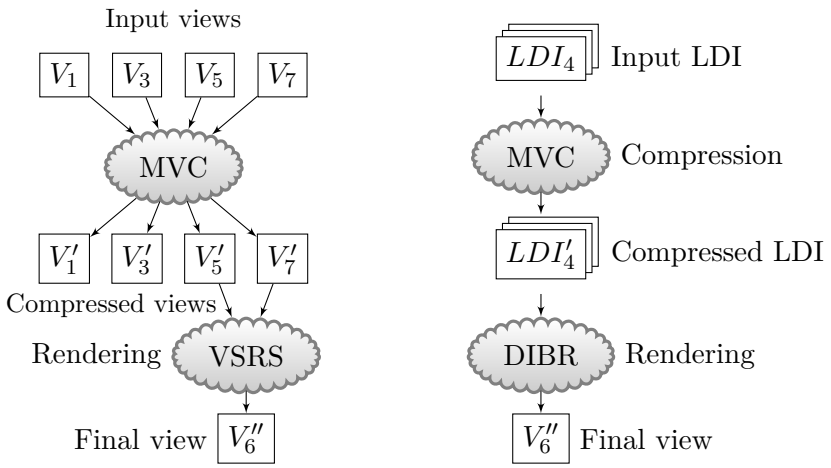
Figure 6.1: Block-diagrams of the configuration of MVC codec for compressing 3D sequences. In Fig 6.1(a), MVC is used with multi-view sequence to exploit both temporal and inter-view correlations. In Fig 6.1(b), MVC is used with sequence of LDI to exploit both temporal and inter-layer correlations.

In this section, we propose to use the MVC method on the sequence of LDI, by considering each layer as a "view" of the scene. The first layer, which is the real view of the scene, is first compressed as a classical video, using the temporal prediction scheme introduced in Section 2.2.2. The additional layers are then predicted from a layer on top, to complement the temporal prediction. Figure 6.1(b) presents the block-diagrams of the MVC compression scheme, used on LDI sequences.

6.1.2 Experimental results

In this section, two multi-view compression schemes are compared. Both use the MVC codec to exploit correlation between either views of multi-view sequence, or layers of LDI sequence. The two experimental protocols are presented in Figure 6.2.

In the first place, views 1, 3, 5 and 7 from "Breakdancing" data set are coded with the MVC algorithm with various quantization parameters, varying from QP=20 to QP=55. Depth maps associated to these views are also coded with the MVC algorithm as a separate step. View 1



(a) MVD compression (MVC)

(b) LDI compression (MVC)

Figure 6.2: Block-diagrams of the two compared compression schemes. Figure 6.2(a) presents the classical multi-view compression scheme, using the MVC codec from MPEG [Tanimoto et al., 2008]. Figure 6.2(b) shows the proposed LDI compression scheme, using the MVC codec on different layers.

is used as the reference view to predict view 3, which is used to predict view 5, itself used to predict view 7. The compressed views 5 and 7, together with their compressed depth maps, are finally used to synthesize virtual views onto viewpoint 6, using the MPEG/VSRS software [Tanimoto et al., 2008]. This process is presented in Figure 6.2(a).

In the second place, a LDI restricted to three layers, is constructed from three input views: the reference view 4 and side views 3 and 5. The MVC algorithm is processed on all color layers one side, then on all depth layers on the other side. A JPF-based rendering method (see Section 4.1) is finally used on the decoded sequence of LDI to synthesize view 6. This process is presented in Figure 6.2(b).

In both cases, the synthesized view 6 is compared to the captured view 6 using two distortion metrics. The resulting rate-distortion curves are presented in Figure 6.3. The used metric for image comparison is either PSNR (Left) or SSIM (right).

One can observe that the LDI format provides some gain in quality compared to conventional standard MVC, when the target bitrate is lower than 10Mbit/s. For higher bitrate, the LDI format reaches quality saturation. This is due to projection errors and contours misalignment. In [Cheng et al., 2007b], the authors perform the same experiments and obtain similar results.

The two representations are not efficient in the same use case. VSRS is designed to synthesize intermediate views, while LDI are can easily synthesize distant views. That explains why the two experiments do not use the same input views in order to synthesize the final view 6.

To improve compression efficiency, and increase the rendering quality, next sections propose to remove some unnecessary redundancies between layers, and to organize pixels in these layers in order to enhance depth continuity.

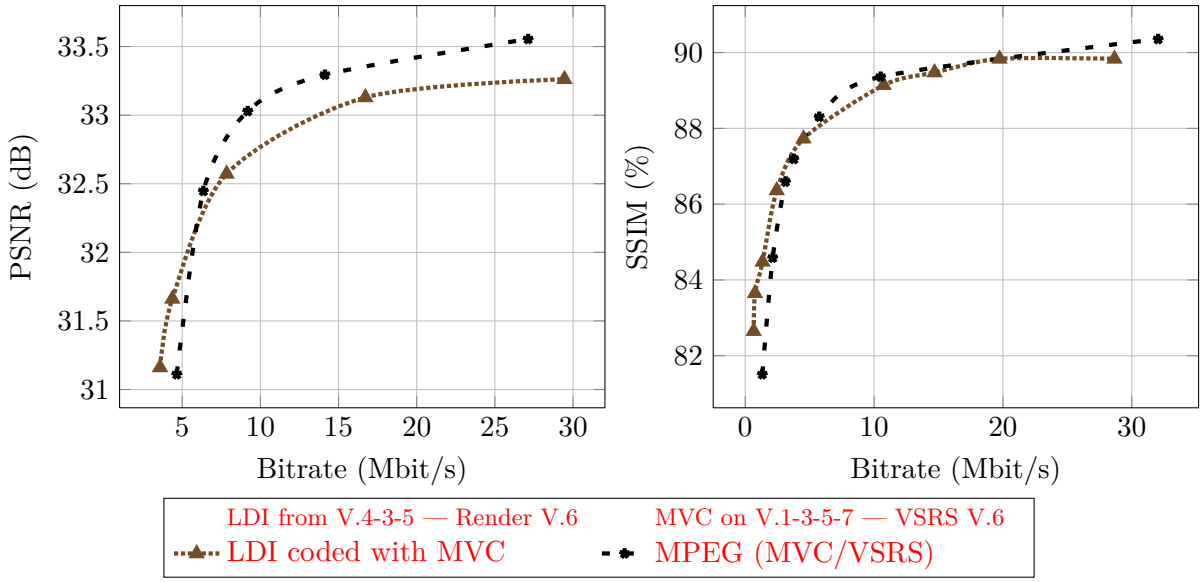


Figure 6.3: Rate distortion curves of Breakdancing multi-view video, firstly compressed by MVC and rendered with VSRS algorithm, and secondly for LDI compressed by MVC and rendered by a point-based projection.

6.2 Incremental-LDI (I-LDI)

The Layered Depth Image (LDI) is an intermediate representation between acquired multi-view videos and display devices. An efficient algorithm should be used to produce LDI from multi-view sequences. Various algorithms are described in Chapter 5, but classical algorithms produce LDI with many layers. Layers are also correlated and many pixels are redundant between layers.

In [Müller et al., 2008a], the authors introduce the concept of residual information, in order to remove correlations from multi-view data set. The idea is to keep only the reference view together with discovered information necessary to synthesize side views. The residual information is thus defined as the texture information which is disoccluded during a projection. It can be obtained by projecting the reference view onto each additional viewpoint and by extracting disoccluded texture from the corresponding acquired view. The authors propose to preserve these disoccluded depth-pixels onto their original viewpoint. Figure 6.4 shows the extraction of residual information.

(a) Reference view. (b) Ref. view after projection. (c) Discovered information from side view.

Figure 6.4: Extracting information from uncovering additional views.

This section describes an Incremental algorithm for LDI construction (I-LDI) from multi-view plus depth data sets. The method is based upon the extraction of residual information proposed in [Müller et al., 2008a], to avoid increasing the number of layers containing a low density of pixels. The JPF method, described in Section 4.1, is used to handle cracks and ghosting during the projection.

Section 6.2.1 introduces the incremental scheme for LDI construction, names I-LDI, which significantly reduces the number of layers.

Section 6.2.2 presents the results obtained on Ballet MVD data sets, which shows that extra layers in I-LDI contain only 10% of first layer pixels, compared to 50% for classical LDI.

6.2.1 Incremental scheme for LDI construction

This section introduces the incremental construction scheme, based on residual information extraction [Müller et al., 2008a]. The algorithm is incremental, which means that input views are treated sequentially, in a fixed order.

The algorithm starts with an empty I-LDI, for which the reference viewpoint is chosen freely. The reference viewpoint is often one of the input viewpoints, but is sometimes an intermediate viewpoint between two input views.

The following three steps are then iterated for each input view, in a fixed order. First, the I-LDI is processed by a view synthesis algorithm to synthesize one acquired viewpoint. Then, the synthesized view is compared with the original acquired view, and the discovered information is isolated. Finally, this discovered information is warped back into the reference viewpoint and inserted in the I-LDI layers. These three steps are iterated for every input viewpoint. Figure 6.5 illustrates the incremental construction scheme.

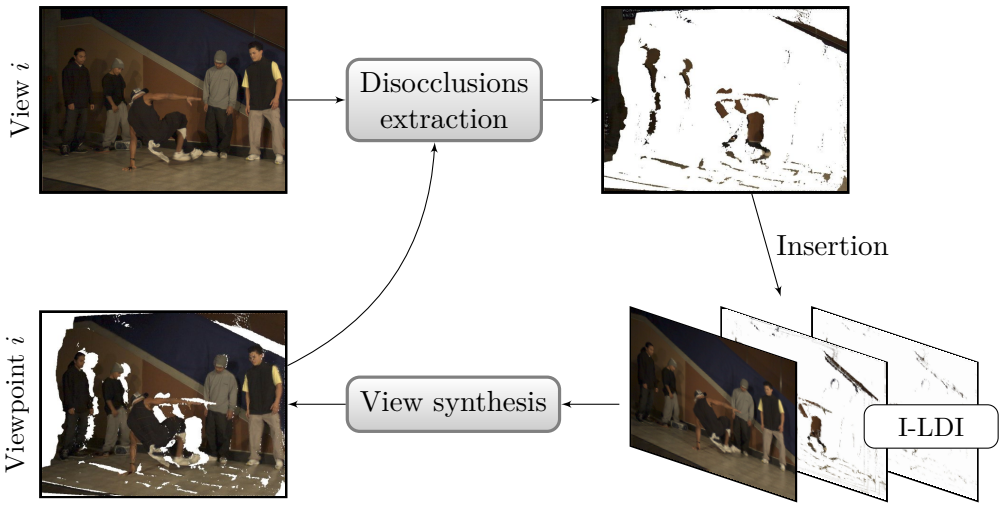


Figure 6.5: Step of I-LDI construction for view i, with residual information extraction.

The first three layers of such an I-LDI are presented in Figure 6.6. Compared to LDI layers, I-LDI layers contain fewer pixels, and these pixels are grouped in connected clusters.

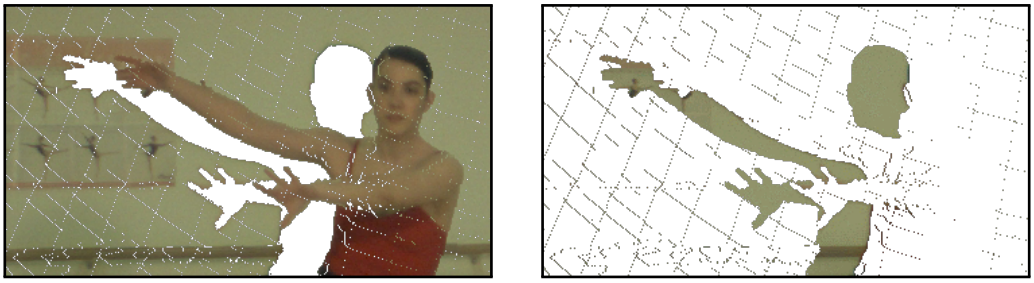
Thanks to this method, only required residual information from side views is inserted, and no pixels from already defined areas are added to the I-LDI. On the other side, all the information present in the MVD data is not inserted in the I-LDI, reducing the correlation between layers.

(a) 1st layer (b) 2nd layer (c) 3rd layer

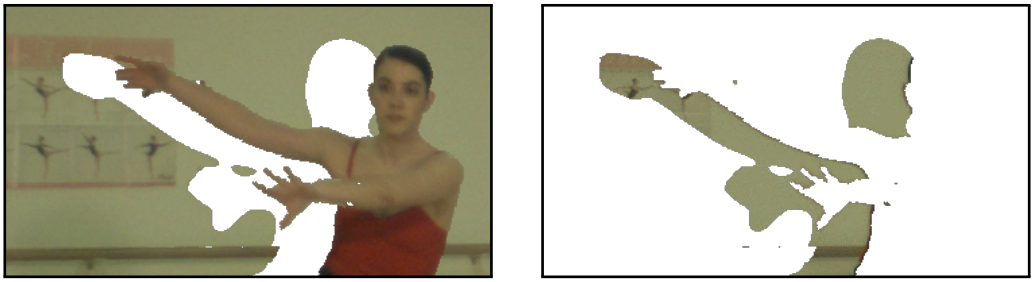
Figure 6.6: First layers of an I-LDI frame. All 8 inputs views are used for the generation, in a B-hierarchical order.

The incremental scheme uses several parameters, like the rendering method used to synthesize virtual view from the I-LDI, and the ordering of input views. These parameters are discussed in the following.

The rendering method used at each iteration to synthesize virtual views from I-LDI, should be fast but should not introduce artifacts. A classical point based projection generates three kinds of artifacts: disocclusions, cracks and ghosting artifacts. We use the JPF method to handle cracks and ghosting artifacts without the need of an additional filtering step. The JPF method also fills in small disocclusions, what reduces the size of the residual information to be inserted in the I-LDI.



(a) Point-based view synthesis.



(b) The JPF method.

Figure 6.7: Residual information obtained with 6.7(a) a point based view synthesis and 6.7(b) the JPF method. The first column shows the rendered view of an intermediate I-LDI and the second column shows the corresponding residual information which should be inserted into the I-LDI.

Figure 6.7 presents a typical residual information obtained by comparing an acquired view

with the corresponding synthesized view. Two different views synthesis methods are compared, which are either a naive point based projection, or our proposed JPF method, described in Section 4.1. In both cases, textures corresponding to large disocclusions are well detected and thus inserted into the I-LDI. Pixels corresponding to cracks or small disocclusions are not useful and the JPF method allows us to remove them from residual information.

The input views ordering has a low impact on the final I-LDI, except for the first input view to be used, which contributes to a major part of the final I-LDI. This view should thus be the one for which the camera viewpoint is the nearest to the reference viewpoint of the I-LDI. The reference viewpoint is very often chosen as one of the input cameras, in this case, the first layer of the I-LDI is almost identical to the reference view. The only difference is located around depth discontinuities, where the JPF method uses a confidence scoring to remove ghosting artifacts, as explained in Section 4.1.

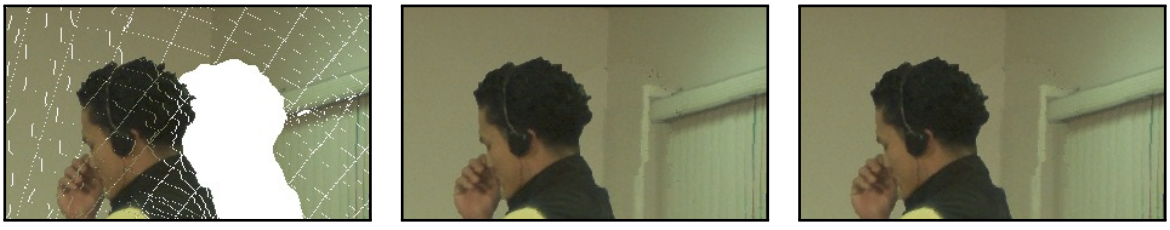
Other views can then be ordered arbitrarily, but local rendering artifacts may appear, depending on views insertion order. Best results are obtained when input views are sorted by increasing distance compared to the reference camera. In that case, each input view contributes to the LDI construction by bringing little bands of pixels

6.2.2 Experimental results

Experiments have been conducted on ten random frames from "Ballet" data set from MSR [Zitnick et al., 2004]. Parameters of the 8 acquisition cameras and all associated depth maps are already estimated and provided with the video.

The viewpoint number 4 is considered as the reference viewpoint. For the LDI construction, all 8 acquired views are warped into the reference viewpoint. A small merging threshold value $\Delta_d = 0.1$ is used (see Section 5.1.3 for explanations). For the I-LDI construction, views are inserted in a B-hierarchical order (4; 0; 7; 2; 6; 1; 5; 3).

Figure 6.8 shows the result of a virtual view synthesis from a LDI and an I-LDI. One can observe that both LDI representations enable to fill the disocclusion area with the real background texture. The visual rendering is of similar quality with both LDI and I-LDI construction schemes.



(a) Disoclusions. (b) LDI rendering. (c) I-LDI rendering.

Figure 6.8: Virtual view synthesis from a LDI and an I-LDI. There is no significant rendering difference between the two LDI representations.

For both construction schemes, all 8 input views are used, but all pixels from each view are not inserted in the LDI. When using the classical LDI construction scheme, some pixels are ignored because a similar pixel, according to the depth threshold Δ_d, is already in the LDI.

When using the incremental construction scheme, only residual data are inserted into the LDI, and all other pixels are ignored.

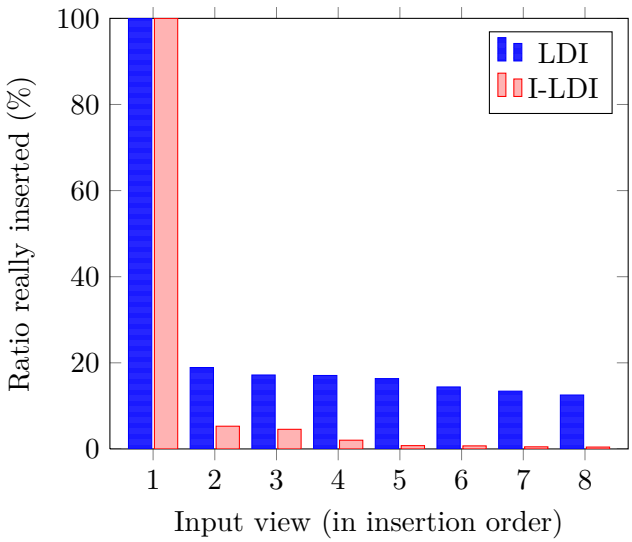


Figure 6.9: Percentage of pixels from the different views kept on the layers of LDI and I-LDI.

Figure 6.9 presents the ratio of pixels from each view which is really inserted in the LDI. We can observe that, compared to classical LDI, fewer pixels are inserted from view 5 to 8, which means that these views become almost useless with the I-LDI construction scheme. Using only a subset of acquired views (the reference and two side views) provides almost the same I-LDI layers.

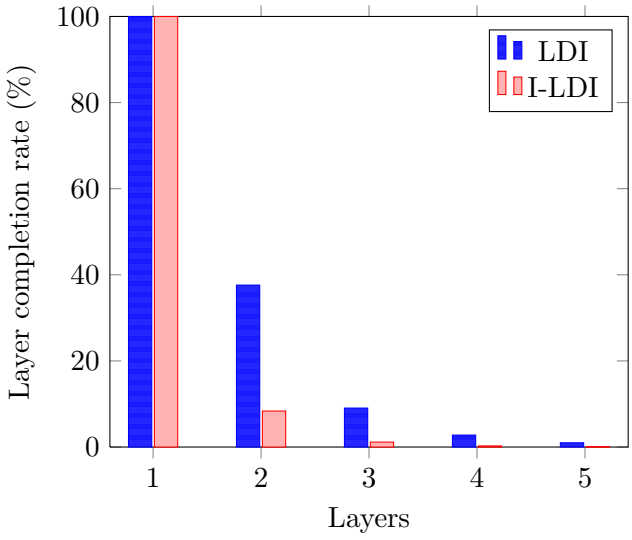


Figure 6.10: Layers completion rate for LDI and I-LDI.

Figure 6.10 shows the ratio of defined pixels in each layer for both LDI and I-LDI construction schemes. For both constructions, the first layer contains 100% of its pixels. Differences appear for subsequent layers. For the LDI, subsequent layers represent more than 50% of the size (in number of pixels) of the first layer, whereas for the I-LDI, extra layers represent less than 10%. One can observe than layers beyond the 3rd one are quite empty, and can thus be ignored.

Figure 6.11 presents the rate-distortion curves of I-LDI compressed with MVC, and compares it with the one obtained with LDI. The used metric for image comparison is either PSNR (Left) or SSIM (right). The experimental protocol is divided in two steps.

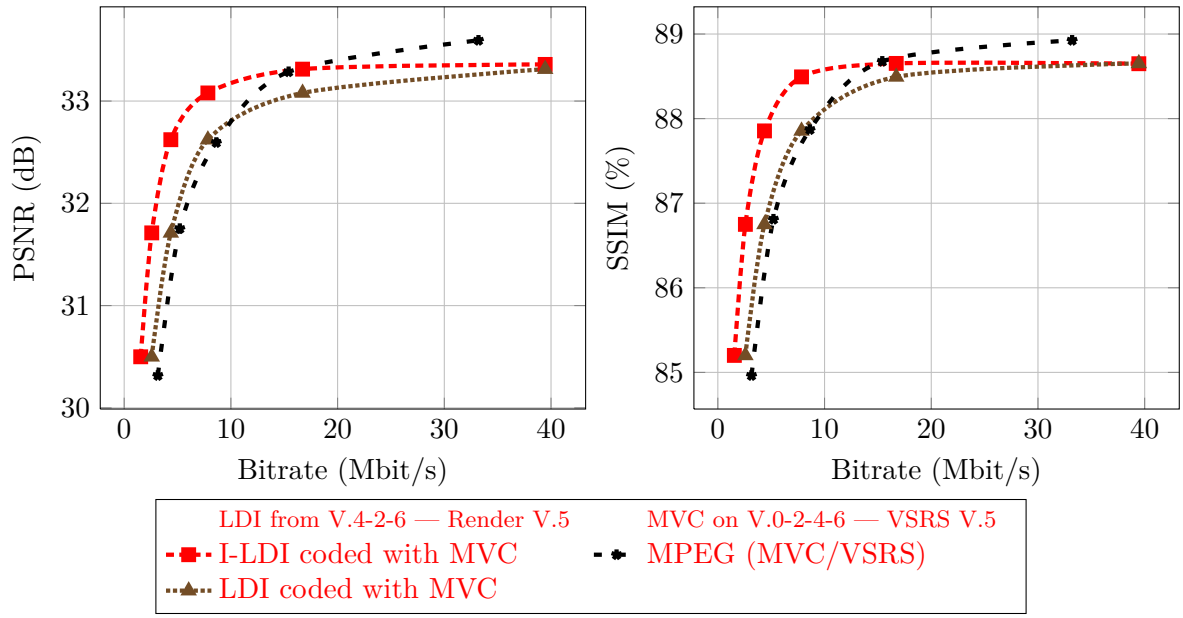


Figure 6.11: Rate distortion curves firstly for LDI (incrementally constructed or not) compressed by MVC and rendered by our point-based projection, and secondly for multi-view video compressed by MVC and rendered with VSRS algorithm. Left is for PSNR, and right for SSIM.

In the first place, views 0, 2, 4 and 6 from "Ballet" data set are coded with the MVC algorithm with various quantization parameters, varying from QP=30 to QP=55. Depth maps associated to these views are also coded with the MVC algorithm as a separate step. The compressed views 4 and 6, together with their compressed depth map, are finally used to synthesize virtual views onto viewpoint 5, using the MPEG/VSRS software [Tanimoto et al., 2008].

In the second place, a LDI restricted to three layers, and an I-LDI with two layers, are constructed from three input views: the reference view 4 and side views 2 and 6. Color layers and depth layers are coded with the MVC algorithm as separate step. After decoding, the sequence of LDI is used to synthesize view 6.

One can observe that the incremental construction scheme generates a dense LDI representation, which improves both compression efficiency, and rendering quality, compared to classical LDI construction, presented in Section 5.1.3. This advanced construction scheme increases the maximum bitrate (from 13 to 15 Mbit/s), where the rate-distortion curve of the LDI representation is higher than the one of the MVD representation.

6.2.3 Conclusion

The last section has presented an incremental procedure to generate LDI from natural multiple image plus depth data. The minimum information necessary to fill disocclusion areas is inserted into LDI layers which makes layers easier to compress. They contain 80% less pixels, and they have a more compact distribution.

Results show that the number of pixels in additional layers is much lower than the number of

pixels in the first layer. In practice, only a few number of layers can be kept, without significantly degrading the rendering quality.

This reduction of the number of significant layers, and the compact distribution of the pixels makes easier the I-LDI compression, and improves the rendering qualities. For the same bitrate, results show a gain of sometime 1dB in the quality of the synthesized virtual view, compared to classical LDI construction scheme.

This I-LDI construction scheme, as others detailed in Section 5.1, organizes layers by visibility: the first layer contains all pixels which are visible from the reference viewpoint, it is the classical 2D image; the other layers contain pixels in the camera scope, but hidden by objects in previous layers.

With this organization, each layer may contain pixels from the background and pixels from foreground objects in a same neighborhood, creating texture and depth discontinuities within the same layer. These discontinuities are blurred during layers compression when using a classical DCT-based scheme. This blurring of depth discontinuities, shown in Figure 6.12(a), significantly reduces the rendering quality obtained by classical rendering methods. For example, Figure 6.12(b) shows artifacts on objects boundaries, when rendered by the MPEG-VSRS rendering method [Tanimoto et al., 2008].

(a) Compressed depth map. "Ballet", view 4, MVC (QP=48).

(b) Synthesized virtual view. MPEG-VSRS, camera 3.

Figure 6.12: Impact of depth map compression on edge rendering.

The next section describes a novel object-based LDI representation, improving synthesized virtual views quality, in a rate-constrained context. In this so called O-LDI, pixels from each LDI layer are reorganized to enhance depth continuity.

6.3 Object-based LDI

In this section, we present a novel object-based LDI representation to address both compression and rendering issues. This object-based LDI is more tolerant to compression artifacts, and it is compatible with fast mesh-based rendering [Liu et al., 2000, Wang & Adelson, 1994].

For later comparison, Figure 6.13 presents an I-LDI with only two layers, where pixels are organized by visibility. As explained in Section 6.2, this visibility-based organisation leads to annoying artifacts. Visual quality gets even worse after a DCT-based compression of the LDI,

because of blurring effects on depth discontinuities.

(a) 1st layer. (b) 2nd layer.

Figure 6.13: Two first layers (color + depth map) of a visibility-based LDI.
Synthesized from "Ballet", views 4–3–5 at $t=0$, with incremental method (see Section 6.2).

Section 6.3.1 presents two methods for pixels classification into object-based layers. The first approach called global approach, is based on a depth threshold estimation. It gives a good compromise between accuracy and complexity. The second approach, based on a region growing algorithm, improves classification accuracy for complex scene geometry.

Section 6.3.2 explains how to use off-line inpainting methods to fill holes in the background layer and thus speed up the live rendering process.

Section 6.3.3 exposes compression results for both the visibility-based and object-based LDI representations.

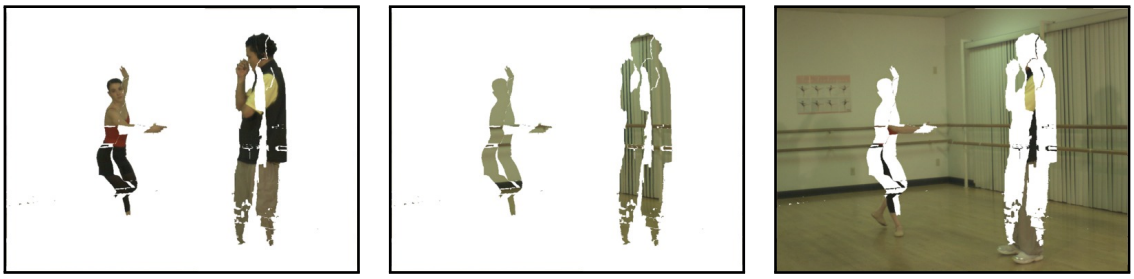
6.3.1 Object-based classification

In order to overcome artifacts which result from depth discontinuities, in particular after depth map compression, a novel object-based LDI representation is proposed. This representation organizes LDI pixels into two separate layers (foreground and background) to enhance depth continuity. If depth pixels from a real 3D object belong to the same layer, then compression is more efficient thanks to higher spatial correlation which improves effective spatial prediction of texture and depth map. Moreover, these continuous layers can be rendered efficiently (in terms of both speed and reduced artifacts) by using mesh-based rendering techniques.

The number of layers inside a LDI is not the same for each pixel position. Some positions may contain only one layer, whereas some other positions may contain many layers (or depth pixels). If several depth pixels are located at the same position, the closest belongs to the foreground, visible from the reference viewpoint, whereas the farthest is assumed to belong to the background. If there is only one pixel at a position, it is either a visible background pixel, or a foreground pixel in front of an unknown background.

All positions p containing several layers are selected from the input LDI. They define a region R, shown in Figure 6.14, where foreground and background pixels are easily identified. Z_p^{FG} denotes foreground depth, and Z_p^{BG} denotes background depth at position p. For each position q outside the region R, the pixel P_q has to be classified as a foreground or background pixel.

This section presents two methods to organize LDI's pixels into object-based layers. The first



(a) Foreground. (b) Background. (c) Unclassified.

Figure 6.14: Initializing state of the region growing algorithm.

method, described in Section 6.3.1, uses a global approach to estimate a depth threshold, then uses it to separate foreground objects from the background. The second method, described in Section 6.3.1, uses a region growing approach to ensure the depth continuity of each layer.

Classification with global threshold

A classical method to separate foreground from background is to use a threshold on the depth value. The challenge is to estimate the right threshold, according to the scene. The method described in this section uses already known background and foreground pixels to estimate a model of the depth distribution on each layer, and to evaluate the optimal threshold.

The already classified pixels in region R are used to estimate a model of depth distribution for foreground (FG) and background (BG) layers. These distributions are supposed to follow normal distributions, characterized by their means (μ_{FG} and μ_{BG}) and their variances (σ^2_{FG} and σ^2_{BG}). Foreground and background distributions estimated from the sequence "Ballet" are represented in Figure 6.15. The threshold λ which minimizes the classification error is calculated as the crossing point of the probability density functions respectively associated to the foreground and to the background:

$$\lambda \text{ such that } \begin{cases} \mu_{FG} < \lambda < \mu_{BG} \\ \frac{1}{\sigma_{FG}} \phi\left(\frac{\lambda - \mu_{FG}}{\sigma_{FG}}\right) = \frac{1}{\sigma_{BG}} \phi\left(\frac{\lambda - \mu_{BG}}{\sigma_{BG}}\right) \end{cases} \tag{6.1}$$

where $\phi(x)$ is the probability density function of the standard normal distribution $\mathcal{N}(0,1)$.

Finally, pixels which have not been classified yet are compared to this depth threshold λ. If their depth is lower than λ, they are considered to belong to the foreground. If their depth is higher than λ, they are considered to belong to the background.

The classification obtained by this global approach is presented in Figure 6.16 for the "ballet" data-set. One can observe than persons are well extracted from the background. Difficulties appear on the floor, whose pixels are sometimes classified as belonging to the background, and sometimes classified as belonging to the foreground. The next section introduces a region growing algorithm to help classifying large surfaces (like the floor) as belonging to the background.

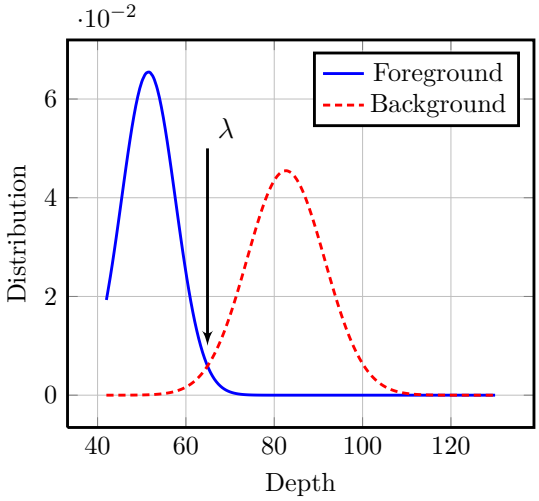
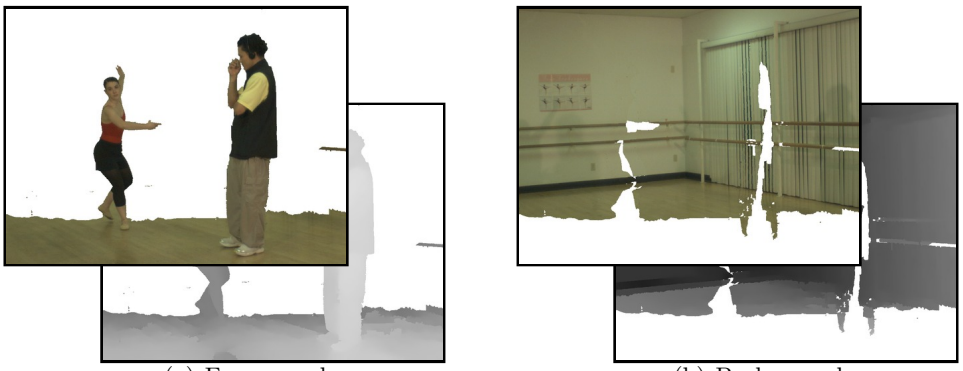


Figure 6.15: Depth distributions for the Ballet sequence modeled by Gaussian distributions. $FG \sim \mathcal{N}(51.5,\ 6.09^2)$, $BG \sim \mathcal{N}(82.6,\ 8.77^2)$ and $\lambda = 64.9$



(a) Foreground (b) Background

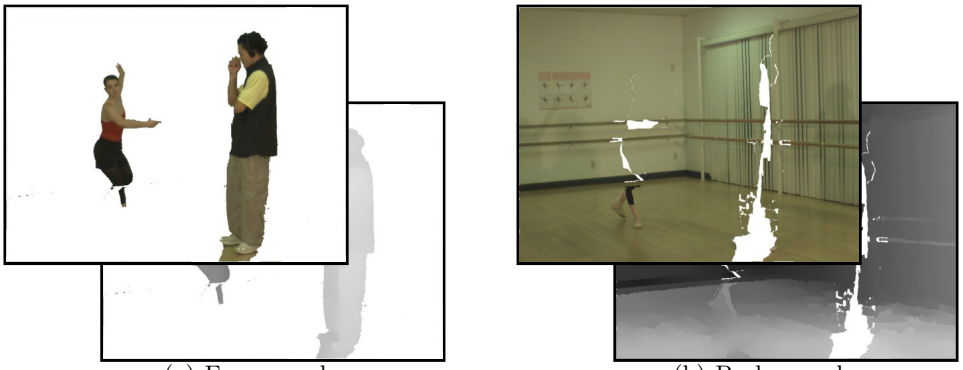
Figure 6.16: Final layers organization with the global threshold

Classification by region growing

The classification scheme proposed in Section 6.3.1 first estimates a global threshold, then uses it on the whole LDI. This global method is inadequate to classify a complex real scene for two reasons. First, some objects cannot be differentiated from the background, because both object and background are on the same side of the threshold. Then, some objects, such as the floor, can cross the threshold and thus be split in many layers. The second proposed method uses a region growing algorithm to ensure local depth continuity of each layer.

The classified region grows pixel by pixel, until the whole image is classified, as shown in Figure 6.17. For each couple of adjacent positions (p, q) around the border of region R such that p is inside R and q is outside R, the region R is expanded to q by classifying the pixel P_q according to its depth Z_q. For classification, Z_q is compared to background and foreground depths at position p. An extra depth value is then given to position q, so that q is associated with both a foreground and a background depth value.

$$P_q \in \begin{cases} \text{foreground} & \text{if } (Z_p^{BG} - Z_q) > (Z_q - Z_p^{FG}) \\ & \text{so } Z_q^{FG} = Z_q \text{ and } Z_q^{BG} = Z_p^{BG} \\ \text{background} & \text{if } (Z_p^{BG} - Z_q) < (Z_q - Z_p^{FG}) \\ & \text{so } Z_q^{FG} = Z_p^{FG} \text{ and } Z_q^{BG} = Z_q \end{cases}$$

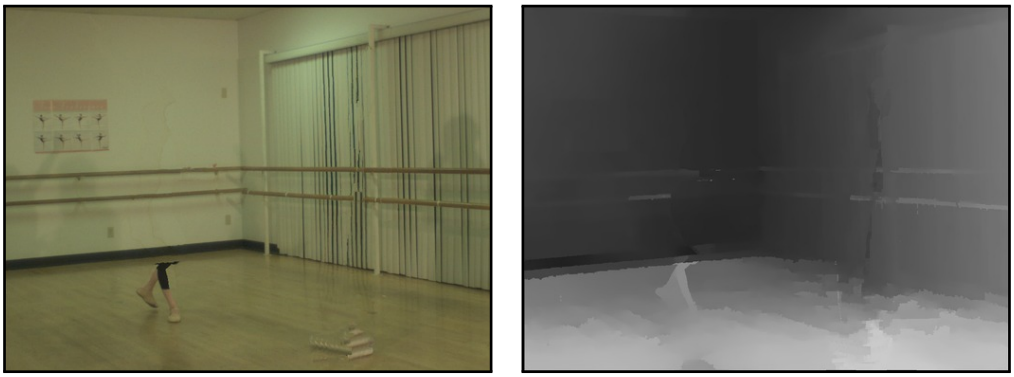


(a) Foreground (b) Background

Figure 6.17: Final layer organization with the region growing classification method.

6.3.2 Background filling by inpainting

Once the foreground/background classification is done, the background layer is most of the time not complete (see Figure 6.17(b)). Some areas of the background may not be visible from any input view. To reconstruct the corresponding missing background texture, one has to use inpainting algorithms on both texture and depth map images. The costly inpainting algorithm is processed once, during the LDI classification, and not during each view synthesis. Figure 6.18 shows the inpainted background with Criminisi's method [Criminisi et al., 2003].

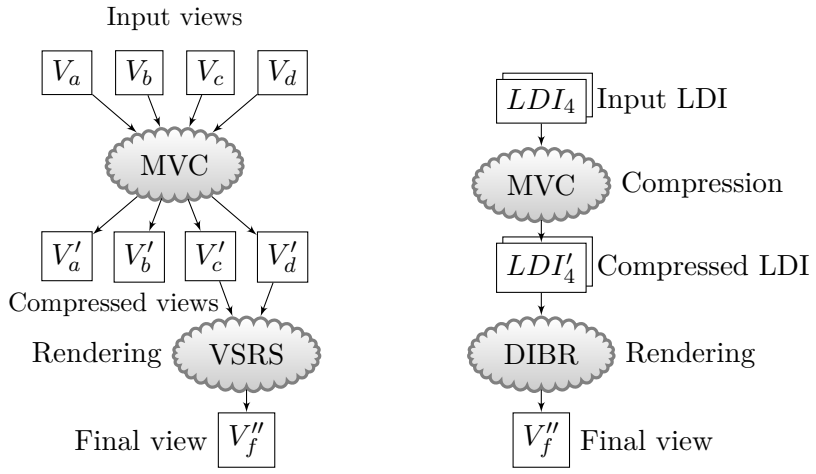


(a) Background texture. (b) Background depth map.

Figure 6.18: Background layer obtained after texture and depth map inpainting with Criminisi's method [Criminisi et al., 2003].

6.3.3 Results

The rendered quality of object-based LDI is compared with state-of-the-art MPEG compression techniques and classical LDI representation. Images are taken from both "Ballet" and "Breakdancing" data sets, provided by MSR [Zitnick et al., 2004].



(a) MVD compression (MVC) (b) LDI compression (MVC)

Figure 6.19: Block-diagrams of the two compared compression schemes. Figure 6.19(a) presents the classical multi-view compression scheme, using the MVC codec from MPEG [Tanimoto et al., 2008]. Figure 6.19(b) shows the proposed LDI compression scheme also using the MVC codec.

In the first place, the state-of-the-art method for multi-view video coding is used on a subset of acquired views, with various quantization parameters, varying from QP=20 to QP=50. Pair views (0, 2, 4, 6) are used for "Ballet" dataset, and odd views (1, 3, 5, 7) for "Breakdancing" dataset. Depth maps associated to these views are also coded with the MVC algorithm as a separate step. For each sequence, two views and associated depth map are decoded and used to synthesize intermediate views, using the MPEG/VSRS software [Tanimoto et al., 2008]. Views 4 and 6 of "Ballet" are used to synthesize view 5, and views 5 and 7 of "Breakdancing" are used to synthesize view 6. The corresponding block-diagram is presented in Figure 6.19(a)

In the second place, a subset of acquired views is used to synthesize the different types of LDI presented in this manuscript. Viewpoint 4 is chosen as the reference viewpoint, while views 2 and 6 for "Ballet", and 3 and 5 for "Breakancing", are chosen as additional views to construct the layers. The constructed LDI are restricted to three layers, while both I-LDI and O-LDI are restricted to two layers. All kind of LDI are compressed using the MVC algorithm on color maps and depth maps, as explained in Section 6.1. Several quantization parameters were used, from QP=18 to QP=54, producing compressed output data flows with bit-rates going from 1 Mbit/s to 40 Mbit/s. The same quantization parameter is used for texture compression and depth map compression, even if we explained in Section 2.2.3 that a better compromise can be founded. The compressed data flows is then used to synthesize virtual views, using the pixel-based projection method. The synthesized viewpoint corresponds to view 5 for "Ballet", and 6 for "Breakdancing". The corresponding block-diagram is presented in Figure 6.19(b)

Finally, the virtual views, synthesized either by VSRS on MVD, or by JPF on LDI, are

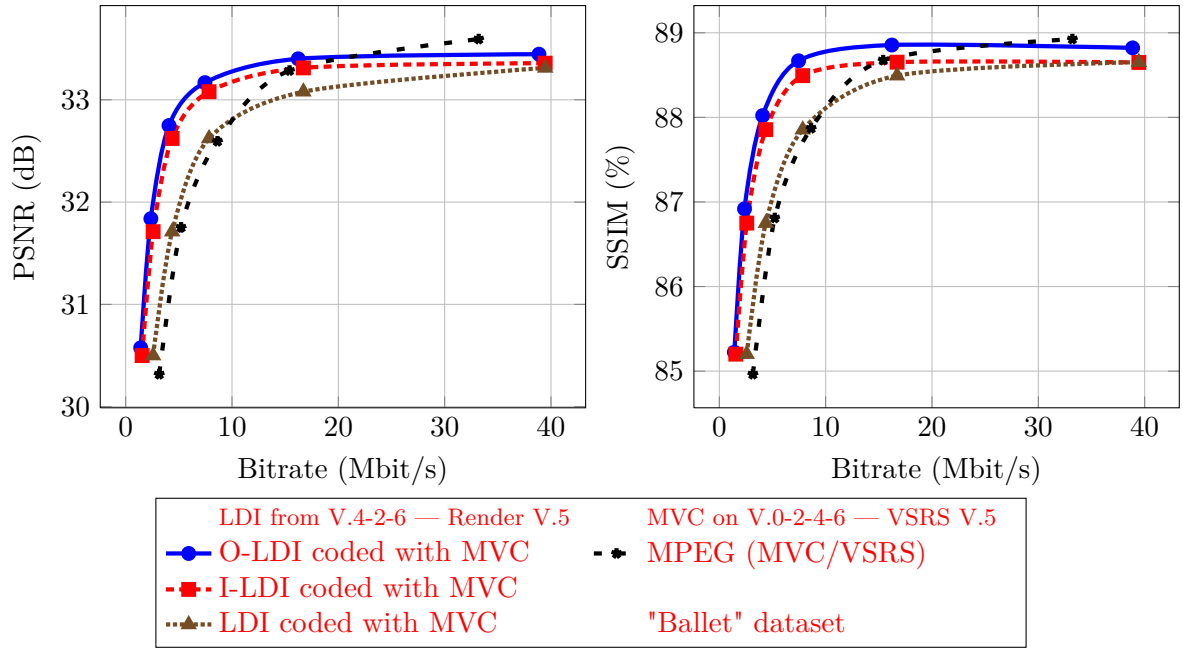


Figure 6.20: Rate distortion curves of "Ballet" multi-view video, firstly for LDI (object-based or not) compressed by MVC and rendered by our JPF method, and secondly for multi-view video compressed by MVC and rendered with VSRS algorithm.

compared to the corresponding original view, using both the PSNR and the SSIM comparison metrics. Figure 6.21 presents all the results as three rate/distortion curves. One can observe that for each quantization parameter, object-based LDI can be better compressed than classical LDI, resulting in a smaller bitrate. The visual rendering quality is also improved, resulting in a higher PSNR and SSIM for the same quantization parameter. Combining these two advantages, the rate/distortion curve for the object-based LDI is higher than the one for other LDI, for every bitrate. Compared to the rendering of Incremental LDI, result shows that the object-based layer reorganization enhances the rendered visual quality by about 0.7% (measured by SSIM) for the same transmission bit-rate, and reduce the required bit-rate by about 30% for the same visual quality.

6.3.4 Conclusion

This section has presented a novel object-based LDI and its benefits for compression and rendering. Two methods are proposed to construct this object-based LDI, based on a foreground and background classification. The first method uses a global threshold estimated from occluded regions. Its computational simplicity and ease of use make it a good first approach for scenes with relatively simple geometry. The second method is based on a region growing algorithm which ensures depth continuity of the layers. Its accuracy makes it a preferred method for complex and multi-objects scenes.

Whatever the method of construction, object-based LDI have some attractive features. The reduced number of depth discontinuities in each layer improves compression efficiency and minimizes compression artifacts for a given bitrate. The next section explains how this enhanced depth continuity in every layer can be exploited to simplify the virtual view synthesis algorithm,

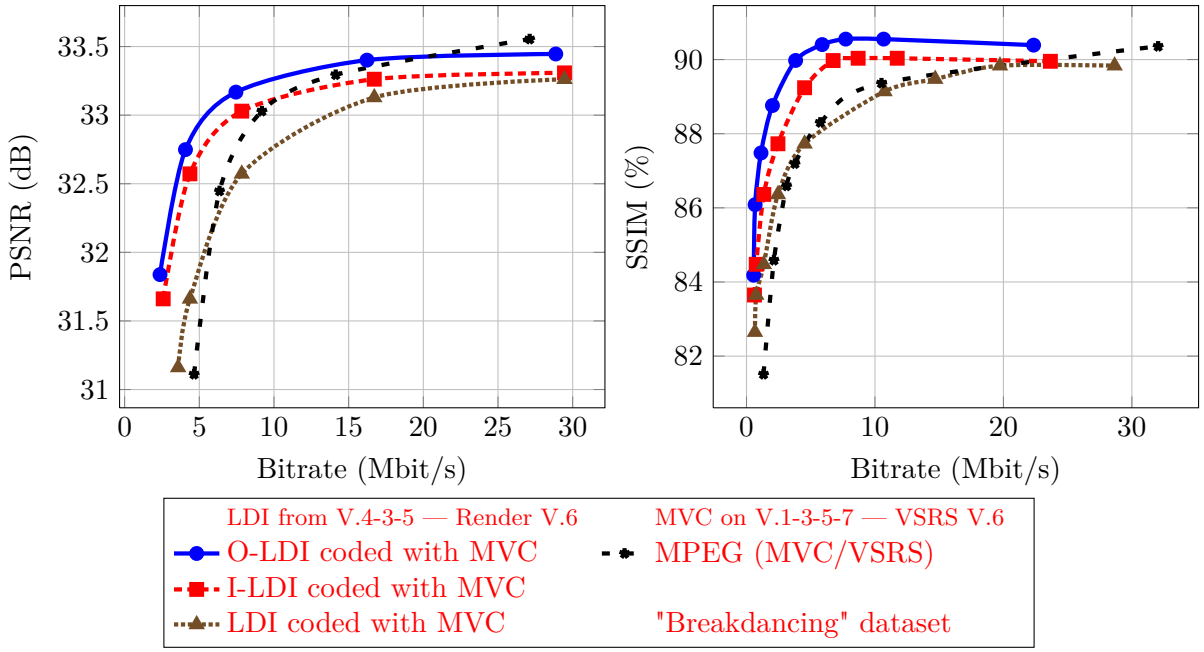


Figure 6.21: Rate distortion curves of "Breakdancing" multi-view video, firstly for LDI (object-based or not) compressed by MVC and rendered by our JPF method, and secondly for multi-view video compressed by MVC and rendered with VSRS algorithm.

and improve its rendering quality.

6.4 LDI Rendering

The Layered Depth Image data (LDI) are represented in a single image coordinate system. For this reason, many Depth Image Based Rendering methods (DIBR) can be adapted to use LDI as input data.

This section presents two rendering methods which have been implemented, focusing respectively on efficiency and quality. The first method uses a mesh-based representation to exploit GPU and parallelization, and thus allows a real time rendering for an eight-views auto-stereoscopic display. The second method uses the JPF method to project each layer onto the desired viewpoint, which removes artifacts like cracks, disocclusions, and ghosting artifacts.

6.4.1 Mesh-based LDI rendering

One of the fastest DIBR methods is to transform depth maps into 3D meshes, whose rendering is optimized by classical graphic cards [Kim, 2008, Sung-Yeol Kim, 2007]. The transformation of a single depth map is performed by selecting a set of depth pixels as vertices, whose 3D coordinates are computed thanks to their depth values. These vertices are then connected into a polygonal mesh, which is textured using the color map.

The mesh does not fit exactly the geometry given by the depth map, but it is a good approximation for flat surfaces. A common solution to support depth discontinuities is to increase the density of vertices around depth irregularities Unfortunately, this solution does not resolve disocclusions problems, and it requires to recompute the vertices and the mesh at each frame,

because the mesh evolves over time.

To avoid recomputing the mesh at each frame, one can use a fixed mesh where only vertices depth varies over time. Our GPU implementation uses the principle of "vertex shader", present on recent graphic cards, to parallelize the computation of the depth of each vertices. Vertices are uniformly distributed on the image as a regular square pattern.

In order to handle disocclusions, the mesh-based rendering method is adapted to use LDI as input data. Each layer is transformed into a mesh, all meshes are then simultaneously rendered. The object-based representation of the LDI, described in Section 6.3, reduces the number of depth discontinuities in each layer, and thus reduces stretching artifacts during mesh rendering.

Some layers of a LDI are not complete, and some depth pixels are missing from the representation. These missing depth pixels should be filled in before the layer is transformed into a mesh, to ensure that every vertex is well defined in three dimensions. The color of these missing depth pixels is simply defined as transparent. The depth of these missing depth pixels is computed by an inpainting method, in order to avoid depth discontinuities with their neighborhood. Here, we use the diffusion-based inpainting method described in [Telea, 2004], to process the depth map of each layer.

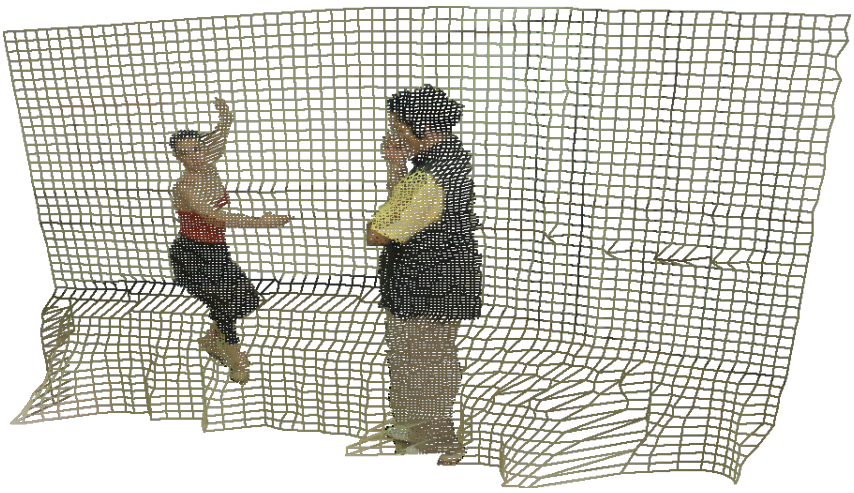


Figure 6.22: Fast 3D rendering of a high detailed foreground mesh, onto a low detailed background mesh, both displayed as grid.

Figure 6.22 shows the two meshes of an object-based LDI with two layers (see Section 6.3). Our first experiments, with this method, have shown the feasibility of real time rendering for an eight-views auto-stereoscopic display. Small details on the depth map are not well rendered by a mesh with large polygons. Reducing the size of the polygons will improve the rendering quality, but also increases the number of polygons, and thus the rendering complexity. Another view synthesis method is proposed in the next section, which provides high quality rendering.

6.4.2 JPF-based view synthesis

The second method improves the visual quality of synthesized views by using the Joint Projection Filling (JPF) method, presented in Part I. The JPF method is based on McMillan's occlusion-compatible ordering introduced in [McMillan, 1995]. It allows cracks handling and disocclusions filling by ensuring that only background pixels are used during interpolation.

Figure 6.23 presents rendering results for both classical and object-based LDI. One can observe that in the case of visibility-based LDI, dissoclusion areas are filled at the rendering stage using a fast inpainting method which results in annoying artifacts. These artifacts are avoided by computing the full background texture once, during the object-based LDI construction, and not during each view synthesis process.

(a) Missing background. (b) Visibility-based LDI. (c) Object-based LDI.

Figure 6.23: Rendering comparison between visibility-based LDI and object-based LDI.

6.5 Conclusion

This chapter has presented novel methods for reducing the size of Layered Depth Image (LDI), based on state-of-the-art video compression techniques, advanced LDI construction algorithms, and improved virtual view rendering methods.

The compression of LDI sequences is performed by the MVC coder, which exploits both temporal redundancies and inter-layer redundancies. This methods offers a backward compatibility with current 2D decoders and 2D displays. Results shows a quality saturation, due to projection errors and contours misalignments.

The LDI generation is then improved, using an incremental procedure in order to reduce the completion rate of additional layers by 80%. The minimum information to fill disocclusion areas is inserted into LDI layers. Results show that only a few layers contain information, and the others may be ignored without significantly degrading the rendering quality.

An object-based pixel reorganization is also proposed, which ensures spatial consistency of each layer, and thus improves compression efficiency. The extracted background is inpainted before the LDI compression, what avoids the use of a rendering-side inpainting, allowing real time rendering for an eight-views auto-stereoscopic display. Results show that the object-based layer reorganization reduces the required bit-rate by about 30% for the same visual quality.

Chapter 7

Conclusion and perspectives

In sight of the huge amount of data concerned by 3D video communication services, which increases with the number of camera channels, compression efficiency is of paramount importance. To this end, this manuscript is devoted to efficiently encode 3D video data, by using a compact intermediate representation.

3D video coders usually fall in two categories: multiview-based and 3D-model-based approaches. Multiview-based coding relies on scenarios where a 3D scene is captured by several cameras (MVV camera system), therefore exploiting pixels correlations between adjacent cameras. 3D-model-based video coding exploits the a priori 3D model knowledge of the scene which is transmitted together with texture and animation parameters. An intermediate technique for multi-view video coding is developed in this manuscript, in which the scene is represented in form of Layered Depth Images sequence (LDI). The LDI representation is helpful to synthesize virtual viewpoints as it contains additional information relative to the scene geometry and the occluded textures.

The sequence of LDI is first constructed from a multi-view plus depth video sequence, by recovering the disoccluded texture from sides views. This conversion from MVD to LDI reduces the video data size as the spatial redundancy among views is removed.

Below, we summarize the contributions of the thesis work in virtual view synthesis, LDI construction, and LDI compression. Some directions for future research are finally proposed to end the manuscript.

7.1 Summary of contributions

In this manuscript, several contributions have been proposed to develop an advanced framework for representation and processing of multi-view plus depth video data, using the concept of Layered Depth Image (LDI). These contributions are classified in two categories, as they relate to the synthesis of virtual view, or to the processing of the intermediate LDI representation.

7.1.1 Virtual view synthesis

This manuscript addresses the disocclusion problem which may occur when using Depth-Image-Based Rendering (DIBR) techniques in 3DTV and Free-Viewpoint TV applications. In addition

to the LDI representation itself, very helpful to fill disocclusions with real textures, we propose to handle disocclusions with new DIBR techniques, adapted either for intermediate view interpolation or virtual view extrapolation. These advanced view synthesis methods allow us to reduce the number of LDI layers without introducing visual artifacts in rendered views.

The techniques described in the first part improve upon state-of-the-art DIBR methods by introducing the following key contributions:

- A Joint Projection Filling method (JPF), which is a novel forward projection technique for DIBR which detects cracks and disocclusions, for which the unknown depth values are estimated while performing the warping.
- A ghost removal method, relying on a depth-based pixel confidence measure, which avoids ghosting artifacts in the rendered views.
- A full-Z depth-aided inpainting method which takes into account all information given by the depth map to fill in disocclusions with textures at the correct depth.
- A texture registration method which handles inaccuracies of both cameras calibration and depth map estimation by correcting texture misalignments.

Joint Projection Filling method

The Joint Projection Filling (JPF) method performs pixels warping for virtual depth map synthesis while making use of an occlusion-compatible pixel ordering strategy to detect cracks and disocclusions, and to select the pixels to be propagated in the occlusion areas filling process.

This method allows us to handle both pixels warping and disocclusion filling simultaneously, by ensuring that only background pixels are used during interpolation. Small cracks and large disocclusions are handled gracefully, with similar computational cost as simple forward projection, avoiding the use of the filtering step as done in the classical approach.

Filling disocclusion areas during the projection ensures that geometrical structures are well preserved. This JPF method is thus particularly suited for virtual depth map synthesis. Synthesized depth maps are very similar to ground truth depth maps, even for complex scene geometry, where other disocclusions filling methods produce artifacts.

Pixel confidence measure for ghosting removal

Pixels along object boundaries contain mixed foreground/background information for texture and depth value, and are thus considered unreliable.

In order to avoid ghosting artifacts which are visible when these pixels are projected, we propose to associate a confidence measure to each pixel. This confidence measure is low for background pixels near boundaries, which allows the JPF method to warp these pixels like the corresponding boundaries, and then use only relevant pixels to fill in the disocclusion.

The confidence-based pixels shifting avoids ghosting artifacts and reduces texture stretching while sharpening discontinuities.

Full-Z depth-aided inpainting

Most inpainting techniques use neighboring pixels solely based upon colorimetric distance, while a disocclusion hole should be filled in with background pixels, rather than foreground ones.

In order to fill disocclusion with textures with the correct depth, we propose a full-Z depth-aided inpainting method which takes into account all information given by the depth map to fill in disocclusions.

The full-Z depth-aided inpainting method fills in disocclusions with textures at the correct depth, preserving structured textures and object boundaries.

Texture registration with texture misalignments handling

In order to handle inaccuracies of both cameras calibration and depth map estimation, we propose a texture registration method, similar to the Floating Texture approach, which helps reducing texture misalignments.

Blending intermediate views together, weighted by the confidence score, allows us to fill in disocclusions with real captured texture. Blurring artifacts may appear if intermediate views are misaligned. Applying a texture registration removes blur on contours and texture details are thus enhanced.

Results have shown that texture misalignments in virtual views can be reduced and that consistency between multiple synthesized views is improved.

Conclusion

The association of these four methods allowed us to develop two DIBR techniques.

The first DIBR technique is a virtual view extrapolation method from a single input view plus depth sequence. The JPF method is used here to synthesize the depth map of the virtual view with minimal errors, in order to conduct the full-Z depth-aided inpainting. The depth-aided inpainting takes into account the high quality of the generated depth map to select correct patches to be duplicated.

The second DIBR technique is a virtual view interpolation method from multiple input video plus depth sequences. The JPF method is used here to produce intermediate views without cracks nor disocclusions, in order to process optical flow estimation required by texture registration.

7.1.2 LDI processing

This manuscript also addresses the compactness issue of multi-view videos. LDI are compact representations which remove redundancies in the data. Efficient compression methods can thus be defined, associated with efficient and high-quality virtual viewpoints rendering algorithm.

The methods proposed in the second part improve upon state-of-the-art LDI processing, by introducing the following key contributions:

- A layer compression scheme, using the MVC coder to exploit both temporal redundancies and inter-layer redundancies.
- An incremental construction scheme (I-LDI) which reduces the completion rate of additional layers.
- An object-based LDI representation (O-LDI), improving synthesized virtual views quality, in a rate-constrained context.

LDI compression

The compression of LDI sequences is performed by the MVC/AVC method. The MVC/AVC coder exploits both temporal redundancies and inter-layer redundancies.

The proposed method uses the first layer as base view in order to provide backward compatibility. The LDI stream thus can be decoded by a classical H.264/AVC decoder and can be rendered in 2D, on a classical 2D device.

The obtained results show that the LDI format reaches quality saturation, mainly due to projection errors and contours misalignments. The LDI representation is thus more suitable for low bitrate, where it provides a significant gain in quality compared to conventional multi-view video coding with MVC method.

Incremental LDI

The proposed method to generate LDI from natural multiple images plus depth uses an incremental procedure.

The minimum information required to fill disocclusion areas is inserted into LDI layers which makes layers easier to compress. Layers containing only few scattered pixels are eliminated to reduce data size. These eliminated pixels are resynthesized at rendering stage by the disocclusions filling method. In practice, only two layers can be kept, without significantly degrading the rendering quality. This representation thus does not depend on the number of input views.

The proposed construction scheme reduces by 80% the number of pixels in additional layers, compared to a classical LDI construction method.

Object-based LDI

We propose to modify the data structure of layers in order to group pixels by semantic object. This pixels reorganisation improves the LDI compression efficiency and also speeds up the virtual view rendering.

The proposed method to construct these object-based LDI is a foreground and background classification, based on a region growing algorithm which ensures depth continuity of the layers. The reduced number of depth discontinuities in each layer improves spacial consistency, and thus compression efficiency. The extracted background is inpainted before LDI compression, what allows a fast mesh-based rendering, without client-side inpainting. The mesh-based LDI rendering is GPU optimised, which allows real time rendering for an eight-views auto-stereoscopic display.

Compared to the rendering of incremental LDI, results show that the object-based layer reorganisation enhances the rendered visual quality by about 0.7% (measured by SSIM) for the same transmission bitrate, and reduces the required bitrate by about 30% for the same visual quality.

From the experimental results, the proposed LDI representation shows better performance than the multi-view coding method in terms of data size, mainly for low bitrate. We therefore believe that the proposed approach could be helpful to represent, and process multi-view video data effectively.

7.2 Perspectives

In this thesis work, we have studied and proposed several contributions to the development of an advanced framework for multi-view plus depth videos processing, using LDI as an intermediate representation, facilitating compression and rendering. Nevertheless, a number of topics can be identified that still require further investigation, and may lead to even better compression performance for the 3D class of video coding algorithms.

These topics concern LDI construction, compression, view synthesis and evaluation methods.

Floating Texture for LDI construction

We have seen that floating texture helps reducing texture misalignments and geometry errors in intermediate view rendering techniques. Therefore, the floating texture technique could also be used as a feedback loop during the incremental construction of the LDI representation. When the residual information is extracted from side views and inserted into the I-LDI, it should be aligned with texture already in the I-LDI to ensure the continuity of visual structures.

Depth map dedicated compression method

In this work, we use the MVC coder, which is a DCT-based compression scheme, to compress both textures and depth maps sequences.

Traditional image compression methods have been designed to provide maximum perceived visual quality, and a direct application is suboptimal for depth map compression, since depth maps are not directly viewed. The depth map boundaries, very important for virtual view synthesis, are also badly coded by DCT-based compression scheme.

Some advanced image coding, dedicated to depth map, should thus be considered.

Temporal coherent inpainting method

The proposed full-Z depth-aided inpainting method processes one frame at a time. It could introduce temporal flickering, or inter-view inconsistency, when synthesising multiple view for an auto-multi-scopic display.

Processing the background inpainting before the compression of object-based LDI reduces inter-view inconsistencies, but has no impact on temporal flickering.

An advanced inpainting method, which is time consistent and able to process multiple synthesized views at once, may be useful to ensure temporal consistency.

3D content distortion measure

We have already mentioned that the distortion measures used in this work were designed for 2D perception. The correlation coefficients of these metrics, compared to perceptual evaluation, is below 40%. One thus has to be extremely careful with the range of validity of these metrics.

Not taking the 3D visual system into account is probably one of the serious drawbacks of the above mentioned measures. 3D perception quality can be regarded as one of the most important issues to be taken into consideration. An efficiency improvement of the 3D video codec can be

proposed by studying a distortion model to characterize the view synthesis quality based on subjective quality.

Bibliography

[Adelson & Bergen, 1991] Adelson, E. H. & Bergen, J. R. (1991). The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing* (pp. 3–20). 36

[Akar et al., 2007] Akar, G., Tekalp, A., Fehn, C., & Civanlar, M. (2007). Transport methods in 3dtv - a survey. *Circuits and Systems for Video Technology, IEEE Transactions*, 17(11), 1622–1630. 29

[Alatan et al., 2007] Alatan, A., Yemez, Y., Gudukbay, U., Zabulis, X., Müller, K., Erdem, C., Weigel, C., & Smolic, A. (2007). Scene representation technologies for 3dtv - a survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1587–1605. 35

[Alliez & Desbrun, 2001] Alliez, P. & Desbrun, M. (2001). Progressive compression for lossless transmission of triangle meshes. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 195–202). New York, NY, USA: ACM. 41

[Ashikhmin, 2001] Ashikhmin, M. (2001). Synthesizing natural textures. In *Interactive 3D graphics (I3D), Proceedings of* (pp. 217–226). New York, NY, USA: ACM. 66

[Balter, 2005] Balter, R. (2005). *Construction d'un modèle 3D évolutifs et scalables pour le codage vidéo*. PhD thesis, IRISA TEMICS - France Télécom R&D/TECH/IRIS. 41

[Barenbrug, 2009] Barenbrug, B. (2009). Declipse 2: multi-layer image and depth with transparency made practical. *Stereoscopic Displays and Applications*, 7237(1), 72371G. 39

[Barnes et al., 2009] Barnes, C., Shechtman, E., Finkelstein, A., & Goldman, D. B. (2009). Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28, 1–11. 65

[Barnes et al., 2010] Barnes, C., Shechtman, E., Goldman, D. B., & Finkelstein, A. (2010). The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision (ECCV): Part III, Proceedings of* (pp. 29–43). Berlin, Heidelberg: Springer-Verlag. 65

[Benzie et al., 2007] Benzie, P., Watson, J., Surman, P., Rakkolainen, I., Hopf, K., Urey, H., Sainov, V., & von Kopylow, C. (2007). A survey of 3dtv displays: Techniques and technologies. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1647–1658. 29, 30

[Bertalmío et al., 2001] Bertalmío, M., Bertozzi, A., & Sapiro, G. (2001). Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition*

(CVPR), volume 1 (pp. 355–362). Los Alamitos, CA, USA: IEEE Computer Society. 65, 68, 70, 80, 81, 82, 83

[Bertalmío et al., 2000] Bertalmío, M., Sapiro, G., Caselles, V., & Ballester, C. (2000). Image inpainting. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 417–424). New York, NY, USA: ACM Press. 30, 65, 68

[Blinn & Newell, 1976] Blinn, J. F. & Newell, M. E. (1976). Texture and reflection in computer generated images. *Communications of the ACM*, 19, 542–547. 36

[Bornard et al., 2002] Bornard, R., Lecan, E., Laborelli, L., & Chenot, J.-H. (2002). Missing data correction in still images and image sequences. In *ACM international conference on Multimedia, Proceedings of* (pp. 355–361). New York, NY, USA: ACM. 65

[Bosc et al., 2010] Bosc, E., Jantet, V., Morin, L., Pressigout, M., & Guillemot, C. (2010). Vidéo 3d: quel débit pour la profondeur ? In *COmpression et REpresentation des Signaux Audiovisuels (CORESA)* (pp. 1–4). Lyon, France. 51

[Bosc et al., 2011a] Bosc, E., Jantet, V., Pressigout, M., Morin, L., & Guillemot, C. (2011a). Bit-rate allocation for multi-view video plus depth data. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video* (pp. 1–4). Antalya, Turkey. 51

[Bosc et al., 2011b] Bosc, E., Köppel, M., Pépion, R., Pressigout, M., Morin, L., Ndjiki-Nya, P., & Lecallet, P. (2011b). Can 3d synthesized views be reliably assessed through usual subjective and objective evaluation protocols? In *International Conference on Image Processing (ICIP)*. 45

[Bruls et al., 2007] Bruls, W., Varekamp, C., Gunnewiek, R., Barenbrug, B., & Bourge, A. (2007). Enabling introduction of stereoscopic (3d) video: Formats and compression standards. *IEEE International Conference on Image Processing (ICIP)*, 1, 89–92. 35

[Buehler et al., 2001] Buehler, C., Bosse, M., McMillan, L., Gortler, S., & Cohen, M. (2001). Unstructured lumigraph rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 425–432). New York, NY, USA: ACM. 48, 88

[Chan et al., 2007] Chan, S., Shum, H.-Y., & Ng, K.-T. (2007). Image-based rendering and synthesis. *Signal Processing Magazine, IEEE*, 24(6), 22–33. 56

[Cheng et al., 2007a] Cheng, X., Sun, L., & Yang, S. (2007a). Generation of layered depth images from multi-view video. *IEEE International Conference on Image Processing (ICIP)*, 5, 225–228. 10, 11, 39, 91

[Cheng et al., 2007b] Cheng, X., Sun, L., & Yang, S. (2007b). A multi-view video coding approach using layered depth image. In *MultiMedia Signal Processing (MMSP), IEEE 9th Workshop on* (pp. 143–146). 93, 98

[Chuang et al., 2002] Chuang, Y.-Y., Agarwala, A., Curless, B., Salesin, D. H., & Szeliski, R. (2002). Video matting of complex scenes. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 243–248). New York, NY, USA: ACM. 64

[Cohen-Steiner et al., 2004] Cohen-Steiner, D., Alliez, P., & Desbrun, M. (2004). Variational shape approximation. *ACM Transactions on Graphics (TOG)*, 23(3), 905–914. 41

[Colleu, 2010] Colleu, T. (2010). *A floating polygon soup representation for 3D video.* PhD thesis, IRISA TEMICS - France Télécom R&D/TECH/IRIS. 35

[Collins, 1996] Collins, R. T. (1996). A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 358–363). Washington, DC, USA: IEEE Computer Society. 91

[Criminisi et al., 2003] Criminisi, A., Pérez, P., & Toyama, K. (2003). Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2 (pp. 721–728).: IEEE Computer Society. 19, 65, 66, 68, 79, 109

[Culbertson et al., 1999] Culbertson, W. B., Malzbender, T., & Slabaugh, G. G. (1999). Generalized voxel coloring. In *International Conference on Computer Vision (ICCV)* (pp. 100–115). London, UK: Springer-Verlag. 41

[Daribo, 2009] Daribo, I. (2009). *Coding and rendering of 3D video sequences; and applications to Three-Dimensional Television (3DTV) and Free Viewpoint Television (FTV).* PhD thesis, Graduate College of Telecom ParisTech, Paris, France. 35

[Daribo & Pesquet, 2010] Daribo, I. & Pesquet, B. (2010). Depth-aided image inpainting for novel view synthesis. In *Multimedia Signal Processing (MMSP), IEEE International Workshop on* (pp. 167–170). 19, 20, 68, 69, 70, 78, 80, 81

[Debevec et al., 1998] Debevec, P., Yu, Y., & Boshokov, G. (1998). *Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping.* Technical Report UCB/CSD-98-1003, EECS Department, University of California, Berkeley. 36, 56

[Debevec et al., 1996] Debevec, P. E., Taylor, C. J., & Malik, J. (1996). Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 11–20). New York, NY, USA: ACM. 36, 41

[Décoret et al., 2003] Décoret, X., Durand, F., Sillion, F., & Dorsey, J. (2003). Billboard clouds for extreme model simplification. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of*: ACM Press. 40

[Do et al., 2009] Do, L., Zinger, S., Morvan, Y., & de With, P. H. N. (2009). Quality improving techniques in dibr for free-viewpoint video. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video* (pp. 1–4). 63, 67

[Dodgson, 2005] Dodgson, N. A. (2005). Autostereoscopic 3d displays. *Computer*, 38, 31–36. 31

[Drori et al., 2003] Drori, I., Cohen-Or, D., & Yeshurun, H. (2003). Fragment-based image completion. *ACM Trans. Graph.*, 22, 303–312. 65

[Duan & Li, 2003] Duan, J. & Li, J. (2003). Compression of the layered depth image. *Image Processing, IEEE Transactions on*, 12(3), 365–372. 10, 93, 94

[Efros & Leung, 1999] Efros, A. A. & Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision*, volume 2 (pp. 1033). Washington, DC, USA: IEEE Computer Society. 65, 66

[Eisemann et al., 2008] Eisemann, M., De Decker, B., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C., & Sellent, A. (2008). Floating textures. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2), 409–418. Received the Best Student Paper Award at Eurographics 2008. 84, 85

[Fan et al., 2004] Fan, L., Jiang, G., Yu, M., Choi, T.-Y., & Kim, Y.-D. (2004). Ray space interpolation based on its inherent characteristics. *TENCON, IEEE Region 10 Conference*, 1, 375–378. 36

[Favalora, 2005] Favalora, G. E. (2005). Volumetric 3d displays and application infrastructure. *IEEE Computer Society Press*, 38, 37–44. 31

[Fecker et al., 2008] Fecker, U., Barkowsky, M., Kaup, A., & Member, S. (2008). Histogram-based prefiltering for luminance and chrominance compensation of multiview video. *Circuits and Systems for Video Technology, IEEE Transaction on*, 18. 29

[Fehn et al., 2002] Fehn, C., Kauff, P., Op De Beeck, M., Ernst, F., Ijsselsteijn, W., Van Gool, L., Ofek, E., & Sexton, I. (2002). An evolutionary and optimised approach on 3d-tv. In *In Proceedings of International Broadcast Conference* (pp. 357–365). 27

[Fitzgibbon et al., 2005] Fitzgibbon, A., Wexler, Y., & Zisserman, A. (2005). Image-based rendering using image-based priors. *Int. J. Comput. Vision*, 63(2), 141–151. 56

[Flierl et al., 2007] Flierl, M., Mavlankar, A., & Girod, B. (2007). Motion and disparity compensated coding for multiview video. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1474–1484. 49, 53

[Furihata et al., 2010] Furihata, H., Yendo, T., Tehrani, M. P., Fujii, T., & Tanimoto, M. (2010). Novel view synthesis with residual error feedback for ftv. *SPIE Digital Library*, 1, 7524. 56

[Furuya et al., 2007] Furuya, H., Kitahara, I., Kameda, Y., & Ohta, Y. (2007). Viewpoint-dependent quality control on microfacet billboarding model for sports video. In *Multimedia and Expo, IEEE International Conference on* (pp. 1199–1202). 37

[Fusiello et al., 2000] Fusiello, A., Trucco, E., & Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12, 16–22. 29

[Gautier et al., 2011] Gautier, J., Le Meur, O., & Guillemot, C. (2011). Depth-based image completion for view synthesis. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*. 68, 80, 82

[Goldlucke & Magnor, 2003] Goldlucke, B. & Magnor, M. (2003). Real-time microfacet billboarding for free-viewpoint video rendering. *IEEE International Conference on Image Processing (ICIP)*, 3, 713–728. 37

[Golovinskiy et al., 2009] Golovinskiy, A., Podolak, J., & Funkhouser, T. (2009). Symmetry-aware mesh processing. In *Proceedings of the 13th IMA International Conference on Mathematics of Surfaces XIII* (pp. 170–188). Berlin, Heidelberg: Springer-Verlag. 41

[Gortler et al., 1996] Gortler, S. J., Grzeszczuk, R., Szeliski, R., & Cohen, M. F. (1996). The lumigraph. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 43–54). New York, NY, USA: ACM. 37

[Gortler et al., 1997] Gortler, S. J., wei He, L., & Cohen, M. F. (1997). *Rendering Layered Depth images*. Technical report, Microsoft Research. 39

[Han et al., 2007] Han, S.-R., Yamasaki, T., & Aizawa, K. (2007). Time-varying mesh compression using an extended block matching algorithm. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1506–1518. 41

[Harrison, 2001] Harrison, P. (2001). A non-hierarchical procedure for re-synthesis of complex textures. In *WSCG Conference proceedings* (pp. 190–197). 65

[Hartley & Zisserman, 2004] Hartley, R. I. & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition. 32, 56, 57, 91

[Hasinoff et al., 2006] Hasinoff, S. W., Kang, S. B., & Szeliski, R. (2006). Boundary matting for view synthesis. *Computer Vision and Image Understanding*, 103, 22–32. 64

[He et al., 2007] He, Y., Ostermann, J., Tanimoto, M., & Smolic, A. (2007). Introduction to the special section on multiview video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1433–1435. 49

[Horn & Schunck, 1981] Horn, B. K. P. & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence*, 17, 185–203. 84

[ISO/IEC, 2007] ISO/IEC (2007). Representation of auxiliary video and supplemental information. In *FDIS 23002-3*. JTC1/SC29/WG11. 39

[Jia & Tang, 2003] Jia, J. & Tang, C.-K. (2003). Image repairing: robust image synthesis by adaptive nd tensor voting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1 (pp. 643–650). Washington, DC, USA: IEEE Computer Society. 65

[Kauff et al., 2007] Kauff, P., Atzpadin, N., Fehn, C., Müller, K., Schreer, O., Smolic, A., & Tanger, R. (2007). Depth map creation and image-based rendering for advanced 3dtv services

providing interoperability and scalability. *Signal Processing: Image Communication*, 22, 217–234. 73

[Khodakovsky et al., 2000] Khodakovsky, A., Schröder, P., & Sweldens, W. (2000). Progressive geometry compression. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 271–278). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. 41

[Kim & Sikora, 2007] Kim, J. & Sikora, T. (2007). Confocal disparity estimation and recovery of pinhole image for real-aperture stereo camera systems. *IEEE International Conference on Image Processing (ICIP)*, 5, 229–232. 91

[Kim, 2008] Kim, S.-Y. (2008). *Depth Map Creation and Mesh-based Hierarchical 3-D Scene Representation in Hybrid Camera System*. PhD thesis, Gwangju Institute of Science and Technology. 112

[Kitahara et al., 2007] Kitahara, M., Kimata, H., Shimizu, S., Kamikura, K., & Yashima, Y. (2007). Progressive coding of surface light fields for efficient image based rendering. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1549–1557. 37

[Krutz et al., 2006] Krutz, A., Frater, M., & Sikora, T. (2006). Improved image registration using the up-sampled domain. *Multimedia Signal Processing (MMSP), IEEE International Workshop on*, (pp. 447–450). 84

[Kubota et al., 2007] Kubota, A., Smolic, A., Magnor, M., Tanimoto, M., Chen, T., & Overview, I. (2007). Multi-view imaging and 3dtv (special issue overview and introduction). *IEEE Signal Processing Magazine, Special Issue on*, 24(6), 1–12. 27

[Kurutepe et al., 2006] Kurutepe, E., Civanlar, M. R., & Tekalp, A. M. (2006). Interactive multi-view video delivery with view-point tracking and fast stream switching. In *Multimedia Content Representation, Classification and Security*, volume 4105/2006: Springer Berlin / Heidelberg. 49

[Kurutepe et al., 2007] Kurutepe, E., Civanlar, M. R., & Tekalp, A. M. (2007). Client-driven selective streaming of multiview video for interactive 3dtv. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1558–1565. 49

[Kutulakos & Seitz, 1998] Kutulakos, K. N. & Seitz, S. M. (1998). A theory of shape by space carving. *Computer Vision, International Journal of*, 38, 307–314. 41, 91

[Lavoué et al., 2008] Lavoué, G., Dupont, F., & Baskurt, A. (2008). *3D Object Processing: Compression, Indexing and Watermarking*, chapter 3D Compression. John Wiley & Sons. 41, 43

[Le Gall, 1992] Le Gall, D. J. (1992). The mpeg video compression algorithm. *Signal Processing: Image Communication*, 4(2), 129 – 140. 29

[Lee et al., 2002] Lee, H., Alliez, P., & Desbrun, M. (2002). Angle-analyzer: A triangle-quad mesh codec. In *Computer Graphics Forum, Proceedings of Eurographics*. 41

[Levoy & Hanrahan, 1996] Levoy, M. & Hanrahan, P. (1996). Light field rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 31–42). New York, NY, USA: ACM. 37

[Levoy & Whitted, 1985] Levoy, M. & Whitted, T. (1985). *The Use of Points as a Display Primitive.* Computer science department, University of North Carolina at Chapel Hill. 41

[Liu et al., 2000] Liu, J., Przewozny, D., & Pastoor, S. (2000). Layered representation of scenes based on multiview image analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 10(4), 518–529. 105

[Maitre & Do, 2009] Maitre, M. & Do, M. N. (2009). Shape-adaptive wavelet encoding of depth maps. In *Picture Coding Symposium, Proceedings of*, PCS'09 (pp. 473–476). Piscataway, NJ, USA: IEEE Press. 29

[Manning & Dyer, 1996] Manning, R. A. & Dyer, C. R. (1996). Dynamic view morphing. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 21–30). 84

[Matusik & Pfister, 2004] Matusik, W. & Pfister, H. (2004). 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics (TOG)*, 23, 814–824. 48

[McMillan, 1995] McMillan, L. (1995). *A List-Priority Rendering Algorithm for Redisplaying Projected Surfaces.* Technical Report 95-005, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. 17, 32, 73, 85, 113

[Merkle et al., 2009] Merkle, P., Morvan, Y., Smolic, A., Farin, D., Müller, K., de With, P. H. N., & Wiegand, T. (2009). The effects of multiview depth video compression on multiview rendering. *Image Communication*, 24, 73–88. 29

[Merkle et al., 2007] Merkle, P., Smolic, A., Müller, K., & Wiegand, T. (2007). Efficient prediction structures for multiview video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1461–1473. 10, 16, 49, 50, 51, 52, 88

[Mori et al., 2009] Mori, Y., Fukushima, N., Yendo, T., Fujii, T., & Tanimoto, M. (2009). View generation with 3d warping using depth information for ftv. *Image Commun.*, 24, 65–72. 63

[Morvan, 2009] Morvan, Y. (2009). *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video.* PhD thesis, Eindhoven University of Technology. 29, 32, 56

[Müller et al., 2008a] Müller, K., Smolic, A., Dix, K., Kauff, P., & Wiegand, T. (2008a). Reliability-based generation and view synthesis in layered depth video. *Multimedia Signal Processing (MMSP), IEEE International 10th Workshop on*, (pp. 34–39). 13, 65, 99, 100

[Müller et al., 2008b] Müller, K., Smolic, A., Dix, K., Merkle, P., Kauff, P., & Wiegand, T. (2008b). View synthesis for advanced 3d video systems. *EURASIP Journal on Image and Video Processing*, 8, 1–11. 64

[Müller et al., 2009] Müller, K., Smolic, A., Dix, K., Merkle, P., & Wiegand, T. (2009). Coding and intermediate view synthesis of multiview video plus depth. *IEEE International Conference on Image Processing (ICIP)*, 16, 741–744. 84

[Müller et al., 2004] Müller, K., Smolic, A., Merkle, P., Kaspar, B., Eisert, P., & Wiegand, T. (2004). 3d reconstruction of natural scenes with view-adaptive multi-texturing. In *3D Data Processing, Visualization and Transmission (3DPVT), 2nd International Symposium on* (pp. 116–123). 41, 42, 91

[Ndjiki-Nya et al., 2011] Ndjiki-Nya, P., Koppel, M., Doshkov, D., Lakshman, H., Merkle, P., Müller, K., & Wiegand, T. (2011). Depth image-based rendering with advanced texture synthesis for 3-d video. *Multimedia, IEEE Transactions on*, 13(3), 453–465. 56, 63

[Nguyen et al., 2009] Nguyen, Q. H., Do, M. N., & Patel, S. J. (2009). Depth image-based rendering from multiple cameras with 3d propagation algorithm. In *Immersive Telecommunications (IMMERSCOM), Proceedings of*, volume 6 (pp. 1–6). ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 56, 63, 68, 69, 78

[Oh et al., 2009] Oh, K.-J., Yea, S., & Ho, Y.-S. (2009). Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-d video. In *Picture Coding Symposium (PCS)* (pp. 233–236). Piscataway, NJ, USA: IEEE Press. 17, 67

[Onural et al., 2006] Onural, L., Sikora, T., Ostermann, J., Smolic, A., Civanlar, M. R., & Watson, J. (2006). An assessment of 3dtv technologies. *National Association of Broadcasters: Broadcast Engineering Conference*, (pp. 456–467). 27

[Pollefeys et al., 1999] Pollefeys, M., Koch, R., & Gool, L. J. V. (1999). A simple and efficient rectification method for general motion. In *ICCV'99* (pp. 496–501). 29

[Richter-Gebert & Richter-Gebert, 2011] Richter-Gebert, J. & Richter-Gebert, J. (2011). Working with diagrams. In *Perspectives on Projective Geometry* (pp. 247–267). Springer Berlin Heidelberg. 57

[Sarim et al., 2009] Sarim, M., Hilton, A., & Guillemaut, J.-Y. (2009). Wide-baseline matte propagation for indoor scenes. In *Conference Visual Media Production (CVMP), Proceedings of* (pp. 195–204). Washington, DC, USA: IEEE Computer Society. 64

[Sayood, 2005] Sayood, K. (2005). *Introduction to Data Compression, Third Edition*. Morgan Kaufmann Series in Multimedia Information and Systems, 3 edition. 29

[Schreer et al., 2005] Schreer, O., Kauff, P., & Sikora, T. (2005). *3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication*. John Wiley & Sons. 29

[Shade et al., 1998] Shade, J., Gortler, S., He, L.-w., & Szeliski, R. (1998). Layered depth images. In *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), Proceedings of* (pp. 231–242). New York, NY, USA: ACM. 9, 39, 40, 65, 88, 89, 90, 91

[Shimizu et al., 2007] Shimizu, S., Kitahara, M., Kimata, H., Kamikura, K., & Yashima, Y. (2007). View scalable multiview video coding using 3-d warping with depth map. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1485–1495. 50

[Shum et al., 2007] Shum, H.-Y., Chan, S.-C., & Kang, S. B. (2007). *Image-Based Rendering*. Springer, 2 edition. 9, 30, 56

[Shum & Kang, 2000] Shum, H.-Y. & Kang, S. B. (2000). A review of image-based rendering techniques. In *SPIE Digital Library*: Institute of Electrical and Electronics Engineers, Inc. 9, 56

[Slinger et al., 2005] Slinger, C., Cameron, C., & Stanley, M. (2005). Computer-generated holography as a generic display technology. *IEEE Computer Society Press*, 38, 46–53. 31

[Smolic et al., 2007a] Smolic, A., Merkle, P., Müller, K., Fehn, C., Kauff, P., & Wiegand, T. (2007a). Compression of multi-view video and associated data. In *Three-Dimensional Television* (pp. 313–350).: Springer Berlin Heidelberg. 43

[Smolic et al., 2004] Smolic, A., Müller, K., Merkle, P., Rein, T., Kautzner, M., Eisert, P., & Wiegand, T. (2004). Representation, coding, and rendering of 3d video objects with mpeg-4 and h.264/avc. *Multimedia Signal Processing (MMSP), IEEE International 6th Workshop on*, (pp. 379–382). 49

[Smolic et al., 2007b] Smolic, A., Müller, K., Stefanoski, N., Ostermann, J., Gotchev, A., Akar, G., Triantafyllidis, G., & Koz, A. (2007b). Coding algorithms for 3dtv - a survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1606–1621. 51, 88

[Sourimant, 2010a] Sourimant, G. (2010a). *Depth maps estimation and use for 3DTV*. Technical Report 0379, INRIA Rennes Bretagne Atlantique, Rennes, France. 32, 56, 91

[Sourimant, 2010b] Sourimant, G. (2010b). A simple and efficient way to compute depth maps for multi-view videos. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON 2010)*. 30

[Sun et al., 2005] Sun, J., Yuan, L., Jia, J., & Shum, H.-Y. (2005). Image completion with structure propagation. *ACM Trans. Graph.*, 24, 861–868. 65

[Sung-Yeol Kim, 2007] Sung-Yeol Kim, Y.-S. H. (2007). Hierarchical decomposition of depth map sequences for representation of three-dimensional dynamic scenes. *IEICE Trans. on Information and Systems*, E90-D, 1813–1820. 112

[Tanimoto et al., 2008] Tanimoto, M., Fujii, T., Suzuki, K., Fukushima, N., & Mori, Y. (2008). *Reference Softwares for Depth Estimation and View Synthesis*. Technical Report M15377, ISO/IEC JTC1/SC29/WG11 (MPEG), Archamps. 10, 15, 18, 20, 21, 22, 37, 92, 98, 104, 105, 110

[Tanimoto et al., 2009] Tanimoto, M., Fujii, T., Suzuki, K., Fukushima, N., & Mori, Y. (2009). *Depth Estimation Reference Software (DERS) 4.0*. Technical Report M16605, ISO/IEC JTC1/SC29/WG11 (MPEG), London. 32, 56

[Tauber et al., 2007] Tauber, Z., Li, Z.-N., & Drew, M. (2007). Review and preview: Disocclusion by inpainting for image-based rendering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(4), 527 –540. 39, 65

[Telea, 2004] Telea, A. (2004). An image inpainting technique based on the fast marching method. *Journal of Graphics, GPU, and Game Tools*, 9(1), 23–34. 65, 113

[Tsai, 1986] Tsai, R. Y. (1986). An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. Computer Vision and Pattern Recognition (CVPR)* (pp. 364–374). Miami Beach, FL. 29

[Tschumperlé, 2006] Tschumperlé, D. (2006). Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *Computer Vision, International Journal of*, 68(1), 65–82. 65

[Tsung et al., 2009] Tsung, P.-K., Lin, P.-C., Ding, L.-F., Chien, S.-Y., & Chen, L.-G. (2009). Single iteration view interpolation for multiview video applications. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video* (pp. 1–4). 63

[Wang et al., 2007] Wang, H., Li, H., & Li, B. (2007). Video inpainting for largely occluded moving human. *Multimedia and Expo, IEEE International Conference on*, (pp. 1719–1722). 39

[Wang & Cohen, 2007] Wang, J. & Cohen, M. F. (2007). Image and video matting: a survey. *Found. Trends. Comput. Graph. Vis.*, 3, 97–175. 64

[Wang & Adelson, 1994] Wang, J. Y. A. & Adelson, E. H. (1994). Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5), 625–638. 105

[Wang et al., 2004] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4), 600–612. 45

[Waschbüsch et al., 2007] Waschbüsch, M., Würmlin, S., & Gross, M. (2007). 3d video billboard clouds. In *Computer Graphics Forum, Proceedings of Eurographics*, volume 26 (pp. 561–569). Prague, Czech Republic. 39

[Xu & Sun, 2010] Xu, Z. & Sun, J. (2010). Image inpainting by patch propagation using patch sparsity. *Trans. Img. Proc.*, 19, 1153–1165. 66

[Yamamoto et al., 2007] Yamamoto, K., Kitahara, M., Kimata, H., Yendo, T., Fujii, T., Tanimoto, M., Shimizu, S., Kamikura, K., & Yashima, Y. (2007). Multiview video coding using view interpolation and color correction. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11), 1436–1449. 50

[Yamazaki et al., 2002] Yamazaki, S., Sagawa, R., Kawasaki, H., Ikeuchi, K., & Sakauchi, M. (2002). Microfacet billboarding. *Proc. the 13th Eurographics Workshop on Rendering*, 13, 169–179. 37, 38

[Yea & Vetro, 2009] Yea, S. & Vetro, A. (2009). Multi-layered coding of depth for virtual view synthesis. In *Picture Coding Symposium, Proceedings of*, PCS'09 (pp. 425–428). Piscataway, NJ, USA: IEEE Press. 29

[Yoon & Ho, 2007] Yoon, S.-U. & Ho, Y.-S. (2007). Multiple color and depth video coding using a hierarchical representation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17, 1450–1460. 93

[Yoon et al., 2004] Yoon, S.-U., Kim, S.-Y., & Ho, Y.-S. (2004). Preprocessing of depth and color information for layered depth image coding. In *Advances in Multimedia Information Processing - PCM 2004*, volume 3333/2005 (pp. 622–629).: Springer Berlin / Heidelberg. 39

[Yoon et al., 2007] Yoon, S.-U., Lee, E.-K., Kim, S.-Y., & Ho, Y.-S. (2007). A framework for representation and processing of multi-view video using the concept of layered depth image. *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, 46, 87–102. 9, 11, 65, 89, 91, 93, 96

[Yoon et al., 2006a] Yoon, S.-U., Lee, E.-K., Kim, S.-Y., Ho, Y.-S., Yun, K., Cho, S., & Hur, N. (2006a). Coding of layered depth images representing multiple viewpoint video. *Picture Coding Symposium (PCS)*, SS3-2, 1–6. 10, 92, 93

[Yoon et al., 2006b] Yoon, S.-U., Lee, E.-K., Kim, S.-Y., Ho, Y.-S., Yun, K., Cho, S., & Hur, N. (2006b). Inter-camera coding of multi-view video using layered depth image representation. In *Advances in Multimedia Information Processing - PCM 2006*, volume 4261/2006 (pp. 432–441).: Springer Berlin / Heidelberg. 39

[Zach et al., 2007] Zach, C., Pock, T., & Bischof, H. (2007). A duality based approach for realtime tv-l1 optical flow. In *Pattern recognition, 29th DAGM conference on* (pp. 214–223). Berlin, Heidelberg: Springer-Verlag. 84, 85, 86

[Zhang & Chen, 2004] Zhang, C. & Chen, T. (2004). A survey on image-based rendering–representation, sampling and compression. *Signal Processing: Image Communication*, 19(1), 1 – 28. 56

[Zhang, 2000] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22, 1330–1334. 29

[Zitnick et al., 2004] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S., & Szeliski, R. (2004). High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3), 600–608. 19, 21, 36, 64, 88, 102, 110

[Zwicker et al., 2002] Zwicker, M., Pfister, H., van Baar, J., & Gross, M. (2002). Ewa splatting. *Visualization and Computer Graphics, IEEE Transactions on*, 8, 223–238. 41

[Zwicker et al., 2007] Zwicker, M., Yea, S., Vetro, A., Forlines, C., Matusik, W., Zwicker, M., & Electric, M. (2007). Multi-view video compression for 3d displays. In *Signals, Systems and Computers. Conference Record of the Forty-First Asilomar Conference on* (pp. 1506 –1510). 49

Publications

[1] E. Bosc, V. Jantet, L. Morin, M. Pressigout, and C. Guillemot, "Vidéo 3d: quel débit pour la profondeur?," in *COmpression et REpresentation des Signaux Audiovisuels (CORESA)*, (Lyon, France), pp. 1–4, Oct. 2010.

[2] E. Bosc, V. Jantet, M. Pressigout, L. Morin, and C. Guillemot, "Bit-rate allocation for multi-view video plus depth data," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, (Antalya, Turkey), May 2011.

[3] V. Jantet, C. Guillemot, and L. Morin, "Joint projection filling method for occlusion handling in depth-image-based rendering," *3D Research, Topical issue on "Real 3D Imaging and Display"*, vol. 2, pp. 1–13, 2011.

[4] V. Jantet, C. Guillemot, and L. Morin, "Object-based layered depth images for improved virtual view synthesis in rate-constrained context," in *IEEE International Conference on Image Processing (ICIP)*, (Brussels, Belgium), Sept. 2011.

[5] V. Jantet, L. Morin, and C. Guillemot, "Génération, compression et rendu de ldi," in *COmpression et REpresentation des Signaux Audiovisuels (CORESA)*, (Lyon, France), Oct. 2010. Best papers award.

[6] V. Jantet, L. Morin, and C. Guillemot, "Incremental-ldi for multi-view coding," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, (Potsdam, Germany), pp. 1–4, May 2009.

[7] G. Sourimant, T. Colleu, V. Jantet, and L. Morin, "Recalage gps / sig / video, et synthèse de textures de bâtiments," in *COmpression et REpresentation des Signaux Audiovisuels (CORESA)*, (Toulouse, France), pp. 1–4, Mar. 2009. Best papers award.

[8] G. Sourimant, T. Colleu, V. Jantet, L. Morin, and K. Bouatouch, "Toward automatic gis–video initial registration," *Annals of Telecommunications*, vol. 67, pp. 1–13, Mar. 2011.

Layered Depth Images for Multi-View Coding

Vincent Jantet

Manuscrit de Thèse, 2012

Résumé Cette thèse présente un système de traitement des vidéos multi-vues plus profondeur (MVD), en utilisant le concept d'images multi-couches (LDI). Plusieurs contributions sont proposées, à la fois dans le domaine de la synthèse de vues à partir de cartes de profondeur et dans le domaine de la construction et de la compression des LDI.

La méthode de synthèse de vues proposée est basée sur une nouvelle technique de projection (JPF). La projection JPF prend en entrée des données de type image plus profondeur et synthétise un nouveau point de vue quelconque ; elle est conçue pour détecter et remplir les fissures et les petits découvrements pendant la projection. Une mesure de confiance sur les pixels est introduite afin d'éviter les artéfacts fantômes au niveau des contours. Cette projection JPF est utilisée d'une part avec une technique de ré-alignement de textures lors de l'interpolation de vues intermédiaires, et d'autre part avec une nouvelle méthode de synthèse de textures lors de l'extrapolation de vues virtuelles.

Afin d'encoder efficacement les séquences de LDI, un schéma de compression multi-vues (MVC/AVC) est adapté pour exploiter à la fois les redondances temporelles et inter-couche. Une construction incrémentale des LDI est proposée, qui réduit le taux de remplissage des couches additionnelles. L'efficacité de la compression est améliorée par une réorganisation des couches par objet, assurant ainsi une meilleure cohérence spatiale. Deux méthodes de rendu sont finalement proposées : la première utilise la projection JPF alors que la seconde utilise des maillages 3D pour un affichage temps réel sur un écran multi-scopiques.

Abstract This thesis presents an advanced framework for multi-view plus depth video processing and compression based on the concept of layered depth image (LDI). Several contributions are proposed for both depth-image based rendering and LDI construction and compression.

The first contribution is a novel virtual view synthesis technique called Joint Projection Filling (JPF). This technique takes as input any image plus depth content and provides a virtual view in general position and performs image warping while detecting and filling cracks and other small disocclusions. A pixel confidence measure is introduced to avoid ghosting artifacts in the rendered views. For intermediate view interpolation, JPF is used in collaboration with a floating texture realignment technique. For virtual view extrapolation, JPF is combined with a novel full-Z depth aided inpainting technique.

In order to efficiently encode the proposed LDI representation, a compression scheme based on MVC/AVC standard is adapted to exploit both temporal redundancies and inter-layer redundancies in the LDI sequence. An incremental construction scheme for the LDI is proposed, called I-LDI. This construction scheme reduces the completion rate of additional layers. An object-based layer organization of the LDI is then presented which ensures spatial consistency of each layer, and thus improves compression efficiency in comparison with a standard AVC/MVC scheme in rate-constrained context. Two rendering methods are finally proposed: the first one uses the JPF method, while the second one uses a 3D mesh for real-time rendering on an eight-views auto-stereoscopic display.