Système multimodal de prévisualisation “on set” pour le cinéma

Timothée de Goussencourt

UNIVERSITÉ GRENOBLE ALPES

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE-ALPES

Spécialité : **Signal, image, paroles, télécoms**

Arrêté ministériel : 7 août 2006

Présentée par
Timothée de GOUSSENCOURT

Thèse dirigée par **Pascal BERTOLINO**
préparée au sein du laboratoire
Grenoble Images Parole Signal Automatique (GIPSA-lab)
dans l'école doctorale
Électronique Électrotechnique Automatique et Traitement du Signal (EEATS)

Système multimodal de prévisualisation "on-set" pour le cinéma

Thèse soutenue publiquement le **19 décembre 2016**,
devant le jury composé de :

Pierre-Yves COULON
Grenoble INP, Président du jury
Marc VAN DROOGENBROECK
Université de Liège, Rapporteur
Christian JACQUEMIN
Université Paris-Sud, Rapporteur
Christian GUILLON
ENS Louis Lumière, Examinateur
Gérard BRIAND
Technicolor, Examinateur
Pascal BERTOLINO
Université Grenoble-Alpes, Directeur de thèse
Jean-François SZLAPKA
Solidanim, Invité
Gilles POULIQUEN
Nintendo, Invité

UNIVERSITÉ GRENOBLE-ALPES

ÉCOLE DOCTORALE EEATS

Électronique Électrotechnique Automatique et Traitement du Signal

T H È S E

pour obtenir le titre de

docteur en sciences

de l'Université de Grenoble-Alpes

Mention : Traitement d'images

Présentée et soutenue par

Timothée de GOUSSENCOURT

Système multimodal de prévisualisation "on-set" pour le cinéma

Thèse dirigée par Pascal BERTOLINO

préparée au laboratoire Grenoble Images Parole Signal Automatique (GIPSA-lab)

soutenue le 19 décembre 2016

Jury :

Président :	Pierre-Yves COULON	-	Grenoble INP
Rapporteurs :	Marc VAN DROOGENBROECK	-	Université de Liège
	Christian JACQUEMIN	-	Université Paris-Sud
Examinateurs :	Christian GUILLON	-	ENS Louis Lumière
	Gérard BRIAND	-	Technicolor
Directeur :	Pascal BERTOLINO	-	Université Grenoble-Alpes
Invités :	Jean-François SZLAPKA	-	Solidanim
	Gilles POULIQUEN	-	Nintendo

Remerciements

Si ces mots sont écrits c'est que la thèse est maintenant terminée. Il m'a bien fallu quelques semaines avant de pouvoir revenir sur ce manuscrit ![1] Je ne regrette rien de ces trois années intenses et je veux remercier toutes les personnes qui ont contribué à la réussite de celle-ci.

Je tiens dans un premier temps à remercier tous les membres du jury qui ont pris le temps de lire et d'évaluer mon travail. Le sujet est transversal, le jury l'était également. Je suis fier d'avoir pu défendre mon mémoire devant eux et je les remercie de m'y avoir autorisé.

Je tiens plus particulièrement à remercier Pascal Bertolino mon directeur de thèse. J'ai apprécié sa patience et la liberté qu'il m'a donné. Je tiens spécifiquement à lui témoigner toute ma gratitude pour ses conseils, son enthousiasme et son intuition. Merci Pascal.

Je remercie SolidAnim pour sa confiance. Je tiens à remercier l'ensemble de ses collaborateurs. Ceux qui m'ont donné l'envie de me lancer dans la réalisation d'une thèse et tous les ingénieurs de mon équipe R&D. Un grand merci également aux intermittents du spectacle qui ouvrent nos horizons : les graphistes, les animateurs, les mocap cleaner, les modeleurs, les riggeurs.

Je tiens à remercier tous les doctorants du Gipsa-lab. La fabuleuse équipe des Gipsa King. Merci pour votre bonne humeur. J'ai vraiment découvert une famille dans ces murs. Merci pour les coinches, les parties de baby, les débats, les repas à 11h30 au restau U.

Merci à tous mes amis. Ceux qui me suivent de près ou de loin. Merci pour votre soutien ! Un merci spécial à tous ceux qui m'ont hébergé lors de mes voyages à Paris !

Merci à ma famille. Mes parents, mon frère et ma sœur pour leur présence. Vous avez toujours été là pour moi. Un merci plus particulier à ma maman pour la relecture. Un merci aussi à ma marraine pour son accompagnement et de très bons moments parisiens.

Le meilleur pour la fin. Merci à toi qui me fait sans arrêt découvrir le monde avec un regard nouveau. Sans toi il faut bien l'avouer jamais je ne l'aurai fini ce manuscrit !

1. Le lecteur appréciera la concision de ce chapitre qui traduit l'intensité de tous les échanges, les idées, les personnes rencontrées...plutôt que des mots place à l'émotion !

Table des matières

Table des figures

Liste des tableaux

Table des sigles et acronymes

FUI	*Fonds Unique Interministériel*
HD	*Haute Définition*
ToF	*Time of Flight*
CGI	*Computer Generated Imagery*
CG	*Computer Graphic*
SLAM	*Simultaneous Localization And Mapping*
SDK	*Software Development Kit*
RGB	*Red Greeen Blue*
RGBD	*Red Greeen Blue Depth*
GPU	*Graphic Processing Unit*
GLSL	*Open**GL** **S**hading **L**anguage*
HDR	*Hight Dynamic Range*
LUT	*Look Up Table*
IBL	*Image Based Lighting*
CCD	*Charge Couple Device*
CMOS	*Complementary Metal Oxide Semiconductor*
SDI	*Serial Digital Interface*
3D	*Trois Dimensions*

Introduction

Une thèse appliquée au domaine de l'audiovisuel

Un partenaire industriel : *SolidAnim*

Ce travail de recherche est réalisé en partenariat avec la société *SolidAnim* au sein de laquelle j'occupe le poste d'ingénieur R&D depuis 2011. Créée par 3 fondateurs en 2007, l'entreprise compte désormais une vingtaine de collaborateurs (ingénieurs, animateurs 3D [2], graphistes) à temps plein. L'effectif varie en fonction des productions. La société *SolidAnim* est spécialisée en animation 3D pour la création d'effets visuels dans le domaine du cinéma, la télévision et les jeux vidéos. L'entreprise utilise un système optique de captation de mouvements basé sur des marqueurs pour créer ses animations. Cette technique permet de capter les mouvements d'un acteur afin de les reproduire sur un personnage virtuel. En tant qu'ingénieur, je travaille sur des algorithmes de vision pour la captation du regard, la détection des expressions faciales sans marqueur et la fusion de données d'un réseau de caméras de profondeur. Depuis 2011, *SolidAnim* développe sa solution propre de prévisualisation d'effets spéciaux en temps réel (voir figure 1). Ce travail de recherche s'inscrit dans le développement de cette technologie. L'entreprise a par ailleurs participé à d'importants projets audiovisuels dont Charlie et la chocolaterie, Hugo Cabret, The Walk, Warcraft et Ghostbuster 3.

Thèse financée par le Fonds Unique Interministériel : le projet *Previz*

Mon travail de recherche s'inscrit dans un programme FUI [3]. Le FUI est un dispositif destiné à soutenir la recherche appliquée et permet de financer des projets de R&D collaboratifs entre industriels et laboratoires. Lors du 15^{eme} appel à projet FUI, *SolidAnim* et le *GIPSA-lab* ont pris part à un consortium regroupant au total 9 partenaires industriels et académiques, afin de présenter le projet *Previz*. Celui-ci a été validé par les pôles de compétitivité *Imaginove*, *Cap Digital*, *Images et Réseaux*. Il a démarré en juin 2013 pour une durée de 2 ans. Ce projet est décrit dans [BBZ+14] et est également visible au travers de son site internet [4]. La liste complète des partenaires est donnée dans le tableau ci-dessous (tableau 1).

La prévisualisation pendant le tournage

Dans le domaine du cinéma, le terme prévisualisation, aussi appelé *previz*, fait souvent référence à l'élaboration d'une séquence en images de synthèse préparée en amont du tournage.

2. *Trois Dimensions*
3. *Fonds Unique Interministériel*
4. `http://previz.eu/`

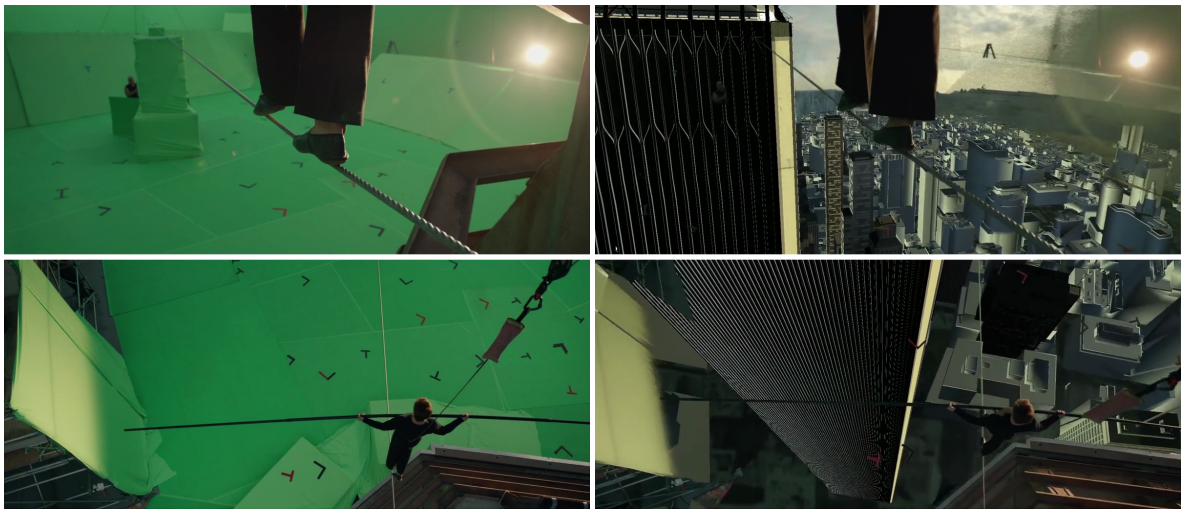


FIGURE 1 – Exemple de prévisualisation réalisée par *SolidAnim* dans le cadre de la réalisation du long métrage *The Walk* de Robert Zemeckis sorti en 2015. Il s'agit d'une scène où l'acteur exécute la traversée des *Twin Towers* de New-York sur un câble. A gauche, les rushes originaux tournés en studio. A droite, la prévisualisation en temps réel diffusée au réalisateur pendant le tournage et utilisée pour réaliser la séquence.

Cette séquence, nommée également animatique, est une aide visuelle apportée aux équipes du film pour anticiper le tournage. La préviusalisation n'est pas une exclusivité réservée aux films avec effets spéciaux. Tous les films peuvent faire appel à la prévisualisation afin de préparer le tournage. Dans cette thèse nous nous intéressons à la prévisualisation qui a lieu pendant le tournage.

Le projet *Previz* est lié à l'industrie du cinéma avec effets spéciaux. Il se situe à l'interface des méthodes développées pour la conception d'un film telles que le *storyboard*, et la post-production qui réalise le plan final. Prenons l'exemple de la séquence de la traversée des *Twin Towers* dans le film *The Walk* (figure 1). Les plans sont tournés avec des acteurs réels sur un fond vert. Dans un schéma sans prévisualisation en temps réel la scène est d'abord tournée en studio, puis les rushes sont envoyés aux équipes de post-production qui remplaceront le fond vert par un décor. La proposition de la prévisualisation pendant le tournage, aussi appelée *previz on-set*, est de pouvoir montrer au réalisateur une vue assemblée du plan final pendant le tournage. Le fond vert est remplacé par le décor final. Ce dispositif correspond à un système interactif, de telle sorte que si la caméra se déplace, l'affichage des éléments virtuels du plan est automatiquement mis à jour afin que l'ensemble reste cohérent. L'intérêt de la *previz on-set* n'est pas seulement de développer la créativité du réalisateur. Pendant le tournage, l'enregistrement des séquences de la prévisualisation et l'enregistrement de toutes les données qui lui sont associées permet aux équipes de post-production d'avoir une meilleure compréhension du plan final à produire, et potentiellement d'accélérer sa création. La prévisualisation pendant le tournage permet également d'éviter les erreurs dans le cadrage des plans ou dans le choix des focales à utiliser. La *previz on-set* permet d'optimiser les prises de vue (rushes) qui seront

Partenaire	Type
École Nationale Supérieure Louis Lumière	académique
GIPSA-lab	laboratoire
IRISA	laboratoire
LIRIS	laboratoire
Loumasystems	entreprise
Polymorph	entreprise
SolidAnim	entreprise
Technicolor	entreprise
Ubisoft	entreprise

Tableau 1 – Liste des partenaires du projet *Previz*

utilisées en post-production. L'objectif du projet *Previz* est de construire une méthodologie visant à optimiser la chaîne de traitement de la post-production, tout en favorisant la créativité du réalisateur lors du tournage.

Dans la suite de ce manuscrit nous utilisons souvent les termes de prévisualisation ou de *previz*. Dans le cadre spécifique de ce travail, ces termes font référence à la prévisualisation pendant le tournage et à la *previz on-set*.

Le travail présenté dans cette thèse s'intéresse à une étape spécifique de la prévisualisation : le *compositing*. Cette étape consiste à mélanger plusieurs sources d'images pour composer un plan unique et cohérent. Dans notre cas, il s'agit de mélanger une image de synthèse avec l'image issue de la caméra présente sur le plateau de tournage. Les effets spéciaux numériques sont ainsi ajoutés à la prise de vue réelle. L'objectif de cette thèse consiste donc à proposer un système permettant l'ajustement automatique du mélange entre l'image de synthèse et l'image réelle. La méthode proposée nécessite la mesure de la géométrie de la scène filmée. Pour cette raison, un capteur de profondeur est ajouté à la caméra de tournage. Les données sont envoyées à l'ordinateur qui exécute un algorithme permettant de fusionner les données du capteur de profondeur et de la caméra de tournage. La carte graphique de l'ordinateur est sollicitée pour assurer un fonctionnement en temps réel. Pour réaliser ce dispositif, le choix a été fait d'utiliser un moteur de jeux vidéo.

Organisation du manuscrit

Pour répondre à cet objectif la démarche a été de développer un prototype fonctionnel.

Le chapitre 1 décrit précisément le contexte de la prévisualisation au cinéma. Un rappel est fait sur l'évolution des effets spéciaux au cinéma. Une mise en contexte permet de comprendre les mécanismes mis en jeux dans le trucage d'un plan numérique. L'accent est mis sur la gestion des occultations qui constitue le cœur de cette thèse. Nous détaillons les méthodes

actuellement mises en place avant de dresser une esquisse du système bâti dans la cadre de notre travail de recherche. Les verrous technologiques sont également énoncés afin de bien comprendre les enjeux.

Le chapitre 2 dresse le bilan des outils existants pour capter la profondeur d'une scène en temps réel. Nous justifions le choix du capteur à temps de vol *Kinect 2* dans notre application et nous développons le système matériel mis en place, comprenant la caméra de tournage et le capteur de profondeur. Une étape de calibration [5] des capteurs est nécessaire afin d'assurer le bon fonctionnement du système. Nous expliquons l'intégration de notre système matériel avec le moteur de jeux vidéos. Une description des moteurs de jeux est faite afin de relever leurs principaux atouts pour notre domaine d'application. En effet, le moteur de jeux correspond à la plateforme centrale de notre cadre de travail. En ce sens, nous développons une description précise de l'utilisation du pipeline graphique. Cette implémentation est illustrée par plusieurs images issues des séquences de réalité augmentée produites avec notre système.

Cette implémentation comporte des limites. L'objet du chapitre 3 est précisément de développer une méthode permettant de dépasser ces limites, notamment celles concernant la qualité des masques aux abords du contour des objets. L'objectif de cette méthode est d'affiner le contour du masque de délimitation d'un objet issu du capteur de profondeur en prenant en compte l'image couleur qui lui est associée. Une comparaison avec d'autres méthodes détaillées dans l'état de l'art est également proposée.

La dernière partie synthétise les principaux résultats obtenus et présente diverses perspectives pour les futurs travaux.

Le domaine d'application de notre travail produit de nombreux termes spécifiques. Nombre de ces termes sont employés en anglais dans la rédaction de ce manuscrit. Nous tenons à préciser que les termes anglais conservés sont ceux utilisés par la communauté scientifique internationale.

5. Le terme exact en français est calibrage, néanmoins nous utiliserons l'anglicisme *calibration*.

Chapitre 1

Contexte

Sommaire

1.1 Les trucages au cinéma

Ce travail de recherche est lié au développement des effets spéciaux au cinéma. Avant d'aborder les méthodes d'insertion de plan virtuel dans le plan réel, et en lien avec notre sujet, ce chapitre se focalise sur quelques éléments déterminants dans l'histoire du cinéma en lien avec notre sujet. L'objet de ce chapitre est donc de présenter l'évolution des techniques utilisées au cinéma pour créer l'illusion, et d'introduire l'apparition du numérique.

1.1.1 Un peu d'histoire

La photographie permet de faire une projection de la réalité sur un support en deux dimensions. Les artistes ont très tôt cherché à truquer cette projection. Il ne s'agit plus de photographier le monde extérieur, il s'agit de le mettre en scène. Dès lors, l'apparition du studio photo permet de contrôler tous les paramètres nécessaires à l'obtention du cliché recherché : il peut s'agir de l'éclairage mais également d'un décor reconstitué. C'est le début des effets

spéciaux. Ceux-ci peuvent être élaborés en amont ou réalisés pendant le tournage. Une seconde catégorie de trucages concerne la manipulation a posteriori, c'est-à-dire après que la scène ait été prise en photo. Soit en ajoutant des éléments au résultat final sans que ceux-ci n'aient été présents au moment de la captation, soit en effaçant a posteriori certains éléments du plan. Une technique consiste à photographier plusieurs éléments à des instants ou des lieux différents, puis à découper les éléments intéressants dans chaque photographie afin de les regrouper dans une seule photographie. La technique consiste à placer les images côte-à-côte, à les recouvrir partiellement ou encore à mélanger certains éléments visuels. Cette pratique s'appelle le *compositing*. Oscar Gustave Rejlander est considéré comme l'un de ses initiateurs (figure 1.1).

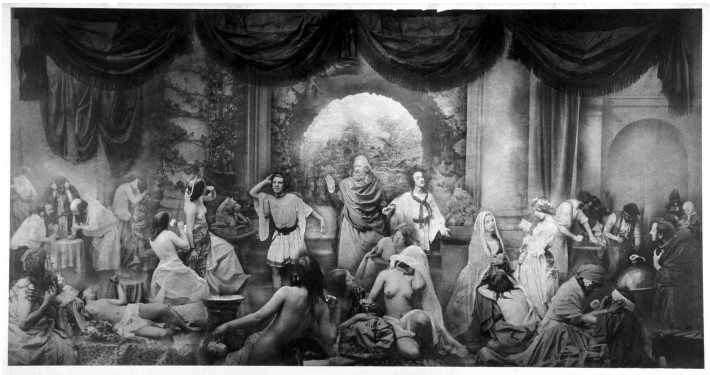


FIGURE 1.1 – Un des premiers *compositing* de l'histoire : *The Two Ways of Life* de Oscar Gustave Rejlander, réalisé en 1857. L'image finale est composée de 32 images.

Le trucage photographique s'est également étendu à la vidéo et notamment au cinéma. On assimile souvent Georges Méliès comme le père des effets spéciaux. *Le voyage dans la lune* est considéré comme le premier film de science fiction de l'histoire (figure 1.2). Les trucages intervenaient aussi bien pendant la prise de vue qu'en post-production et les films étaient coloriés image par image au pinceau. Il n'était pas rare de redessiner des arrières plans à la main sur chaque image d'une pellicule. C'est ce qu'on appelle le *matte-painting*.

FIGURE 1.2 – Extrait de la pellicule en couleur du film *Le voyage dans la lune* de Georges Méliès, 1902.

Dans le cadre de ce travail de recherche nous nous intéressons aux trucages qui sont issus d'une manipulation de la pellicule. L'apparition du numérique a favorisé le recours à ce procédé.

1.1.2 Apparition du numérique

On peut distinguer plusieurs étapes dans la révolution numérique au cinéma. La première étape est marquée par l'apparition des ordinateurs pour traiter et produire des images. L'abandon de la pellicule pour les capteurs CCD [1] ou CMOS [2] (captation numérique) marque une seconde révolution, et enfin l'équipement de projecteurs numériques dans les salles de projection remplace la traditionnelle bobine et marque l'avènement de la diffusion numérique.

L'arrivée des ordinateurs a permis la création des images de synthèse (abrégé en anglais CGI [3]). L'un des premiers films à utiliser massivement ces images est sans doute le film *Tron* sorti en 1982 (figure 1.3). Bien que le film fut un échec lors de sa sortie, il est considéré comme un évènement majeur dans l'histoire du cinéma, ayant inspiré de nombreuses créations a posteriori. L'usage des ordinateurs a pris, petit à petit, une place plus importante dans l'industrie du cinéma. Il ne s'agit pas toujours comme dans *Tron* de générer une image entièrement de synthèse. Il s'agit d'ajouter des éléments numériques à l'image d'origine, ou bien de manipuler les bobines. Pour cela, les pellicules sont scannées pour que chaque image puisse être traitée numériquement : effets spéciaux, montage, étalonnage, ... autant de procédés démultipliés par l'usage du numérique.

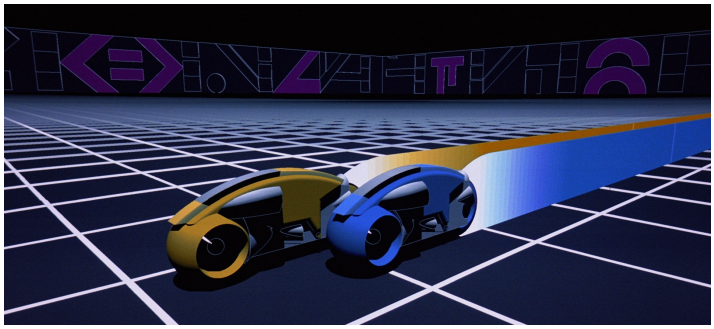


FIGURE 1.3 – Une image de synthèse générée pour le film *Tron* sorti en 1982.

C'est en 2001, avec le film *Vidock*, qu'une production est tournée pour la première fois à l'aide d'une caméra numérique. Il est suivi de près par l'épisode 2 de *Star Wars*, *L'attaque des clones*. Les capteurs numériques sont intégrés aux caméras de tournage, les photosites de la pellicule sont remplacés par les pixels de l'image numérique.

A la fin de cette chaîne de production, le film est imprimé sur bobine afin d'être diffusé dans les salles de projection. Il faut attendre les années 2000 pour que les salles de cinéma s'équipent majoritairement de projecteurs numériques.

Dans ce manuscrit, nous nous intéresserons au *compositing* de l'image vidéo numérique avec des *digital matte*, c'est-à-dire au mélange de l'image issue de la caméra de tournage avec des images de synthèse générées par ordinateur.

1. *Charge Couple Device*
2. *Complementary Metal Oxide Semiconductor*
3. *Computer Generated Imagery*

Retenons de cette section la définition du terme *compositing*. Celui-ci désigne le fait de mélanger plusieurs images de sources différentes afin de produire une image finale cohérente. Notre travail s'intéresse au mélange automatique d'une image de synthèse avec une image réelle issue d'une caméra de tournage, intimement liée à l'apparition du numérique.

1.2 Insérer un élément virtuel dans un plan vidéo

Différentes étapes déterminantes permettent d'intégrer un élément virtuel dans une séquence vidéo. Ces étapes sont réalisées traditionnellement en post-production pour truquer un plan. Dans notre propos, nous désignons par *caméra film* la caméra utilisée pour produire la séquence vidéo. Chaque image de cette séquence est appelée image réelle. L'élément virtuel est un élément modélisé numériquement en trois dimensions. On l'appelle aussi élément 3D. Le terme 3D fait référence à la manière dont il est créé, il ne s'agit pas de la façon dont il est visualisé par le spectateur final (lunettes 3D). Un ensemble de nœuds (*vertices*) sont reliés entre eux par des arrêtes (*edges*) et forme un maillage (*mesh*) qui représente l'élément que l'on veut modéliser. Cela correspond à la modélisation surfacique de l'élément avec un certain nombre de triangles. Ce maillage est défini numériquement, de sorte que chaque nœud est référencé dans un fichier, ainsi que les indices des nœuds reliés entre eux. Ensuite, un moteur de rendu permet de passer de la représentation numérique du maillage 3D à une image en 2D sur un écran classique. Il s'agit en fait de la projection de l'objet 3D au travers d'une caméra virtuelle d'observation. La projection 2D d'un élément virtuel est une image de synthèse. La figure 1.4 illustre la visualisation d'un modèle 3D à travers plusieurs caméras virtuelles.

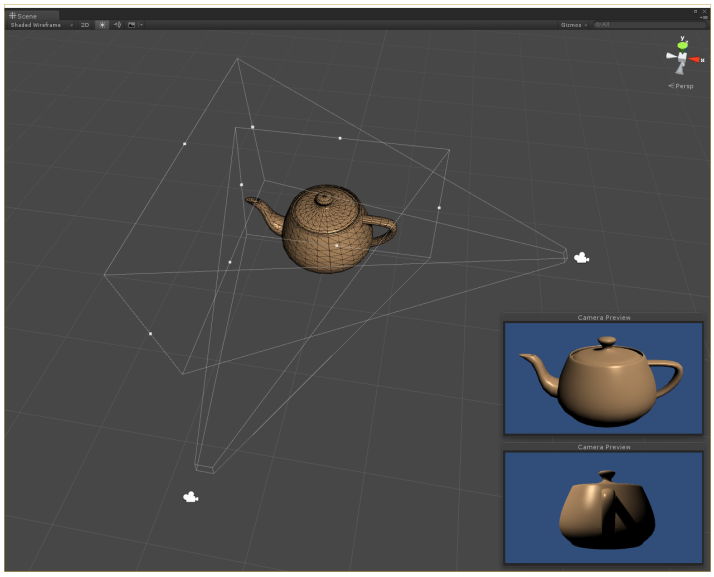


FIGURE 1.4 – Capture d'écran de *Unity 3D*. On peut repérer l'objet *teapot* modélisé en 3D et apercevoir le maillage du modèle. Deux caméras virtuelles sont placées autour du modèle. A travers chacune des caméras virtuelles, le rendu du modèle 3D est affiché en bas à droite de la figure.

Afin d'intégrer l'élément virtuel dans chaque image de la séquence vidéo, trois étapes sont importantes. Il est tout d'abord nécessaire de déterminer le point de vue d'observation de l'élément virtuel, afin de générer dans un second temps le rendu de celui-ci, pour enfin mélanger l'image de synthèse obtenue avec l'image réelle.

1.2.1 Choix du point de vue

Dans l'objectif d'intégrer un objet 3D à une image, il faut déterminer la position de la caméra virtuelle qui produit une projection de l'objet 3D spatialement cohérente avec les éléments de l'image réelle. Il est nécessaire de connaître les paramètres optiques, tels que la focale et la distorsion de la caméra film utilisée. On applique ces paramètres à la caméra virtuelle. La position dans l'espace de la caméra virtuelle correspond exactement à la position de la caméra film. La figure 1.5 permet de visualiser une image réelle en vis-à-vis d'une image de synthèse générée avec une caméra virtuelle à la même position que la caméra film.

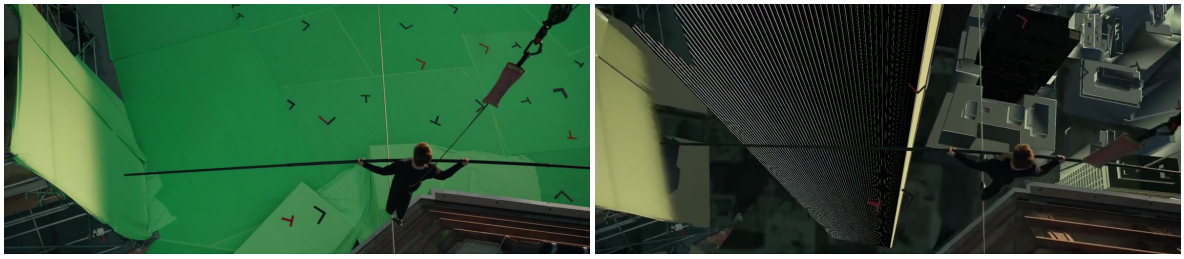


FIGURE 1.5 – Images extraites du film *The Walk*, 2015. Gauche : rush original tourné en studio. Droite : une caméra virtuelle avec les même propriétés que la caméra de tournage (focale, position) permet de faire un rendu de l'arrière plan virtuel avec le bon point de vue.

Le calcul de la trajectoire de la caméra de tournage, appelé *camera tracking*, est longtemps resté une étape de post-traitement. Des logiciels spécialisés tels que 3DEqualizer [4] ou PFTrack [5] permettent de calculer le mouvement de la caméra. Généralement les logiciels procèdent en 2 étapes. La première étape consiste à identifier et suivre des points 2D dans une séquence vidéo (*motion tracking*). Ensuite, à partir de l'ensemble de ces trajectoire 2D, le logiciel est capable de calculer le mouvement 3D de la caméra. Ce calcul n'est pas temps réel. Il est ensuite possible d'incruster un élément 3D dans l'image qui sera en accord avec le mouvement de la caméra (*matchmove*).

Au regard des récentes avancées dans le domaine, il est désormais possible d'avoir des systèmes temps réel qui réalisent cette tâche. Un système de *camera tracking*, associé à une caméra de tournage, permet de connaître la trajectoire tridimensionnelle de la caméra film à tout instant. Il est alors possible d'appliquer cette trajectoire à une caméra virtuelle dans un moteur de rendu. La caméra virtuelle génère ensuite l'image de synthèse qui se superpose de façon parfaite à l'image de la caméra film. Ainsi, l'image réelle et l'image de synthèse sont-elles synchronisées à la fois spatialement et temporellement.

Les technologies employées pour le *camera tracking* peuvent être mécaniques ou bien optiques. L'entreprise *Loumasystem*, partenaire du projet *Previz*, est un fabricant de grues pour caméra. Au sein de leur grue *Louma 2* chaque degré de liberté est encodé ce qui permet de connaître la position de la caméra dans l'espace. Avec sa solution *solidTrack*, *SolidAnim* pro-

4. `https://www.3dequalizer.com/`
5. `http://www.thepixelfarm.co.uk/pftrack/`

pose une technologie optique sans marqueur. Un algorithme de traitement d'images permet de se localiser dans l'espace en fonction de l'environnent observé (SLAM [6]) [KM07]. Dans ce manuscrit, on considère que la position de la caméra film est connue.

1.2.2 Rendu

La qualité du rendu d'un élément virtuel est déterminante pour obtenir une bonne intégration dans une image réelle. L'élément 3D doit être correctement modélisé pour être crédible. L'éclairage, la qualité des textures et le comportement du matériau sont autant d'éléments à prendre en compte afin de générer un rendu de qualité. La figure 1.6 donne un aperçu d'un rendu final obtenu après post-production.

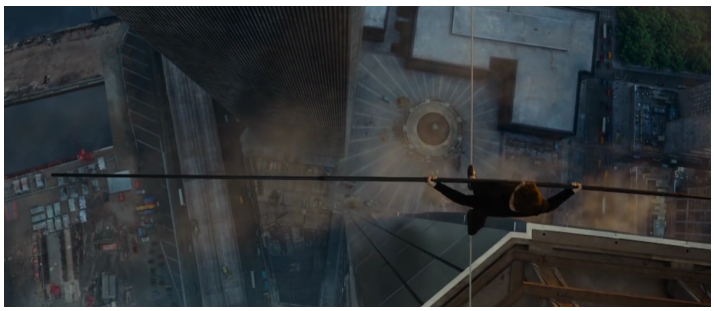


FIGURE 1.6 – Image extraite du film *The Walk*, 2015. Le rendu final est beaucoup plus travaillé que le rendu de la *previz*.

Dans le cadre de ce travail, nous nous attachons à produire un rendu en temps réel. Les modèles 3D utilisés ne sont pas aussi aboutis que les modèles utilisés pour le rendu final. Les textures et la lumière sont simplifiées. L'objectif est d'obtenir le meilleur résultat possible en respectant la condition du temps réel.

1.2.3 Mélange de l'image de synthèse avec l'image réelle

L'action de mélanger une image de synthèse avec une image réelle renvoie à la notion de *compositing*. L'image réelle et l'image de synthèse sont observées avec le même point de vue. Mélanger ces deux images consiste donc à définir une hiérarchie dans l'affichage. La question qui se pose pour chaque niveau de plan est de savoir si le premier plan est issu de l'image réelle ou s'il est issu de l'image de synthèse. La manipulation consiste à découper dans chacune des images les éléments indépendants, puis à les assembler entre eux. Une des approches retenues consiste à considérer que chaque élément découpé fonctionne comme un calque (*layer* en anglais). Le calque apparaît opaque quand il y a un élément et il apparaît transparent quand il n'y en a pas. Les calques sont empilés de manière hiérarchisée, de sorte que le premier plan

6. *Simultaneous Localization And Mapping*

se situe au premier rang et le dernier plan au dernier rang. Une zone transparente dans un calque laisse entrevoir le calque qui est en dessous.

Concernant les éléments virtuels, les dimensions des objets sont parfaitement connues. La distance des objets à la caméra est également maitrisée. On peut alors demander au moteur de rendu de générer une image de synthèse pour chaque niveau de plan. Prenons une scène 3D avec au premier plan une créature virtuelle et une ville virtuelle en arrière plan. Le moteur de rendu peut générer un calque comprenant la créature virtuelle et un calque comprenant également la ville virtuelle. Puisque que chaque élément virtuel est modélisé indépendamment, il est possible de faire un rendu pour chaque objet et de générer autant de calques que nécessaire.

Concernant l'image réelle, le découpage se révèle plus complexe. Quel que soit le nombre d'objets dans la scène réelle, on ne dispose que d'une seule image réelle. Il s'agit d'une projection de la réalité en deux dimensions. Les notions d'objets et de distance à la caméra sont perdues lors de la projection. Si l'on veut isoler un élément du premier plan dans un calque il faut alors en déterminer les contours manuellement. On appelle cette étape la rotoscopie. L'image générée par la rotoscopie est appelé masque ou *matte*. Les zones opaques sont définies par du blanc et celles transparentes par du noir. En multipliant le masque avec l'image originale on obtient le calque avec l'objet uniquement. Une solution pour éviter le travail fastidieux de la rotoscopie est de filmer l'objet à détourer sur un fond uni. Il s'agit de la technique utilisée en studio lorsqu'on se sert d'un fond vert. Avec ce procédé, un algorithme de traitement d'image remplace automatiquement par de la transparence le vert dans l'image.

Dans la séquence du film *The Walk*, le premier plan est issu de l'image réelle. Il s'agit de l'acteur qui marche sur le câble. Cette image est le premier calque. L'arrière plan est la ville de New-York générée en image de synthèse. Il s'agit du deuxième calque. Le fond vert du premier calque est effacé pour laisser voir les tours de la ville de New-York sur le second calque (voir figure 1.7). On peut combiner ces techniques afin d'obtenir des superpositions de couches réelles et virtuelles intercalées.

Des logiciels ont été conçus spécialement pour le *compositing* vidéo. Ces logiciels proposent divers outils pour faire du *matchmove*, de la rotoscopie, de la suppression de fond vert et de la gestion de calques. On peut citer *After Effects*[7] de *Adobe* ou bien *NUKE*[8], développé par *The Foundry* qui sont très utilisés dans l'univers de la post-production. Plus récemment l'*INRIA* a développé une solution assez similaire du nom de *NATRON*[9]. On trouve dans [Bri08] les explications détaillées des méthodes nécessaires au trucage d'un plan vidéo.

7. http ://www.adobe.com/fr/products/aftereffects.html
8. https ://www.thefoundry.co.uk/products/nuke/
9. https ://natron.fr/

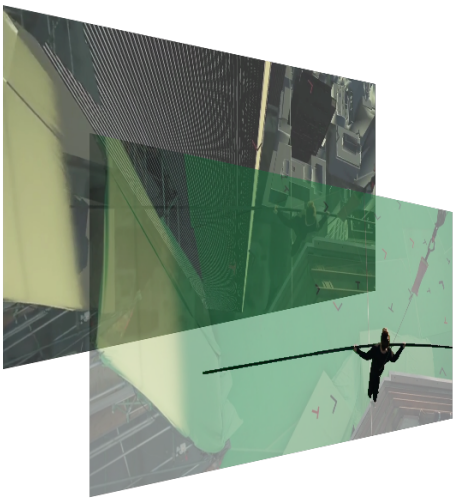


FIGURE 1.7 – Images extraites du film *The Walk*, 2015. Les différentes couches sont visibles : la couche virtuelle en arrière plan, et la couche réelle au premier plan.

Retenons ces trois étapes importantes pour l'incrustation d'un élément virtuel dans une image réelle :

- le *camera tracking*
- le rendu
- le *compositing*

Ces trois étapes sont utilisées en temps réel afin de proposer la visusalisation d'effets spéciaux directement pendant le tournage. La section suivante détaille cette activité.

1.3 La prévisualisation pendant le tournage : *previz on-set*

1.3.1 Définition

Pour Christian Guillon, membre du projet *Previz*, la prévisualisation sur le plateau est un « outil d'assistance à la mise en scène. La prévisualisation permet au réalisateur de voir en direct ce qu'il ne pouvait voir auparavant qu'en post-production : la composition finale de tous les éléments de l'image. Mais elle lui permet surtout d'agir sur ces éléments, de les diriger ensemble, dans un même geste de mise en scène » [10].

Le but du projet *Previz* est d'expérimenter une nouvelle méthodologie lors de la création d'un film. Il s'agit d'ajouter une nouvelle étape, appelée *previz*, pendant le tournage. La création des éléments virtuels et le tournage ne sont plus des étapes décorrélées. La *previz* se situe à l'interface de ces processus afin de proposer une visualisation directement pendant le tournage. Les étapes nécessaires aux mélanges des éléments virtuels avec l'image réelle, décrites dans la section précédente (section 1.2), sont appliquées en temps réel. De plus, les éléments enregistrés pour la *previz* lors du tournage sont envoyés à l'étape de post-production afin d'accélérer cette tâche. La figure 1.8 illustre la place de la *previz* dans le processus de création d'un film.

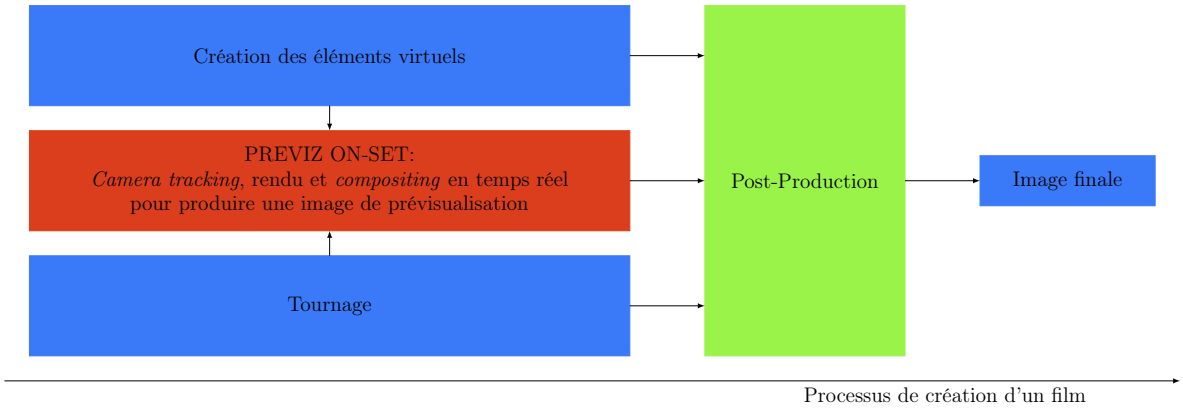


FIGURE 1.8 – Place de la *previz* au sein du processus de création d'un film.

Dans le cadre de la création de film, certaines réalisations nécessitent de manipuler la prévisualisation de manière plus complexe en mélangeant des décors et des acteurs réels avec des éléments virtuels créés par ordinateur. Pour ces productions, diverses étapes de visualisation interviennent. La première étape est l'établissement d'un *storyboard*. Cette étape est initiée au début du processus de création du film. Le *storyboard* constitue une base de travail commune entre tous les intervenants. En outre, il est aujourd'hui courant pour certaines productions de produire une animatique, c'est à dire un *storyboard* animé pour l'ensemble ou certaines scènes du film. La dernière étape de visualisation est la projection du film aux spectateurs. Ce

10. Christian Guillon est enseignant à l'école Louis Lumière ainsi que fondateur de L'EST – précurseur de la présence sur le tournage d'un superviseur VFX – et d'ADN, agence de doublures numériques. La citation est tirée de http ://www.mediakwest.com/production/item/futur-des-vfx-tout-se-joue-avant-et-apres.html

travail de recherche s'intéresse à l'étape de visualisation intermédiaire appelée prévisualisation pendant le tournage. La figure 1.9 illustre les différentes visualisations précédemment citées.

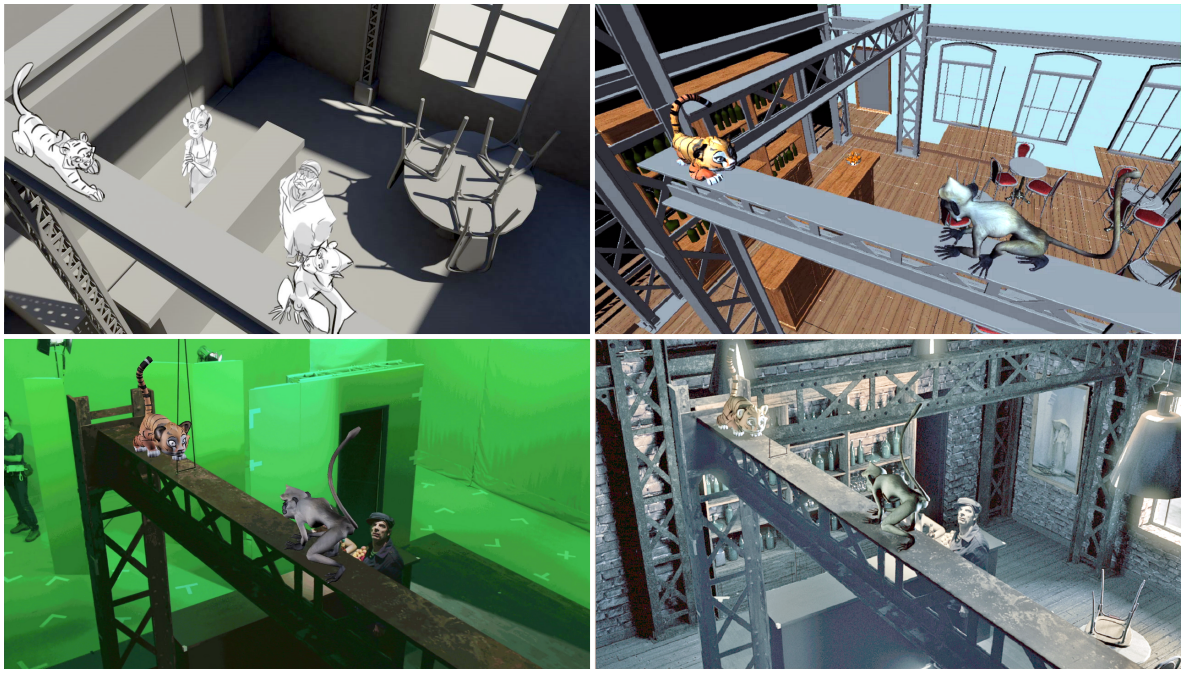


FIGURE 1.9 – Illustration des différentes étapes de visualisation : *storyboard*, animatique, prévisualisation, rendu final. Images extraites de l'expérimentation du projet *Previz* réalisée à l'école nationale supérieure Louis Lumière en 2015.

On trouve une liste intéressante d'expérimentations utilisant la réalité augmentée [11] sur les plateaux de télévision dans [Tho06]. L'article évoque des techniques de suivi de caméra pour superposer les éléments virtuels et les éléments réels. Il est aussi question de segmentation du premier plan dans l'image réelle pour gérer les occultations entre les éléments réels et les éléments virtuels. La technique utilisée repose sur l'usage d'un stroboscope synchronisé à une caméra témoin. Grâce au flash généré par le stroboscope, le premier plan reçoit une quantité importante de lumière tandis que l'arrière plan reste sombre. Par seuillage de l'intensité lumineuse dans l'image issue de la caméra témoin, il est possible d'identifier le premier plan. Cette segmentation est appliquée à l'image de la caméra de tournage. Le stroboscope n'impacte pas la caméra de tournage puisque la fréquence des flashs est en opposition de phase avec la fréquence d'acquisition des images. Récemment, plusieurs autres projets financées par la communauté européenne s'intéressent à la prévisualisation pour le cinéma. On peut citer le projet *Popart* [12] ainsi que le projet *Dreamspace* [13].

11. La réalité augmentée consiste à ajouter un élément virtuel à une prise de vue réelle en temps réel. La *previz* est de la réalité augmentée
12. `http://www.popartproject.eu/`
13. `http://www.dreamspaceproject.eu/`

1.3.2 Problématique de la thèse au sein du projet

Le rôle du *GIPSA-lab* au sein de ce projet, intiment lié au sujet de recherche de cette thèse, est de proposer un système de *compositing* en temps réel qui gère les occultations réelles-virtuelles. Il faut déterminer quels éléments réels ou virtuels seront au premier plan, lesquels seront au second plan et ainsi de suite. Autrement dit, le but est d'améliorer la qualité de la composition en proposant une gestion correcte de la hiérarchie des calques (voir section 1.2.3). Pour atteindre cet objectif, il est nécessaire de connaître la position relative des éléments réels vis-à-vis des éléments virtuels. Le *Gipsa-lab* propose d'utiliser la notion de profondeur. Il s'agit de connaître la distance à la caméra pour chaque objet réel ou virtuel.

Ainsi l'objet de ce travail concerne le développement et l'expérimentation d'une méthode de *compositing* basée sur la profondeur. La recherche se structure autour de trois problématiques structurantes, à savoir la distance des objets à la caméra, le temps réel, et la qualité du mélange.

1.4 *Compositing* basé sur la profondeur

1.4.1 Mesure de la distance des objets à la caméra

On suppose connue la position de la caméra virtuelle dans le moteur de jeux vidéo. L'utilisation d'un système de *camera tracking* permet d'obtenir cette information. La profondeur des objets virtuels par rapport à la caméra virtuelle est alors définie.

Pour la scène réelle nous disposons simplement de l'image réelle. La distance des objets réels à la caméra film est encore inconnue. Cette information est déterminante pour distinguer les éléments qui sont au premier plan et ceux qui sont à l'arrière plan. Dans notre cas, nous avons choisi d'utiliser un capteur de profondeur fixé de façon rigide à la caméra film. Nous désignons ce capteur par *caméra de profondeur*. Celle-ci nous permet d'évaluer la distance des objets qu'elle observe. Il est possible d'obtenir une représentation géométrique des éléments de la scène réelle qui est filmée, comprenant les acteurs et le décor, en temps réel. Les différents systèmes qui existent pour obtenir la profondeur d'une scène sont abordés dans le chapitre suivant.

L'utilisation de la caméra de profondeur ne se réduit pas à la segmentation de l'image réelle afin de générer des calques qui seront mélangés avec les calques CGI. Le système dépasse cette opération puisqu'il permet de définir une profondeur pour chaque pixel de l'image réelle. En effet, la logique de calque présente une limite : chaque élément réel ou virtuel reste prisonnier de sa couche. Il s'agit d'une hiérarchie figée. Prenons l'exemple d'un acteur situé sur une couche réelle définie au second plan derrière une couche virtuelle où se trouve une créature virtuelle définie au premier plan. L'acteur aura beau se rapprocher de la caméra film, il apparaîtra toujours au second plan, même si physiquement il devrait se situer devant la créature virtuelle. Une fois la hiérarchie figée, la logique de couche ne tient pas compte de la distance physique des éléments à la caméra. La méthode proposée passe d'un modèle de *layers* hiérarchisés à un modèle de profondeur par pixel.

Désormais, pour chaque pixel de l'image réelle, sont encodées la couleur de l'objet observé et sa distance à la caméra. Il est possible d'obtenir une image de synthèse avec les mêmes caractéristiques, c'est-à-dire la couleur des éléments virtuels et la distance de ceux-ci à la caméra virtuelle. Le moteur de rendu est capable de générer ces informations.

Le but du *compositing* basé sur la profondeur est d'obtenir une image finale avec une gestion correcte des occultations réelles-virtuelles. On appelle cette image l'image mélangée. Gérer les occultations consiste à décider pour chaque pixel de l'image mélangée, si la couleur provient de l'image réelle ou si elle provient de l'image de synthèse. Ce choix est fait en comparant la profondeur respective de chacun des pixels mis en jeux. Si la profondeur du pixel issu de l'image réelle est plus proche de la caméra que la profondeur du pixel issu de l'image de synthèse alors il sera affiché dans l'image finale. Inversement, si la profondeur du pixel de l'image de synthèse est inférieure à celle du pixel de l'image couleur alors c'est la couleur de l'image de synthèse qui est affichée. Cette technique permet d'obtenir l'image résultant du mélange entre réel et virtuel.

1.4.2 Contraintes du temps réel

Pour la méthode à mettre en place, l'un des enjeux déterminant reste la gestion du temps réel. La définition du temps réel que l'on utilise ici signifie que le temps nécessaire pour produire une image ne doit pas excéder l'intervalle entre deux images successives de la caméra film. Pour une caméra fonctionnant à 30 images par seconde, ce temps équivaut à 16.6 millisecondes. Le temps de calcul alloué à nos algorithmes ne doit pas excéder cette limite. Notre objectif est de démontrer que l'utilisation d'un moteur de jeux vidéo est adapté à cette problématique.

En effet un moteur de jeux vidéo se doit par nature de fonctionner en temps réel. Un moteur de jeu intègre un moteur de rendu, mais également des composants tel qu'un moteur physique pour gérer les collisions entre deux objets virtuels par exemple. Tous les composants sont conçus pour délivrer un résultat dans un laps de temps inférieur à l'intervalle entre deux images. L'objectif est donc de conformer nos algorithmes à ce formalisme pour profiter de l'infrastructure déjà mise en place.

1.4.3 Qualité du mélange

Le but recherché est d'obtenir une image mélangée de haute qualité. Autrement dit, nous cherchons à obtenir une image qui détient la même résolution que la caméra film et dont les occultations sont gérées avec fiabilité. Cependant, les limites du matériel posent problème. En effet la résolution actuelle des systèmes de captation de profondeur temps réel n'égale pas la résolution des caméras de tournage. La caméra de tournage utilisée pour ce travail dispose d'une résolution dîte HD [14] soit 1920×1080 pixels. Le capteur de profondeur utilisé, quant à lui, fonctionne sur le principe du ToF [15] avec une résolution de 512×424 pixels.

A cela s'ajoute la présence d'une zone d'ombre qui apparaît lorsqu'on visualise la profondeur du point de vue de la caméra film. Cette ombre est due au léger décalage spatial entre la caméra film et la caméra de profondeur. Dans cette zone, aucune information de profondeur n'est disponible. Il se peut également que le capteur ne puisse pas donner de mesure de profondeur si certains matériaux de la scène absorbent les ondes infrarouges. Un état de l'art des méthodes d'*inpainting* est exposé dans l'annexe C afin d'obstruer les trous de la carte de profondeur.

Enfin, une imprécision de mesure inhérente à la technologie utilisée, apparaît notamment aux abords des contours des objets. Pour obtenir un résultat satisfaisant, les contours des objets doivent être correctement délimités. En effet, un alignement irrégulier entre la couleur et la profondeur dans l'image réelle est rapidement décelable et produit une sensation désagréable pour l'utilisateur. La méthode développée pour affiner correctement les contours s'appuie sur du filtrage guidé. Cette méthode est décrite dans le chapitre 3.

14. *Haute Définition*
15. *Time of Flight*

Nous posons dans cette section les éléments déterminants liés à notre problématique. Le choix de la technologie utilisée pour mesurer la distance entre les objets réels et la caméra film est expliqué dans le prochain chapitre. Nous décrirons le système matériel mis en place pour cette thèse. Dans ce même chapitre nous détaillerons également la gestion du temps réel via l'intégration au moteur de jeux vidéo. Enfin le dernier chapitre sera une évaluation sur la qualité du mélange obtenu et une proposition sur les façons de l'améliorer en temps réel.

Chapitre 2

Système

Sommaire

2.1 Capture de profondeur

Comme nous l'avons développé dans le chapitre 1, la problématique liée à ce travail de recherche rend indispensable le recours à une modélisation numérique de la scène que l'on est en train de filmer. Cette modélisation, comparée aux éléments de synthèse qui seront ajoutés, permettra de déduire les occultations entre éléments réels et virtuels. Dans le cadre de notre travail, il a été décidé d'utiliser un seul capteur couplé à la caméra film. Ce capteur permet l'acquisition de la profondeur de la scène. La profondeur, associée aux paramètres intrinsèques du capteur, permet de reconstituer une modélisation 3D de la scène. Le but de cette section est de présenter les différentes techniques optiques de l'état de l'art qui permettent l'acquisition de la profondeur d'un environnement en temps réel. Une vue d'ensemble des techniques existantes est présentée dans la figure 2.1.

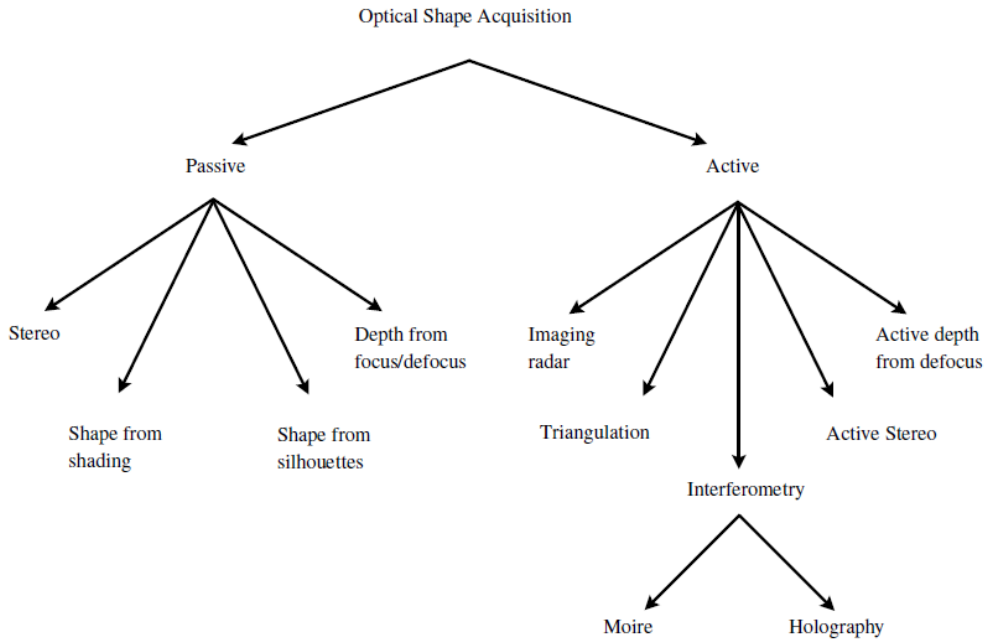


FIGURE 2.1 – Vue d'ensemble des techniques optiques pour l'acquisition de la profondeur d'une scène (d'après [CS00]).

Les sous-sections qui suivent ne constituent pas une liste exhaustive. Seules les techniques considérées pertinentes aujourd'hui sont présentées. Certains dispositifs utilisent un seul capteur ou un ensemble de capteurs optiques traditionnels tandis que d'autres vont modifier la construction de ces capteurs afin de les adapter à une application. On parle alors de *computationnal photography*.

2.1.1 Stéréoscopie passive

L'être humain a deux yeux qui lui permettent d'apprécier les distances des objets qui l'entourent. La présence de ce système de vision stéréo n'est pas le seul facteur de la perception de la profondeur mais néanmoins un outil important. Afin de reproduire ce fonctionnement on utilise deux caméras optiques séparées par une distance appelée *baseline*. Un point de l'espace observé par un système stéréoscopique est projeté à un endroit distinct dans chacune des deux images puisque le point de vue de chaque caméra est différent (voir figure 2.2). Le décalage de projection du point dans chacune des deux images s'appelle la disparité. Plus le point observé est lointain plus la disparité sera faible. Inversement, un point de l'espace qui est très proche des caméras donne lieu à une disparité importante. On définit une des deux caméras du système stéréoscopique comme la caméra maître. La caméra maître définit le repère dans lequel seront exprimées les coordonnées des points. A partir de la disparité, et en connaissant les paramètres de projection de la caméra, il est possible de retrouver les coordonnées du point de l'espace dans le référentiel de la caméra maître. Cela signifie que ce qui s'applique à un point de l'espace peut s'appliquer à tous les points visibles par l'ensemble des caméras, pour ainsi reconstruire en 3D la scène observée.

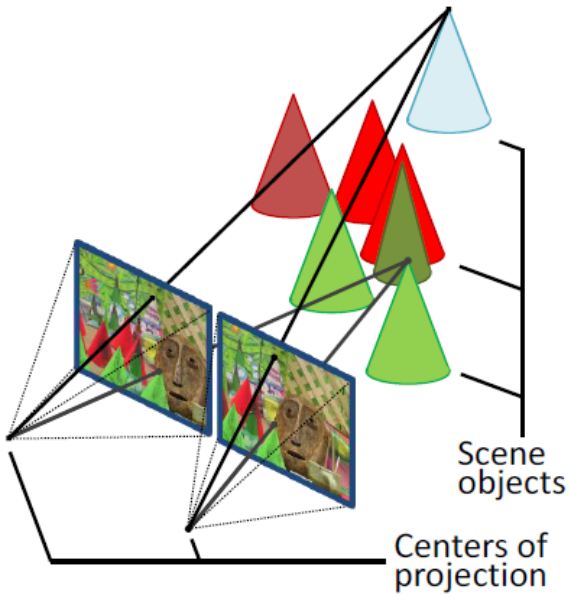


FIGURE 2.2 – illustration du principe de la stéréoscopie, ici deux caméras observent une même scène, les objets sont identifiés dans chacun des points de vue. Une triangulation permet de retrouver la profondeur. (source `http://www.adept.net.au/news/newsletter/201211-nov/article_3D_stereo.shtml`)

La stéréoscopie requiert un minimum de deux caméras mais le principe peut être étendu à un nombre plus important de capteurs. Afin de calculer la profondeur de chaque pixel de l'image de la caméra maître plusieurs étapes sont nécessaires. La première étape consiste à

repérer dans chacune des images produites par les caméras les motifs en commun. Il s'agit d'apparier un point dans la première image avec un point dans la seconde image. La géométrie épipolaire permet de limiter l'appariement d'un point dans l'image 1 à la recherche d'un point le long d'une droite dans l'image 2. Il est nécessaire pour cela de connaître les paramètres intrinsèques et extrinsèques de chaque caméra. Ces paramètres s'acquièrent via une étape de calibration. Une explication détaillée de la calibration des caméras est donnée dans la section 2.3. Il est bien souvent nécessaire de rectifier les images avant d'effectuer l'appariement. La rectification permet de rendre toutes les lignes épipolaires parallèles et alignées horizontalement, c'est-à-dire que l'association d'un point de l'image 1 se fera avec un point situé dans la même ligne dans l'image 2. Les lignes épipolaires sont alignées avec les lignes des pixels de l'image. La rectification des images permet de simuler un système stéréoscopique où les axes optiques des caméras sont parallèles entre eux. La transformation euclidienne entre les référentiels de chaque caméra est une translation pure, au sein de laquelle la rotation est supprimée.

L'appariement est une étape importante. Une première famille de méthodes s'appuie sur des descripteurs capables de rechercher et d'identifier les points d'intérêts dans une image [VL01]. Les points similaires sont associés afin de calculer la disparité. La seconde grande famille de méthodes est la mise en correspondance de blocs, basée sur la corrélation. Cette dernière consiste à évaluer une fonction de similarité [FHM$^+$93] dans les zones bien texturées. Cette méthode est locale puisqu'elle cherche à déterminer le maximum de la fonction dans une fenêtre centrée sur le point considéré. Elle a l'avantage d'être rapide, facilement parallélisable et donc bien adaptée au temps réel. Cependant, les méthodes citées, basées sur les descripteurs ou la corrélation, ont l'inconvénient de produire des cartes de profondeur éparses. D'autres procédés dits denses permettent de calculer en chaque pixel de l'image une valeur de disparité. Ils font le postulat que la disparité est continue entre des pixels voisins. L'opérateur Laplacien est par exemple utilisé par [Hor86] afin de garantir des variations douces dans la carte de profondeur.

Parmi les capteurs stéréoscopiques disponibles sur le marché nous pouvons référencer la caméra ZED de *Stereolabs*. Le matériel est dimensionné pour des scènes de 1 à 15 mètres de profondeur. La société accompagne sa caméra d'un puissant algorithme tirant partie du GPU et notamment de l'architecture CUDA afin de fonctionner en temps réel. On peut observer sur la figure 2.3 un exemple de carte de profondeur obtenue. Il est intéressant de noter qu'il n'y a pas de trou dans la carte de disparité (carte dense). Pourtant certaines zones sont erronées, notamment la zone au dessus du bras gauche. On attend normalement une différence marquée de la disparité entre le bras et le mur à l'arrière plan. En l'absence de texture, l'algorithme utilisé impose un critère de continuité.

On peut également référencer la société *Pelican Imaging* qui conçoit un réseau de caméras dans un espace réduit afin de pouvoir équiper les téléphones portables (figure 2.4). Il s'agit ici de 16 mini-caméras séparées par une très faible *baseline*. Ce système qui s'appuie sur les principes de la stéréoscopie a pour but d'offrir de nouvelles applications aux *smartphones* : scans d'objets, édition de la profondeur de champs d'une photo a posteriori, mesure de distance physique entre deux points d'une image etc...

Ainsi la stéréo fonctionne en intérieur ou extérieur. Elle ne nécessite pas d'éclairage supplé-

FIGURE 2.3 – Profondeur d'une scène acquise en temps réel par la caméra ZED de *Stereolabs*.

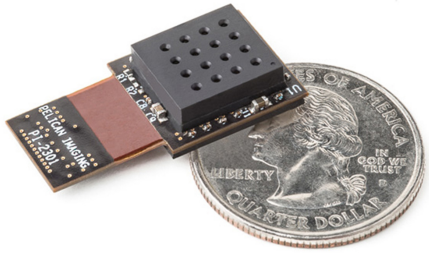


FIGURE 2.4 – Matrice de caméras développée par la société *Pelican Imaging* afin d'équiper les téléphones portables.

mentaire à la lumière naturelle. Sa consommation en énergie est donc faible. Matériellement, la distance entre les optiques va définir l'échelle des mesures possibles. Une *baseline* importante permet de fournir une mesure précise de la distance d'un objet lointain. Malheureusement lorsque la scène n'est pas texturée il n'est pas possible de calculer une disparité. Une solution utilisée dans [ZNI+14] consiste à projeter des motifs sur la scène afin de créer de la disparité pour les caméras stéréo. Cette technique est dite active et se rapproche des méthodes que nous allons voir dans la sous-section suivante.

2.1.2 Lumière structurée

L'un des inconvénients majeurs de la stéréoscopie décrite dans la section précédente 2.1.1 est qu'il n'est pas possible de calculer une mesure de profondeur lorsqu'il n'y a pas de texture dans la scène observée. La technique de la lumière structurée permet de surmonter ce problème. Cette méthode consiste à associer une caméra à un projecteur de lumière. On parle alors de stéréo active en opposition à la méthode de stéréoscopie dite passive abordée dans la

sous-section précédente. Une image connue est projetée sur la scène et la caméra observe la scène éclairée par l'image projetée. Dans la vue de la caméra on va apparier chaque position des motifs observés avec leur position dans l'image projetée. Les paramètres de la caméra et du projecteur sont connus (matrices intrinsèques) ainsi que leurs positions relatives (matrices extrinsèques).Il est alors possible de déterminer la profondeur du point observé par triangulation.

Une méthode décrite dans [RHHL02] consiste à projeter un cycle d'images sur la scène. Ces images sont composées de bandes, colonnes noires ou blanches, comme on peut le voir sur la figure 2.5. L'enchaînement des images au cours du cycle temporel, définit une signature unique pour chaque colonne. L'image issue de la caméra est rectifiée de façon à avoir une association ligne à ligne. C'est-à-dire qu'un pixel ayant pour coordonnées (i, j) dans le cycle d'images de projection sera forcément à la i^{me} ligne de l'image de la caméra (s'il n'y a pas d'occultation). En observant les variations d'illumination d'un pixel dans la caméra au cours du cycle temporel de projection on peut alors déterminer la colonne à laquelle ce pixel est associé. Il est ainsi possible de faire l'appariement entre les pixels projetés et les pixels observés. Cette méthode a une limite. Pour calculer une mesure de la profondeur il faut projeter tout le cycle d'images nécessaires à identifier chaque colonne. Avec un code binaire, un cycle de k images permet de coder 2^k colonnes. Il faut donc que le projecteur et la caméra traitent k images pour obtenir une carte de profondeur. Des optimisations permettent de réduire le nombre d'images par cycle. [LSP06] utilise la couleur, et [SS03] utilise un codage de la lumière avec des fonctions sinus.

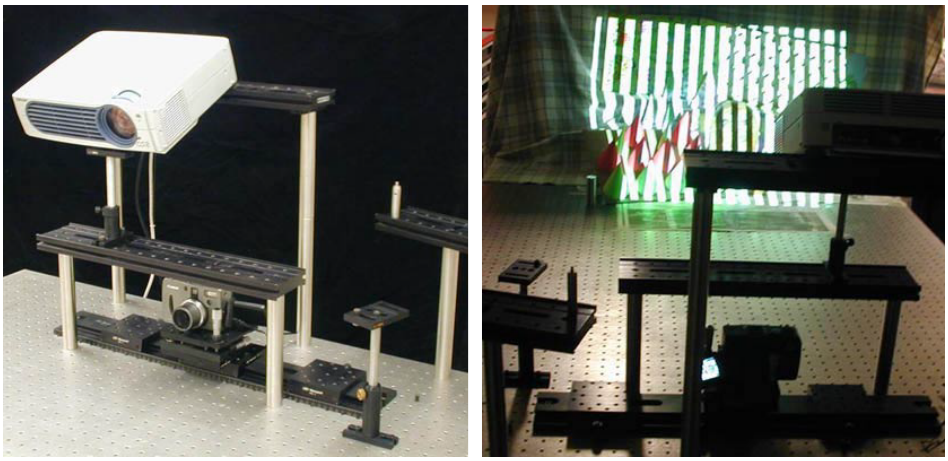


FIGURE 2.5 – Le principe de la lumière structurée requiert un projecteur et une caméra. Le projecteur éclaire la scène d'une image aux motifs connus, ici un cycle temporel d'images composées de bandes. On va rechercher ces motifs dans l'image issue de la caméra qui observe la scène. Une fois l'appariement effectué, une triangulation permet de retrouver la profondeur de la scène observée. Image tirée de [SS03].

L'inconvénient de l'utilisation d'un projecteur dans le spectre du visible est que celui-ci peut éblouir les personnes présentes dans la scène. De plus ce dispositif actif altère la captation des couleurs de la scène. Une solution consiste à projeter dans un spectre différent du visible tel

que le proche infrarouge. Un projecteur et une caméra fonctionnant dans cette longueur d'onde sont utilisés pour capturer la profondeur de la scène, tandis qu'une caméra classique capture la couleur. C'est le cas du dispositif *Kinect* de première génération (*Kinect 1*) popularisé par *Microsoft* en 2010 (figure 2.6). Cet appareil utilise la technologie *Primesense*[1] décrite dans le brevet [SZ13]. Cet appareil est équipé d'un projecteur infrarouge laser qui projette sur la scène une image infrarouge tachetée. La scène est observée à l'aide d'une caméra, sensible uniquement à l'infrarouge. La technologie utilisée par ce dispositif ne nécessite pas un cycle d'images. La même image est projetée en continu. Lors de sa production, l'appareil enregistre dans sa mémoire interne l'image du motif de projection sur un mur blanc situé à 1 mètre de distance [ZZ14]. Lors de la mesure de profondeur d'une scène inconnue, on compare alors l'image observée par la caméra infrarouge et l'image de référence. Le processeur interne du dispositif, détermine les appariements de motifs et délivre une carte de profondeur. Enfin, une caméra sensible uniquement à la lumière visible capture la couleur. Il est possible par une commande logicielle d'activer l'alignement entre l'image de profondeur et l'image couleur. A l'heure actuelle, cet appareil est très largement utilisé.

FIGURE 2.6 – *Kinect 1* de *Microsoft* qui utilise la technologie *Primesense* de lumière structurée. La *Kinect 1* est capable d'acquérir une carte de profondeur d'une résolution de 640×480 pixels avec une fréquence de $30Hz$.

La société *Primesense* a mis au point en 2013 une architecture miniature de son capteur, le module *Capri*. La société *Structure*[2] a développé autour de ce module un capteur 3D pour appareils mobiles tel que l'*iPad*. Lancé officiellement en 2014, la société met régulièrement à jour un kit de développement. Les applications annoncées concernent la mesure des dimensions d'objets dans la scène observée, la capture de la géométrie d'objets (scanner 3D), la réalité augmentée avec prise en compte de l'environnement pour gérer les occultations, et la physique, c'est-à-dire gestion de la gravité et des collisions des objets virtuels. La figure 2.7 donne un aperçu de ce capteur.

L'avantage majeur des dispositifs à lumière structurée est que ceux-ci génèrent eux-mêmes la texture nécessaire à la mesure de la disparité. L'inconvénient est parfois une vitesse d'acquisition limitée pour les dispositifs qui nécessitent un cycle d'images avant d'obtenir une mesure de la profondeur. La *Kinect 1* n'a pas ce problème puisqu'une seule image est projetée, le capteur délivre une carte de profondeur à $30Hz$. Tous ces dispositifs sont sensibles aux variations d'éclairage. En extérieur, le soleil émet également dans le proche infrarouge, ce qui peut saturer la caméra du module *Kinect 1* et rendre impossible la mesure de profondeur. Il faut donc un environnement contrôlé.

1. La société *Primesense* a été acheté par *Apple* fin 2013
2. `http://structure.io/`

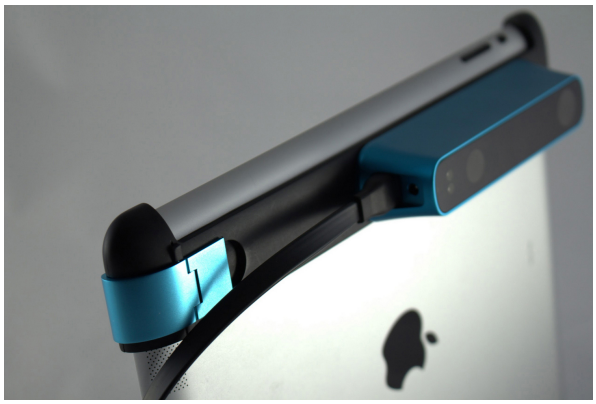


FIGURE 2.7 – Capteur *Occipital* de la société *Structure* basé sur la technologie de lumière structurée de la société *Primesense*. Le capteur est ici monté sur un appareil mobile *iPad*.

2.1.3 Dispositifs à temps de vol

Les capteurs de profondeur à temps de vol, dits ToF, sont largement répandus et fournissent des cartes de profondeur en temps réel.

Une caméra ToF envoie une onde sur la scène et mesure le temps que celle-ci prend pour faire un aller retour de la caméra à l'objet. La vitesse de l'onde c étant connue il est alors possible de déterminer la profondeur à laquelle se trouve l'objet par rapport à la scène. Ce procédé est similaire au principe du LIDAR.

Le signal lumineux est fourni par une diode (ou un réseau de diodes). Ce signal est modulé par une fonction sinusoïdale. La fréquence de modulation f_{mod} est choisie de telle sorte à empêcher toute ambiguïté sur la valeur de profondeur calculée en fonction de la plage de détection souhaitée. En effet connaissant la vitesse d'une onde lumineuse, on détermine que la distance maximale d_{max} détectable par le système vaut :

$$d_{max} = \frac{c}{2f_{mod}} \tag{2.1}$$

Si l'échelle de la scène augmente il faut diminuer la fréquence de modulation. Pratiquement, les équipements ToF utilisent une fréquence de modulation de l'ordre de $20MHz$. Ce qui permet une plage de mesure de 0 à $7.5m$.

Ce qui est détecté par le capteur en chaque photosite est le décalage de phase entre la lumière émise et la lumière reçue, noté Φ_d. La distance d (en mètre) est déterminée de façon matérielle, c'est une fonction proportionnelle au décalage de phase Φ_d. L'équation suivante donne le détail du calcul :

$$d = \frac{c}{2f_{mod}} \frac{\Phi_d}{2\pi} \tag{2.2}$$

Le capteur ToF est constitué d'une matrice de photosites qui convertit la quantité lumineuse en valeurs numériques, et enregistre directement le retard entre l'onde émise et la réception du signal Φ_d. Cela nécessite un encombrement plus important au niveau de chaque photosite, ce qui explique les plus faibles résolutions de ces capteurs, comparés aux systèmes optiques classiques. Un exemple de capteur, fabriqué par l'entreprise *PMD*, est présenté sur la figure 2.8. On aperçoit clairement l'émetteur qui module la lumière à gauche, et le récepteur à droite.

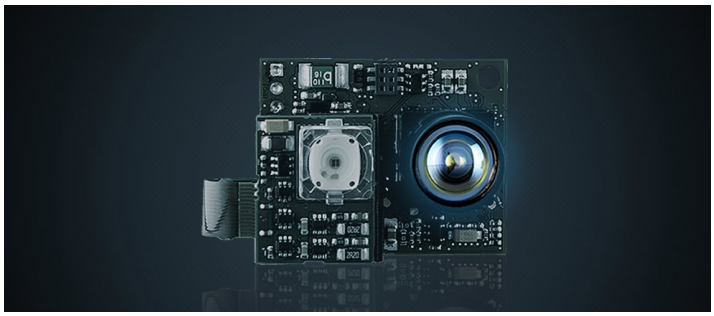


FIGURE 2.8 – Caméra à temps de vol (ToF)de la société *PMD*, la *camboard nano*

La société *ARRI*, fabricant reconnu de caméras dans le monde de l'audiovisuel, a récemment mis au point un prototype nommé *ALEXA SCENE*[3]. Ce prototype est une caméra couplant un capteur vidéo avec une caméra de profondeur à temps de vol. Le flux de données issu de cette caméra, couramment appelé RGBD[4], c'est-à-dire une image couleur *RGB* et une image de profondeur *D*, est synchronisé à la fois spatialement, et temporellement. La figure 2.9 montre le prototype.

FIGURE 2.9 – Prototype d'une caméra professionnelle vidéo et profondeur RGBD, utilisant la technologie à temps de vol. Ce prototype est baptisé *ARRI ALEXA SCENE* (source `http://www.arri.com/news/prototype-motion-scene-camera/`).

En 2014, la société *Microsoft* a fait évoluer le module *Kinect* en abandonnant la technologie de lumière structurée pour adopter un capteur à temps de vol. C'est ce capteur que nous

3. `https://www.arri.de/news/news/prototype-motion-scene-camera/`
4. *Red Greeen Blue Depth*

utilisons dans cette thèse. Le choix de ce capteur est expliqué dans la section 2.2. Il s'agit également de la technologie ToF, choisie par *Google* pour être intégrée dans le projet *Tango*[5], par l'intermédiaire d'un capteur fabriqué par *PMD*. Le capteur est utilisé dans la tablette du kit de développement fourni par *Google*.

L'avantage des capteurs ToF est un débit élevé d'images de profondeur. L'inconvénient majeur est la faible définition des cartes obtenues.

5. Le projet *Tango* promu par *Google* vise à développer un capteur léger capable de se localiser dans un environnement et de capter la géométrie d'une scène dans le but de réaliser des applications de réalité augmentée pour les *smartphones*. `https://get.google.com/tango/`

2.2 Choix du capteur

Dans le cadre de notre travail, les premiers tests ont été réalisés avec la version 1 de la *Kinect*, un capteur de profondeur basé sur la lumière structurée (cf. 2.1.2). Le choix a été porté sur ce type de capteur car il présente l'avantage de pouvoir fournir une carte de profondeur en temps réel, même si la zone n'est pas texturée contrairement à des dispositifs stéréo. En outre le prix de ce capteur le rend très abordable. Quelques mois après le début de la thèse (2014) le capteur à temps de vol *Kinect 2* est arrivé sur le marché. Pour un coût similaire, il offre des performances plus adaptées à nos besoins d'application.

Dans la suite de cette section, nous retraçons l'évolution des capteurs RGBD depuis la *Kinect 1*. En parallèle, nous revenons sur le développement des SDK [6] associés qui ont contribué au succès de ces capteurs. Nous utilisons des fonctionnalités présentes dans l'un de ces SDK au cours de cette thèse.

2.2.1 État de l'art des fonctionnalités du capteur *Kinect*

Le véritable essor des capteurs de profondeur est sans nul doute associé au capteur *Kinect* (voir figure 2.6) de l'américain *Microsoft* en 2010. Afin d'augmenter l'attractivité de sa console de jeux *Xbox 360*, *Microsoft* était à la recherche d'un capteur pour interagir entre l'utilisateur et le contenu visuel. La solution à cette problématique a été développée par la société Israélienne *Primesense*. En effet, cette société a mis en place une puce électronique (voir figure 2.10) capable de gérer la technologie *Structured Light* [SZ13] décrite dans la section 2.1.2.

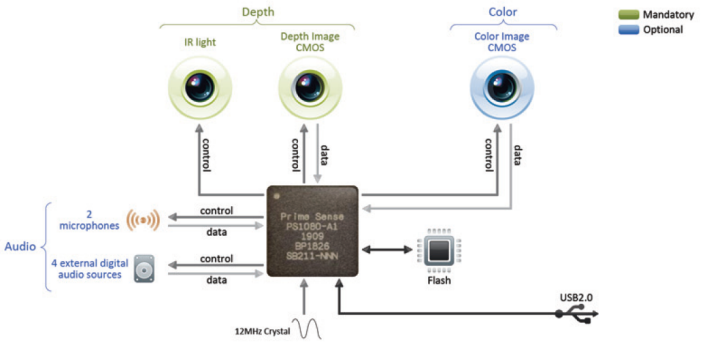


FIGURE 2.10 – Design de la puce *Primesense* équipant la majorité des capteurs RGBD à lumière structurée bon marché (source `https://github.com/adafruit/Kinect/blob/master/FMF_2.PDF`).

Le capteur de *Microsoft* a été un succès sans précédent. La première version est uniquement commercialisée avec la console de jeux *Xbox 360* à partir de novembre 2010. Cette limitation conduit une communauté d'utilisateurs, baptisée *OpenKinect*, à créer un *driver opensource* multiplateformes appelé *libfreenect* afin de pouvoir utiliser le capteur sur un ordinateur. En

6. *Software Development Kit*

février 2012 *Microsoft* commercialise officiellement sa *Kinect 1* avec un SDK pour utiliser tout le potentiel du capteur sur n'importe quel ordinateur utilisant le système d'exploitation *Windows*.

En novembre 2010, un consortium est créé afin de promouvoir l'utilisation des capteurs RGBD et les interactions hommes machines. Ce consortium s'appelle *OpenNI* [Ope10] et propose un SDK universel afin d'exploiter les capteurs RGBD compatibles *OpenNI*. Le consortium regroupe des sociétés comme *Primesense* ou *Asus*. *Microsoft* n'en fait pas partie. La société *Asus* a mis en vente son capteur *Asus Xtion* compatible *OpenNI* qui est équipé d'une puce *Primesense* en avril 2012. Plus tard, la société *Primsense* s'est également mise à commercialiser son capteur appelé *Carmine* (très semblable en terme de design à celui de *Asus*) (voir figure 2.11). La société *Primsense* a également mis à disposition des développeurs au sein du consortium *OpenNI* un module logiciel appelé *Nite* pour la détection de personnes, et la capture de mouvement. Il existe aussi un *driver* non-officiel pour utiliser la *Kinect 1* avec le *framework OpenNI*.

FIGURE 2.11 – Capteur *Carmine* de *Primesense*. Celui-ci est équipée de la même puce que la *Kinect 1* et possède des caractéristiques similaires.

Pour résumer, le début des capteurs RGBD, ceux-ci fonctionnent tous avec une technologie de lumière structurée développée par *Primsesense*. Pour accéder aux données du capteur il existe trois SDK : celui de *Microsoft*, celui de *OpenNI* et celui de *Libfreenect*. Une surcouche logicielle est disponible dans le SDK *Microsoft*, et dans le module *Nite* qui fonctionne de paire avec *OpenNI*, pour l'analyse des données et la détection de personnes (capture de mouvements).

Pour le lancement de sa nouvelle console *Xbox One* en novembre 2013, *Microsoft* rompt son partenariat avec *Primesense* et conçoit désormais en interne le *hardware* et le *software* de son capteur RGBD. Cette compétence est à rapprocher du rachat par *Microsoft* en 2010 de la société *Canesta* fabricant de capteurs à temps de vol. Ainsi le nouveau capteur *Kinect 2* (voir figure 2.12) n'utilise plus de puces *Primesence* mais une technologie propriétaire ToF.

De son côté la société *Primesense* est achetée par *Apple* en novembre 2013. Il n'est alors plus possible d'acheter le capteur *Carmine* fabriqué par *Primesense*. Néanmoins il semble encore possible d'acheter l'équivalent fabriqué par *Asus*, le capteur *Xtion*. La société *Occipital* commercialise toujours le capteur *Structure* basé sur la puce *Capri* de *Primesense* (voir figure 2.7). C'est d'ailleurs la société *Occipital* qui maintient le code du framework *OpenNI*.

FIGURE 2.12 – *Kinect2* de *Microsoft* utilisant la technologie *Time of Flight*. La *Kinect 2* est capable d'acquérir simultanément un flux vidéo HD et une carte de profondeur d'une résolution de 512×424 pixels avec une fréquence de $30Hz$. Ce produit a été lancé en novembre 2013.

2.2.2 Analyse logicielle développée par les constructeurs

Comme nous l'avons précisé précédemment il existe 3 SDK : *Microsoft*, *OpenNI* et *Libfreenect*. Aujourd'hui seulement les deux premiers proposent des modules d'analyse des images RGBD.

Le SDK *Libfreenect*, dont il existe une version pour la *Kinect 1* et une version pour la *Kinect 2*, est multiplateformes. Il permet d'accéder aux différents flux du capteur *Kinect* que sont la couleur, la profondeur et l'infrarouge. Il est possible de recaler l'image de profondeur avec l'image couleur. Cependant il n'intègre pas encore de modules de détection de personnes par exemple. Les outils d'analyse sont sur la *roadmap* de la communauté *OpenKinect*, mais ceux-ci ne sont pas encore prêts.

De son côté, *OpenNI* propose un SDK de base pour accéder aux flux vidéos et profondeur des capteurs compatibles. Il est également possible de télécharger sur le site *OpenNI* un ensemble de blocs logiciels appelés *middleware* (gratuits ou payants). Ces *middleware* s'insèrent dans le *framework* entre le SDK de base et l'application finale pour ajouter rapidement des fonctionnalités. Le plus populaire est certainement le *middleware Nite* développé par *Primesense*. Ce module permet d'identifier des personnes dans une scène, non seulement les silhouettes mais également les squelettes, c'est-à-dire un ensemble d'articulations caractéristiques (voir figure 2.13). Il existe d'autres *middleware* pour l'analyse précise des gestes de la main, des outils pour le scan d'objets ou encore la reconnaissance de visage.

Microsoft, quant à lui, propose deux versions de son SDK pour ses *Kinect 1* et *2*. Ceux-ci permettent d'acquérir les données du capteur et également d'analyser les données. On retrouve un module similaire à *Nite* pour la détection de personnes et la capture de mouvements. Il y a également un module de détection des expressions du visage et du suivi de la tête. Il met aussi à disposition une implémentation de *KinectFusion* [IKH+11] qui permet de scanner des objets en agglomérant un flux de données RGBD et de déterminer la pose de la caméra *Kinect* (voir figure 2.14). Enfin on trouve des outils pour la substitution de l'arrière plan en le remplaçant par une autre image. Il existe également de nombreuses fonctionnalités liées à la reconnaissance vocale, dont il n'est pas fait état ici.

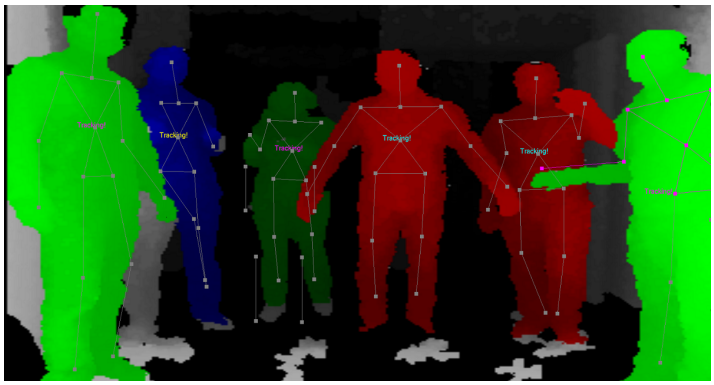


FIGURE 2.13 – Segmentation en temps réel de personnes avec le *framework OpenNI* et le module *Nite*. La silhouette de chaque utilisateur est détectée mais également la posture représentée par un squelette d'articulations caractéristiques (source [Pri10]).

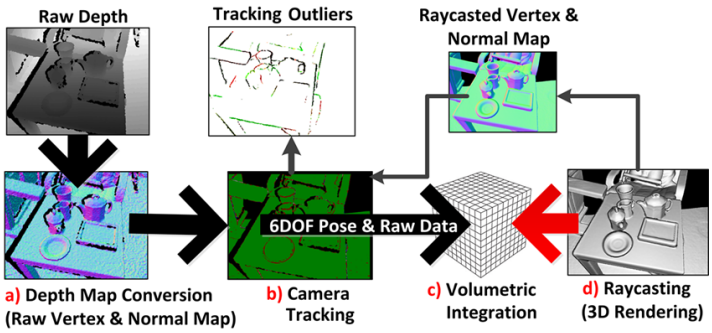


FIGURE 2.14 – Schéma de l'algorithme *KinectFusion* développé par *Microsoft* et disponible dans le SDK de la *Kinect 2*. Image tirée de [IKH+11].

Un important domaine de recherche concerne la détection de personnes dans une image de profondeur [PGKT10]. On trouve d'un côté le module *Nite* (développé par *Primesense*) pour les capteurs compatibles *OpenNI* et de l'autre le *framework Microsoft* pour les capteurs *Kinect*. Les deux *frameworks* détectent plusieurs personnes simultanément (jusqu'à 6 personnes) dans une scène et produisent un masque 2D pour chacune d'elles en temps réel. En outre, les deux méthodes permettent d'associer en temps réel un squelette à chaque silhouette détectée. Les deux modules sont relativement équivalents. Si l'on n'a pas de certitude sur l'algorithme implémenté pour *Nite*, l'algorithme de *Microsoft* est basé sur [SSK+13] : dans cette méthode le corps humain est découpé en 31 zones. L'algorithme va essayer d'associer chaque pixel de l'image de profondeur à l'une des zones composant le corps humain. Une importante base de données a été réalisée en *motion capture* traditionnelle afin d'obtenir les silhouettes d'un échantillon caractéristique de personnes dans une multitude de positions (de l'ordre de 100 000 images). Cette base de données a permis d'entraîner un classifieur de type *randomized decision forest* [Bre01]. Le résultat du test de chaque pixel à travers le classificateur permet d'identifier à quelle zone du corps le pixel appartient. Une fois tous les pixels classifiés, l'algorithme construit

un squelette du corps humain avec des articulations caractéristiques (voir figure 2.15). Il est intéressant de noter que cet algorithme n'utilise aucune cohérence temporelle. Une seule image suffit à détecter la pose de la personne.

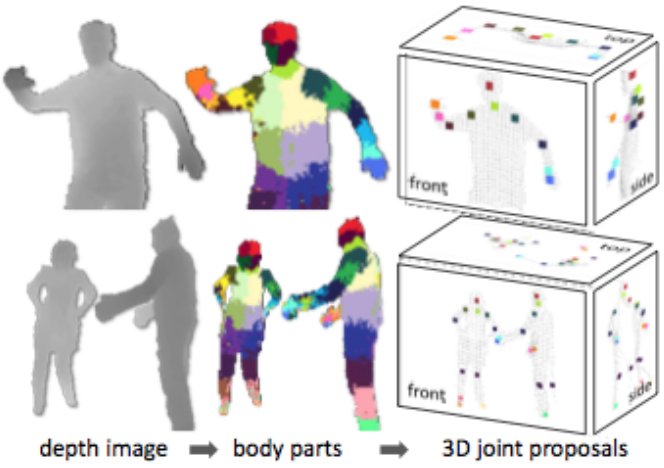


FIGURE 2.15 – Illustration de l'algorithme de détection de silhouette implémenté dans le SDK de la *Kinect*. Chaque pixel est classifié dans une zone du corps humain, puis un squelette d'articulations caractéristiques est déterminé (source [SSK+13]).

2.2.3 Comparaison des *Kinects*

Les deux versions du capteur *Kinect* sont capables de délivrer une carte de profondeur en temps réel et leur prix respectif est abordable. Cela fait de ces capteurs des candidats potentiels pour la conception de notre système. Une comparaison détaillée des performances de ces deux capteurs est faite dans [LMM+15]. Nous avons regroupé dans le tableau 2.1 les caractéristiques principales du capteur *Kinect 1* face au capteur *Kinect 2*.

	Kinect 1	Kinect 2
technologie	lumière structurée	temps de vol
plage de mesure (mètres)	0.8 à 3.5	0.5 à 5
résolution du capteur de profondeur (pixels)	640 × 480	512 × 424
fréquence d'échantillonage (images par seconde)	30	30
angle de vue horizontal (degrés)	57	71
prix (euro)	250	200

Tableau 2.1 – Comparaison de la *Kinect 1* avec la *Kinect 2*

La résolution spatiale de l'image de profondeur de la *Kinect 1* est plus importante que celle de la *Kinect 2*. En effet, la résolution des capteurs ToF est souvent limitée car il faut instrumenter chaque photosite afin de calculer la fonction d'auto-corrélation entre le signal émis et le signal reçu. Cette instrumentation a l'avantage de fournir pour chaque photosite du

capteur *Kinect 2* une mesure de profondeur indépendante. En revanche, ce constat n'est pas valable pour le module *Kinect 1* qui utilise la technologie de lumière structurée. La résolution spatiale supérieure n'est donc pas un argument en sa faveur. A ce titre, le module *Kinect 2* propose un rapport résolution capteur / prix très intéressant.

De plus le bruit de mesure des capteurs de profondeur basés sur la triangulation est une fonction quadratique de la profondeur [ZZ14], comme c'est le cas pour la *Kinect 1*. Tandis qu'avec un capteur ToF on peut considérer que le bruit est indépendant de la distance [SHKZ14]. Cela donne un avantage à l'utilisation de la *Kinect 2*.

Un autre paramètre à prendre en compte est la perturbation par la lumière ambiante. Cette perturbation peut empêcher la puce *Primesense* de faire l'appariement pour calculer la disparité dans la *Kinect 1*. Dans le capteur à temps de vol, c'est le décalage de phase qui est mesuré. La lumière ambiante, à moins d'être modulée, n'affectera donc pas le dispositif. La *Kinect 2* est donc plus robuste aux perturbations liées à la lumière ambiante, ce qui lui donne un autre avantage par rapport à la *Kinect 1*. Bien entendu si la lumière ambiante est trop puissante au point de saturer les capteurs, il ne sera pas possible d'obtenir une carte de profondeur.

Enfin l'angle de vue du capteur *Kinect 2* est plus important que celui du *Kinect 1*. Cela étend la gamme de focale que l'on peut utiliser avec la caméra film tout en ayant le champ de vision complètement couvert par la caméra de profondeur. Il est cependant important de préciser qu'il est préférable d'utiliser une focale sur la caméra film proche de celle du capteur de profondeur afin de bénéficier de toute la résolution du capteur de profondeur.

2.3 Calibration du système

2.3.1 Utilisation d'une caméra professionnelle

L'objectif de notre travail de recherche est d'intégrer des effets spéciaux dans un plan vidéo en temps réel pour le monde de l'audiovisuel et du cinéma. Pour l'acquisition de la vidéo il est possible d'utiliser une *webcam* ou même la caméra couleur de la *Kinect 2*. Cependant, notre objectif est de construire un démonstrateur plus abouti tenant compte des contraintes du milieu. Pour cette raison, pour construire le prototype, nous faisons le choix d'utiliser une caméra professionnelle. C'est cette caméra que nous appelons *caméra film* tout le long de notre propos.

Hormis le fait d'être un élément incontournable d'un plateau de tournage, une caméra de tournage professionnelle présente plusieurs avantages. Tout d'abord, la taille du capteur est généralement plus grande, il est possible de monter différents objectifs et tous les réglages de la caméra sont configurables. D'un point de vue technique, il est possible d'ajuster manuellement la fréquence d'acquisition dans des formats respectant les standards du monde audiovisuel et le débit d'images est stable. Enfin, au niveau de la manipulation et de la prise de vue, les paramètres tels que la vitesse de l'obturateur, le gain et la sensibilité sont réglables.

Notre besoin se définit autour de plusieurs critères. Notre travail rend nécessaire l'utilisation d'une caméra capable de fournir un signal vidéo au format HD (1980×1080) à une fréquence minimum de 30 images par secondes. Nous privilégions la connectique SDI [7] pour la transmission du flux vidéo pour respecter les standards utilisés sur les plateaux de tournage. Pour ces raisons, nous choisissons d'utiliser la caméra *HD Studio* de *Blackmagic* (figure 2.16). Cette caméra est vendue comme une caméra de studio compacte adaptée pour la production en direct. Toutes les connectiques sont professionnelles. Cette caméra a la capacité d'émettre un flux vidéo, et de recevoir un flux vidéo externe de façon simultanée. Il est possible d'afficher sur l'écran le plan vidéo mélangé avec les éléments virtuels, de sorte que la caméra envoie son flux vidéo via un câble SDI à l'ordinateur, qui traite la séquence et renvoie en temps réel un flux vidéo en prenant en compte le *compositing*.

La caméra *Blackmagic* dispose également d'une entrée de référence (ou *genlock*) qui permet de synchroniser le flux vidéo avec d'autres appareils de capture. Ceci est utile dans un studio pour la prise de vue en direct afin que tous les signaux vidéos qui arrivent à la régie soient synchronisés. C'est le cas sur les plateaux de télévision par exemple. Ce mode de fonctionnement est appelé *broadcast* en anglais. Le fonctionnement sur un plateau de tournage de type cinéma est différent. Pour le cinéma, on utilise un appareil de type horloge qui génère et envoie un *timecode* à l'ensemble des appareils de capture (vidéo et son). Chaque flux qui est enregistré grave dans ses métadonnées le *timecode*. Cela permet de synchroniser les médias en post-production.

En ce sens, la caméra *Kinect 2* n'est pas un système professionnel de capture puisqu'il

7. *Serial Digital Interface*

FIGURE 2.16 – Caméra *HD Studio* de *Blackmagic* utilisé pendant la thèse (source `https://www.blackmagicdesign.com/products/blackmagicstudiocamera`).

n'est pas possible de synchroniser le flux d'images de profondeur avec une horloge externe. De plus, tous les paramètres ToF sont gérés automatiquement, de sorte qu'il est impossible de gérer le temps d'intégration du capteur par exemple. D'autres constructeurs fournissent des caméras ToF prenant en compte ces contraintes. On peut citer la *CamCube*[8] de *PMD* ou la *StarForm 3D*[9] de *Odos Imaging*. Ces caméras sont cependant hors budget pour la thèse et n'offrent pas une résolution spatiale aussi bonne que la *Kinect 2*. Le signal de la *Kinect 2* est synchronisé logiciellement avec la caméra film en utilisant le *timestamp* interne accessible par le SDK de *Micrososft*.

2.3.2 Description du système

Le prototype mis en place au sein de notre laboratoire est composé d'une caméra film et du capteur de profondeur qui a été choisi (voir section 2.2). Ces deux capteurs sont solidement fixés l'un à l'autre grâce à une platine métallique spécialement conçue pour nos besoins. On appelle ce montage un *rig* bi-caméra. Il est primordial que les deux capteurs ne bougent pas l'un par rapport à l'autre pendant le tournage d'une séquence. Nous avons principalement utilisé la caméra film *Blackmagic*, cependant le capteur de profondeur peut s'adapter à plusieurs type de caméras. La figure 2.17 donne un aperçu du système mis en place avec deux caméras de tournage différentes. Cette modularité est un élément de différenciation de notre système vis-à-vis du modèle *ALEXA SCENE* vu en section 2.9. Un *rig* similaire est mis en place dans [GMP12], cependant celui-ci n'est pas destiné à de la *previz on-set* temps réel.

La caméra film est considérée comme la caméra maître du système puisqu'elle représente le point de vue choisi par le réalisateur. C'est dans ce référentiel que l'on cherche à exprimer la profondeur. Par la suite, on indiquera avec l'indice $_f$ les éléments qui font référence à ce capteur. De la même manière, les éléments faisant référence au capteur de profondeur seront annotés avec l'indice $_d$ (pour *depth*).

8. `http://www.pmdtec.com/news_media/video/camcube.php`
9. `http://www.odos-imaging.com/product/starform-3d-time-of-flight-camera/`

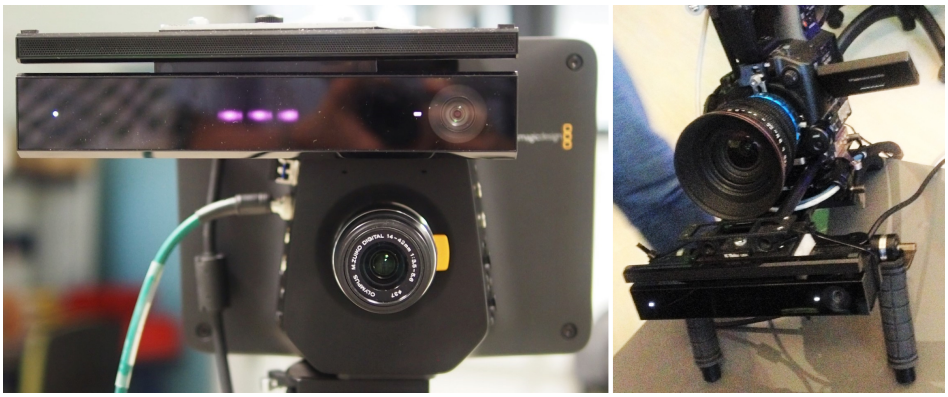


FIGURE 2.17 – Notre système couplant le capteur de profondeur *Kinect 2* avec des caméras différentes. Gauche : *Blackmagic Studio HD* avec une optique 14mm. Droite : *Panasonic AG-AF100A* avec une optique 18mm.

Comme dans le cas d'un capteur stéréo, une *baseline* est fixée entre le capteur couleur, qui correspond à la caméra film, et le capteur profondeur, qui correspond à la caméra infrarouge de la *Kinect 2*. Dans notre cas la *baseline* est principalement verticale et mesure environ 8 cm. Ce décalage est problématique et introduit un effet d'ombre. En effet, le point de vue de la scène est donc différent pour chaque capteur, il peut exister des zones de l'espace qui sont visibles par un capteur et invisibles par l'autre. Ainsi, une fois mise dans le référentiel de la caméra film, la carte de profondeur obtenue par le module *Kinect* comporte des zones d'ombres. Il s'agit des zones où aucune profondeur ne peut être calculée car elles ne sont pas visibles par le capteur de profondeur. Pour réduire ce phénomène, la distance inter-capteurs est minimisée. Cependant, nous restons physiquement limités par la taille des objectifs du matériel.

Par ailleurs, il est possible d'obtenir un écart nul en utilisant des *splitter beamer*, dans le but de parvenir à fusionner les axes optiques des capteurs tout en séparant la couleur de l'infrarouge [ZNI+14]. Il s'agit en fait d'un miroir permettant de diviser en 2 un rayon lumineux. De cette façon les capteurs peuvent être montés perpendiculairement et ainsi s'affranchir des contraintes physiques liées à la taille des objectifs. De tels systèmes sont utilisés au cinéma pour la prise de vue en stéréo. Il est alors possible de choisir une *baseline* en fonction du plan à tourner (voir figure 2.18). Pour notre problématique, il s'agirait de supprimer complètement la *baseline* afin que l'image de profondeur et l'image couleur soient physiquement alignées (modulo un facteur d'échelle suivant la focale utilisée sur chaque système). Un tel dispositif est très coûteux et nécessite une précision mécanique très importante. Ce n'est pas le cas de notre système.

Pour utiliser notre système, une calibration est nécessaire afin d'avoir une connaissance exacte de la transformation spatiale (rotation et translation) entre les deux capteurs. La figure 2.19 donne une vue schématique de la position relative des capteurs. La calibration permet également de déterminer précisément les paramètres optiques de chaque capteur. Cette calibration permet d'obtenir les paramètres nécessaires à la mise en correspondance des informations de profondeur dans le référentiel de la caméra film. Pour le tournage d'un plan, nous considérons que les optiques et la distance inter-capteurs sont figées. Aussi, le processus de calibration

FIGURE 2.18 – Exemple de montage utilisant un *splitter beamer* afin de régler finement l'écart entre deux caméras (source `http://www.3dfilmfactory.com`).

peut-il être réalisé une fois pour toute en amont du tournage.

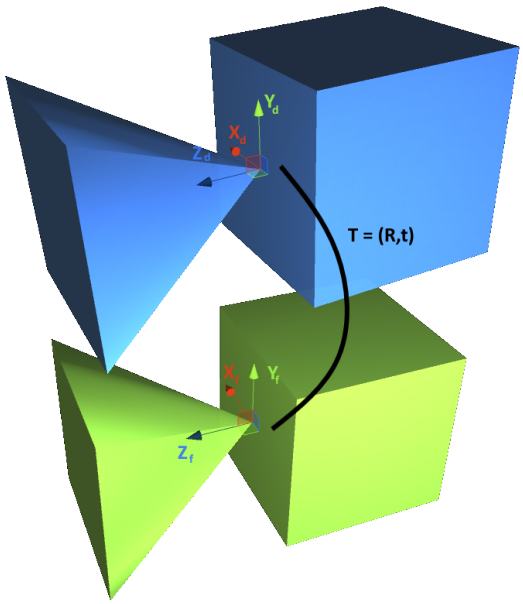


FIGURE 2.19 – Schéma de la position relative des caméras ainsi que les repères associés. Dans notre version du système la plus utilisée, la caméra de profondeur (représentée en bleu) est située au dessus de la caméra film (représentée en vert). La transformation T permet d'exprimer un point 3D du repère de la caméra de profondeur dans le repère de la caméra film.

2.3.3 Modèle du sténopé

Le modèle mathématique que l'on utilise pour modéliser les optiques des capteurs est le modèle du sténopé, ou *pinhole camera* en anglais [BK08]. Il s'agit du modèle le plus couramment utilisé en vision par ordinateur. Ce modèle, qui assimile l'objectif à un trou d'épingle, implique que la profondeur de champ est infinie. Avec ce modèle, l'image sera toujours nette puisque qu'un rayon lumineux ne pourra pas atteindre plus d'un photosite à la fois. La figure 2.20 illustre ce procédé.

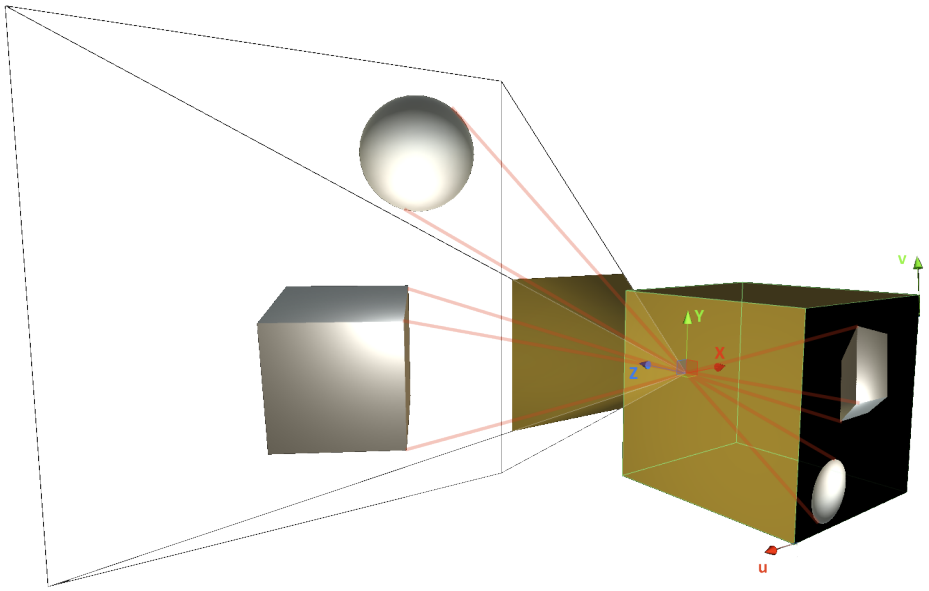


FIGURE 2.20 – Illustration du modèle *pinhole camera*. L'objectif de la caméra est représenté par un point par lequel passent tous les rayons.

Ce modèle traduit la projection perspective d'un point de l'espace $\boldsymbol{p} \in R^3 = (x, y, z)^\mathsf{T}$ en un point dans le plan image $\boldsymbol{u} \in R^2 = (u, v)^\mathsf{T}$. Le plan image est un plan perpendiculaire à l'axe optique (axe Z) de la caméra, il est situé à la distance f du trou d'épingle. La distance f représente la focale de la caméra. Le plan image correspond à la dalle de photosites qui reçoit les rayons lumineux et permet de numériser l'image. Les paramètres physiques de la projection telle que la distance du plan image au trou d'épingle sont stockés dans la matrice de projection $\boldsymbol{K}$. On utilise la notation avec un point au dessus de la lettre du vecteur pour désigner le vecteur homogène tel que $\dot{\boldsymbol{u}} = (\boldsymbol{u}^\mathsf{T}|1)^\mathsf{T}$. On obtient l'équation de projection suivante 2.3 :

$$z\dot{\boldsymbol{u}} = \boldsymbol{K}\boldsymbol{p} \tag{2.3}$$

La matrice de projection $\boldsymbol{K}$ représente les caractéristiques optiques d'une caméra. Cette

matrice est définie comme suit :

$$\boldsymbol{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

avec f_x et f_y les focales horizontales et verticales exprimées en pixel. Pour le modèle *pinhole camera*, la focale correspond à la distance du centre optique (trou d'épingle) au plan image. Pour des pixels carrés, on a $f_x = f_y$. Les grandeurs c_x et c_y représentent les coordonnées en pixel de la projection du centre optique de la caméra sur le plan image. Ce point est aussi appelé point principal. Lorsque l'optique est parfaitement centrée avec le capteur, alors c_x et c_y correspondent à la moitié de la résolution de la dalle de photosites. Notons qu'en pratique ce n'est pas toujours le cas, il y a souvent un léger décalage.

D'autres défauts sont également présents dans les caméras. Ceux-ci sont liés à la précision de l'assemblage et à la qualité des optiques utilisés. Dans le cas où le capteur n'est pas parallèle à la lentille, on observe par exemple de la distorsion tangentielle. En outre, les lentilles utilisées, contrairement au modèle mathématique, sont rarement des trous d'épingles. Une distorsion radiale apparait dans l'image lorsqu'on s'éloigne du centre de l'image. Cela peut, par exemple, donner à l'image un effet de barillet. Ces distorsions sont décrites dans [Bro71, FB86]. On caractérise la distorsion radiale par trois paramètres : p_1, p_2 et p_3 tandis que la distorsion tangentielle est modélisée par deux paramètres k_1 et k_2. Ceux ci sont rangés dans le vecteur de distorsion $\boldsymbol{d} = (p_1, p_2, k_1, k_2, p_3)^\mathsf{T}$.

La matrice de projection $\boldsymbol{K}$ et le vecteur de distorsion $\boldsymbol{d}$ sont les paramètres intrinsèques d'une caméra. Ils sont estimés lors de l'étape de calibration de la caméra.

2.3.4 Calibration des paramètres intrinsèques

Afin de calibrer les paramètres intrinsèques des deux capteurs du systèmes, nous utilisons une méthode basée sur la reconnaissance de damier [Zha99, Zha00, Bou14]. Cette étape est indépendante pour chacun des capteurs. Un damier plan, composé de carrés noirs et blancs, est placé devant le capteur à calibrer. Une acquisition de plusieurs images du damier avec diverses orientations et positions est réalisée. Le traitement de cette séquence d'images par un algorithme de calibration permet de déterminer la matrice de projection $\boldsymbol{K_f}$ et les paramètres de distorsion $\boldsymbol{d_f}$ pour la caméra film (respectivement $\boldsymbol{K_d}$ et $\boldsymbol{d_d}$ pour la caméra infrarouge du capteur de profondeur). En effet il est important de noter que le référentiel de la carte de profondeur est celui de la caméra infrarouge du capteur ToF, c'est pour cette raison que nous procédons à la calibration de la caméra infrarouge. La figure 2.21 montre la détection du motif de calibration dans la vue du capteur de profondeur.

La calibration des paramètres intrinsèques est implémentée avec des routines de la librairie *OpenCV*. La première étape de l'algorithme de calibration consiste à déterminer la position de la caméra vis-à-vis du damier. On parle de méthodes d'estimation de la pose à partir d'une

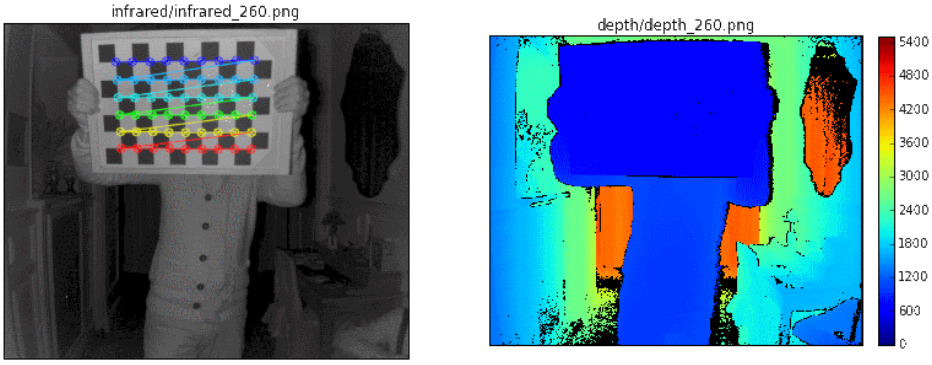


FIGURE 2.21 – Exemple d'images issues de la caméra de profondeur. Gauche : détection du damier dans l'image infrarouge du capteur de profondeur (l'histogramme de l'image infrarouge a été étalé afin de mieux voir les détails de l'image). Droite : image de profondeur associée.

information 2D [GHTC03, LMNF09] . Il n'est pas nécessaire d'utiliser ces méthodes avec le capteur de profondeur. En effet, on peut déterminer la position du capteur de profondeur vis-à-vis du damier directement avec les mesures de distances obtenues. A partir de l'observation des coins du damier dans l'image infrarouge et d'une estimation de la matrice de projection il est possible de construire le nuage de points 3D correspondant aux coins du damier. Le barycentre de ce nuage de points nous donne la translation entre le capteur et le centre du damier. Si on applique ensuite une décomposition en valeur singulière (SVD) à ce nuage de points centrés (on recentre chaque point en soustrayant le barycentre), on peut déduire la matrice de rotation qui est appliquée au damier. Chaque colonne de la matrice de rotation est en fait un vecteur propre de la SVD réalisée précédemment. On utilise la transformation calculée pour projeter dans l'image les coins d'un damier parfait. Il est alors possible de calculer l'erreur entre les points observés et les points projetés. Une méthode d'optimisation itérative permet par la suite de déterminer la matrice de projection $\boldsymbol{K_d}$ du capteur de profondeur qui minimise cette erreur. Cette méthode que nous avons développée est détaillée dans l'annexe B.

2.3.5 Calibration des paramètres extrinsèques

La calibration du système est réalisée afin de pouvoir exprimer la profondeur de la scène dans le référentiel de la caméra film. Il est nécessaire de connaître précisément la transformation géométrique afin de passer du repère de la caméra de profondeur au repère de la caméra film. On note $T_{d\to f} \in SE(3)$ cette transformation. Il s'agit d'une transformation euclidienne composée d'une rotation $R_{d\to f} \in SO(3)$ et d'une translation $\boldsymbol{t_{d\to f}} \in R^3$. La notation $SO(3)$ désigne le groupe spécial orthogonal qui contient les rotations dans un espace en trois dimensions. La notation $SE(3)$ désigne le groupe spécial euclidien qui contient les transformations euclidiennes (rotation et translation) dans l'espace 3D.

Cette transformation est représentée sous la forme d'une matrice carrée de dimension 4

(équation 2.5) :

$$T_{d\to f} = \begin{pmatrix} R_{d\to f} & \boldsymbol{t_{d\to f}} \\ 0 & 1 \end{pmatrix} \tag{2.5}$$

Avec la transformation $T_{d\to f} \in SE_3$, un point $\boldsymbol{p_d} \in R^3$ dans le référentiel de la caméra de profondeur s'exprime alors comme le point $\boldsymbol{p_f} \in R^3$ dans le référentiel de la caméra film (équation 2.6) :

$$\dot{\boldsymbol{p}}_{\boldsymbol{f}} = T_{d\to f}\dot{\boldsymbol{p}}_{\boldsymbol{d}} \tag{2.6}$$

Afin de calculer cette transformation, nous utilisons également une méthode mobilisant les damiers. Dans cette seconde configuration, le damier doit être entièrement visible simultanément par chacun des capteurs. Cela peut être problématique si les focales des deux caméras sont très différentes.

L'algorithme de calibration permet alors de calculer pour chaque vue du damier la position relative de celui-ci par rapport au système de coordonnées de la caméra. En ce sens, en combinant la position relative de chaque caméra par rapport au damier, on calcule pour chaque vue une estimation de la transformation euclidienne $T_{d\to f}$ entre le système de coordonnées de la caméra film et celui de la caméra de profondeur. Ensuite, on enregistre une séquence d'images avec le damier à divers distances du système simultanément avec la caméra film et la caméra de profondeur. Chaque couple d'images est traité, et la mesure de la transformation euclidienne obtenue est ajoutée à une pile. Enfin, dans le but d'obtenir une calibration extrinsèque optimale, un ajustement global est accompli à l'aide d'un algorithme itératif de Levenberg-Marquardt [BK08]. Il est alors possible de modéliser la position relative des capteurs entre eux (voir figure 2.22).

2.3.6 En pratique

Il est possible d'obtenir les paramètres intrinsèques et extrinsèques en une seule manipulation mais il est souvent plus approprié de le faire en deux temps. En effet, le champ de vision n'est pas toujours le même entre les deux capteurs. Il est nécessaire de maximiser la taille du damier dans la vue de la caméra lors de la phase de calibration des paramètres intrinsèques, tandis qu'il faut aussi que chaque capteur voit le damier en entier pendant la phase de calibration des paramètres extrinsèques.

La phase de calibration est effectuée une seule fois avant l'utilisation du système. Tous les paramètres sont ensuite sauvegardés dans un fichier qui sera chargé à chaque utilisation du système.

Nous aborderons la description d'une base de données constituée de séquences vidéo avec information de profondeur dans le chapitre 3. Nous avons enregistré des séquences d'images

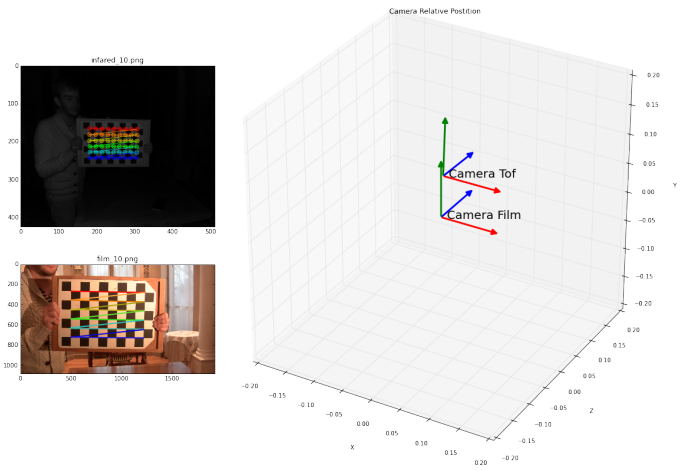


FIGURE 2.22 – La calibration extrinsèque des capteurs permet de positionner la caméra ToF relativement à la caméra film. Dans notre cas le capteur de profondeur est situé verticalement au dessus de la caméra film.

contenant un damier afin de calibrer la caméra film et la caméra de profondeur indépendamment (paramètres intrinsèques) ainsi que l'une par rapport à l'autre (paramètres extrinsèques). Nous avons calculé pour chaque étape l'erreur de reprojection afin de quantifier la qualité de la calibration. La formule utilisée pour le calcul de l'erreur est la suivante (2.7) :

$$e = \sqrt{\frac{\sum_{i=0}^{N-1} \|\boldsymbol{u_i} - \tilde{\boldsymbol{u}}_{\boldsymbol{i}}\|_2^2}{N}} \tag{2.7}$$

Avec $\boldsymbol{u_i}$ les coordonnées en pixel du coin i du damier observé dans l'image, et $\tilde{\boldsymbol{u}}_{\boldsymbol{i}}$ les coordonnées en pixel de la projection du coin i du damier dans l'image (projeté avec les paramètres intrinsèques et extrinsèques estimés). Un coin est un sommet d'un carré du damier. Le nombre N représente le nombre de l'ensemble des coins détectés dans les images servant pour la calibration.

Le tableau suivant (2.2) donne un aperçu du nombre d'images utilisées ainsi que l'erreur de reprojection.

Calibration	Nombre de damiers	erreur de reprojection (pixels)
Intrinsèques caméra film	403	0.31
Intrinsèques caméra de profondeur	630	0.14
Extrinsèques	267	0.25

Tableau 2.2 – Nombre de damiers utilisés et erreur de reprojection pour la calibration du système avec les séquences de tests.

2.4 Moteur de jeux vidéos

Les besoins en développement logiciel sont importants. Plusieurs blocs tels que l'acquisition des images des différents capteurs, la mise en relation des données, la visualisation 3D des données, l'importation de modèles 3D, le rendu 3D, l'incrustation, la gestion de l'animation des éléments virtuels, la gestion des interactions entre éléments virtuels et la gestion de la physique sont nécessaires. Le développement de chacun des blocs nécessite un investissement en temps considérable. Aux prémices de ce travail de recherche, nous avons utilisé le *framework OpenFramework*[10]. Son utilisation est très adéquate pour développer une application sur ordinateur qui nécessite de manipuler la 3D. Il s'agit d'un cadre de travail *open source* développé en C++. Une première version du logiciel est développée avec ce *framework*, comme l'illustre la figure 2.23.

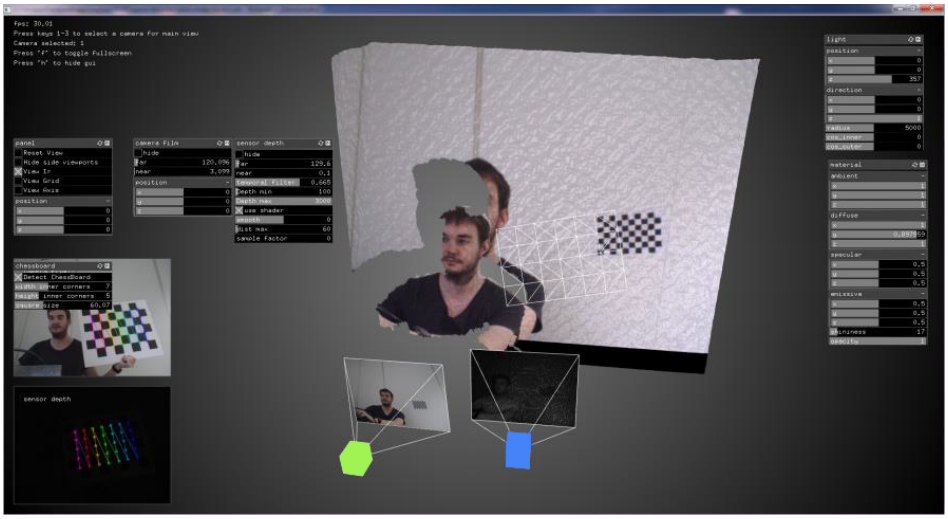


FIGURE 2.23 – Capture d'écran de la première version du logiciel développé avec le *framework OpenFrameworks*. On aperçoit en bas à gauche des vignettes faisant apparaître le flux de la caméra film et de la caméra *Kinect*. La vue principale correspond à une visualisation 3D de la géométrie de la scène sur laquelle est plaquée la texture issue de la caméra film.

Il s'agit d'une interface de haut niveau aux API graphiques (OpenGL, DirectX ...) qui permettent de tirer parti des performances du matériel d'accélération graphique tel que le GPU[11]. Les API graphiques offrent un cadre de travail simplifié pour exploiter le pipeline graphique. Ce pipeline est paramétrable avec différents niveaux de contrôle en fonction du matériel utilisé. Nous utilisons les *shaders* afin de paramétrer le pipeline (voir figure 2.24). Un *shader* correspond à un bloc de code à destination de la carte graphique. Celle-ci le compile puis utilise le programme résultant pour contrôler une étape du rendu. Nous utilisons le langage GLSL[12] pour écrire nos *shaders*. Dans ce travail de recherche nous développons des *vertex shaders*, des *geometry shaders* et des *fragment shaders*.

10. `http://openframeworks.cc/`
11. *Graphic Processing Unit*
12. *Open**GL** **S**hading **L**anguage*

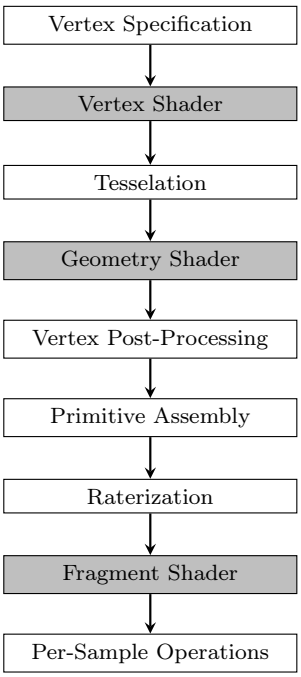


FIGURE 2.24 – Pipeline de rendu utilisé par OpenGL. Les blocs en gris correspondent aux blocs programmables par l'intermédiaire d'un *shader*. (schéma inspiré de `https://www.opengl.org/`)

Comme nous le verrons dans le chapitre suivant, nous utilisons des filtres pour améliorer la segmentation de la silhouette d'un acteur. Les *fragment shaders* sont particulièrement bien adaptés au filtrage des images. En effet, lorsque le filtrage est indépendant pour chaque pixel, il est possible de décrire le traitement à appliquer à l'intérieur de ce *shader*. Le code présent dans le *fragment shader* est exécuté pour chaque pixel qui doit être rendu. Grâce aux nombreux processeurs présents dans la carte graphique l'exécution est réalisée en parallèle. Le pseudo-code d'un *fragment shader* de filtrage moyen horizontal est donné dans la suite.

```
// Fragment shader: filtrage moyen horizontal
void main( out vec4 fragColor, in vec2 fragCoord )
{
    vec2 xy = fragCoord.xy; //coordonnees du pixel courant
    vec4 texColorLeft = texture2D(iTexture0,xy - vec2(1,0)); //valeur du pixel a ↩
        gauche du pixel courant dans la texture iTexture0
    vec4 texColorCenter = texture2D(iTexture0,xy); //valeur du pixel courant dans la↩
         texture iTexture0
    vec4 texColorRight = texture2D(iTexture0,xy + vec2(1,0)); //valeur du pixel a ↩
        droite du pixel courant  dans la texture iTexture0
    fragColor = 1.0 / 3.0 * (texColorLeft + texColorCenter + texColorRight); //on ↩
        affecte la moyenne  au  pixel de sortie
}
```

Nous nous sommes aperçus que les fonctionnalités développées au début de la thèse au travers d'*OpenFrameworks*, se révèlent très similaires à celles proposées par un moteur de jeux vidéo. Nous avons alors revu notre approche pour utiliser un moteur de jeux. Le moteur de

jeux a l'avantage de proposer un socle fiable et éprouvé. Nous exploitons le moteur de jeux *Unity 3D*. Il s'agit de la plateforme de développement logiciel de notre système. *Unity 3D* prend également en charge les *shaderes* GLSL afin de programmer le pipeline graphique. Afin que l'utilisateur soit en mesure de pratiquer rapidement différents types de rendus, des *shaders* sont pré-installés dans *Unity*.

Traditionnellement un moteur de jeux vidéo tel que *Unity* est utilisé pour la création de jeux 2D ou 3D pour diverses plateformes telles que *Android*, *iOS*, PC (*Windows*, *Mac*) ou même consoles (*Playstation*, *Xbox*). Il s'agit d'un logiciel où l'on peut importer des assets 3D, mettre en place un éclairage, et scripter le comportement des éléments graphiques en fonction d'évènements extérieurs tels qu'une entrée au clavier du joueur par exemple.

Cette section liste les principales contributions du moteur de jeux vidéo qui démontre une grande pertinence pour la prévisualisation temps réel sur les plateaux de tournages. La figure 2.25 illustre l'architecture mise en place. Les étapes traitant du recalage spatial de la profondeur et du *compositing* sont détaillées dans les sections 2.5 et 2.6.

2.4.1 Acquisition des différents flux

Notre système comporte un capteur vidéo couleur HD, appelé aussi caméra film, associé à un capteur de profondeur qui correspond à la *Kinect 2*.

Le flux vidéo couleur haute définition est transféré de la caméra film à l'ordinateur via une carte d'acquisition (SDI à USB3). L'image film HD (1920×1080 pixels) est stockée dans une texture 2D, une structure spécifique gérée par la carte graphique de l'ordinateur. Le capteur *Kinect 2* délivre une carte de profondeur avec une définition de 512×424 pixels, 16 bits par pixel. L'image de profondeur est également stockée dans une texture 2D sur la carte graphique.

Comme nous l'avons explicité dans la section 2.3 le module *Kinect* n'est pas un module de captation professionnel. Le flux d'images de profondeur produit n'est pas toujours constant et il n'est pas possible de synchroniser temporellement le débit de ce flux par une horloge externe. Cependant, la *Kinect* accole un *timestamp* à chaque image qu'elle produit. Le SDK de la *Kinect* permet d'accéder à la valeur de ce *timestamp*. Nous avons mis en place un *buffer* circulaire où chaque image de profondeur est stockée avec son *timestamp* associé. Lorsqu'une image film est interceptée par le moteur de jeu, la carte de profondeur avec le *timestamp* le plus proche est sélectionnée et associée à cette image (voir figure 2.26). Cette méthode permet de compenser le temps de latence qui peut apparaître entre le flux couleur et le flux de profondeur pour maintenir une synchronisation temporelle.

2.4.2 Édition et contrôle de la scène

L'édition et le contrôle de la scène représente deux manipulations essentielles pour permettre à l'opérateur d'être en mesure d'interférer en temps réel avec la scène virtuelle afin de répondre aux exigences du réalisateur.

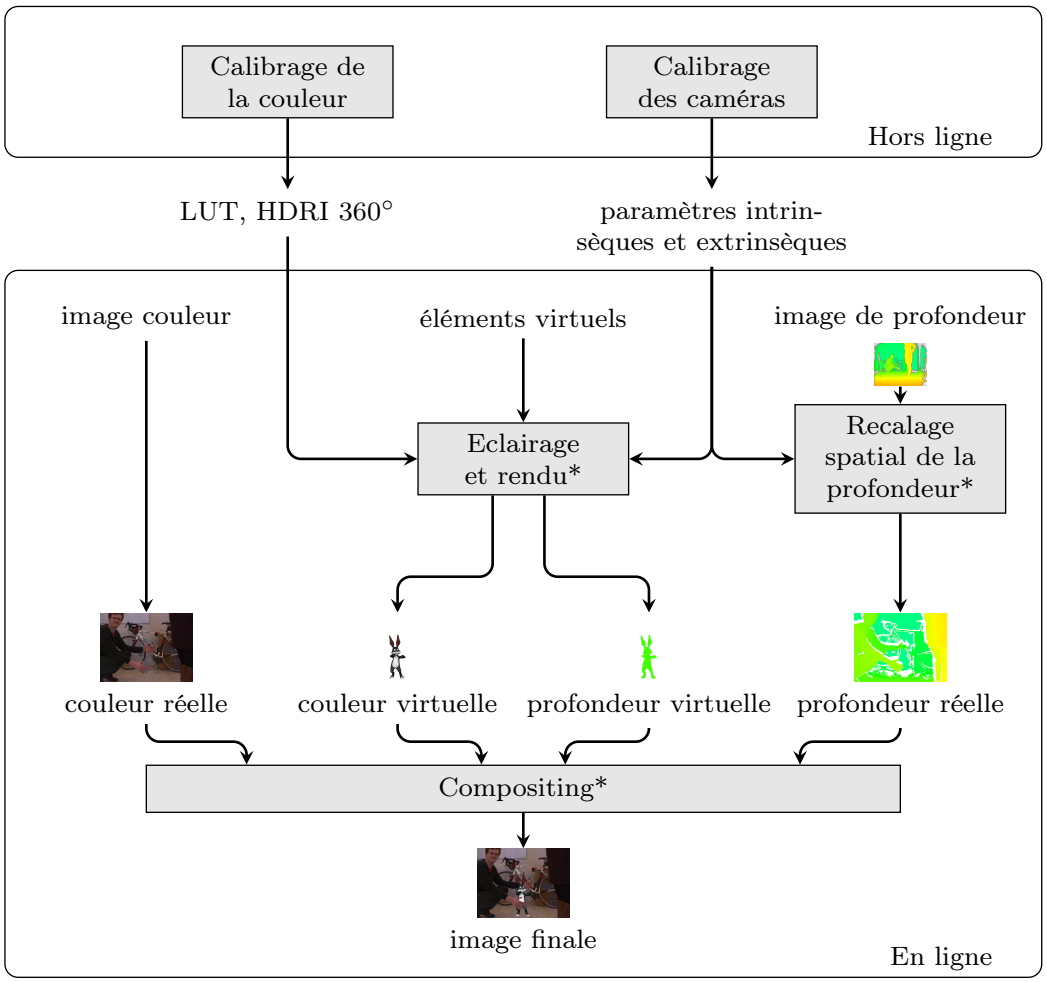


FIGURE 2.25 – Architecture globale du système au sein du moteur de jeux vidéo. Le processus hors ligne est effectué une seule fois au démarrage. Le processus en ligne est exécuté en temps réel pour chaque image de la caméra film à l'intérieur du moteur de jeux vidéo. Les blocs marqués d'une * utilisent des *shaders* dans leur fonctionnement.

Un moteur de jeux vidéo est utilisé pour le rendu en temps réel d'une scène virtuelle. Il s'agit d'un outil optimisé pour traiter un nombre important d'éléments virtuels, d'éclairages, et d'interactions physiques. Dans notre contexte, nous considérons le moteur de jeux comme un puissant mélangeur vidéo. Plusieurs entrées sont mélangées pour produire la prévisualisation tout en utilisant l'ensemble des fonctionnalités exploitables à partir d'un moteur de jeux. Il est possible de programmer le comportement du moteur de jeu en codant des fonctions qui seront appelées dans la boucle principale. Cette boucle principale est le cœur du fonctionnement du moteur de jeux vidéo. Celle-ci tourne dès le commencement du jeu et c'est elle qui va permettre la mise à jour de l'affichage des éléments à l'écran. Dans notre cas nous synchronisons cette boucle avec la fréquence d'acquisition de la caméra film, de sorte que chaque fois qu'une image de la caméra film est interceptée, une boucle de code est exécutée.

Unity met également à disposition un mode d'édition temps réel afin de contrôler chaque

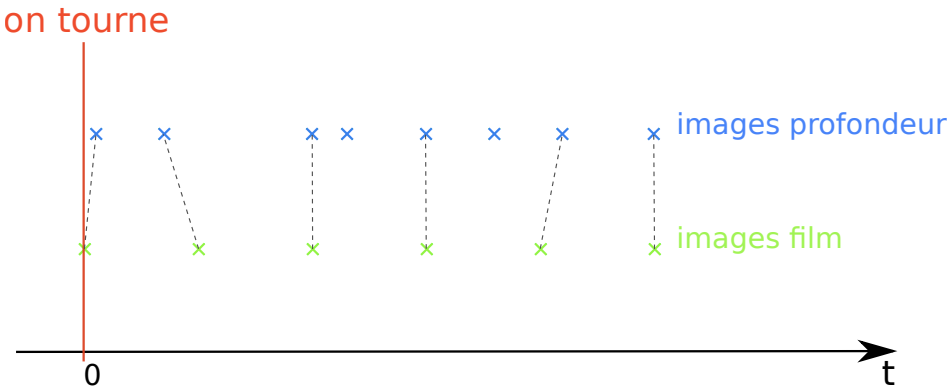


FIGURE 2.26 – Synchronisation temporelle. Chaque image film s'associe avec l'image de profondeur la plus proche temporellement.

paramètre de chaque élément de la scène virtuelle au moyen d'une interface graphique. Cela signifie qu'il est possible d'ajuster l'éclairage, d'ajouter des éléments 3D, de déplacer les objets virtuels ou encore de déclencher des animations. Les contrôles principaux sont déjà inclus dans le *framework* du moteur de jeux, et il est également simple d'ajouter des contrôles personnalisés dans l'interface d'édition.

2.4.3 Illumination et étalonnage automatiques

Afin de mélanger de façon homogène les éléments virtuels avec ceux du monde réel il est important de reproduire l'éclairage et la réponse colorimétrique [DM08] de la scène réelle sur les éléments virtuels[KTPW11]. Pour accomplir cela, une image HDR [13] de la scène est capturée en utilisant un appareil photo dédié avec une lentille à 360° (voir la figure 2.27, haut). On utilise ensuite une mire de couleur, *Greta MacBeth ColorChecker* pour calculer deux fonctions de transfert A et B qui sont stockées dans des LUT [14](voir la figure 2.27, bas). La fonction A transfère un espace couleur neutre (sRGB) dans l'espace couleur de la caméra de tournage. La fonction B transfère l'espace couleur de l'appareil photo 360° vers un espace de couleur neutre.

L'image HDR est transférée dans l'espace de couleur neutre en utilisant B. Le résultat est ensuite utilisé pour éclairer les éléments virtuels dans l'espace neutre en exploitant la technique IBL [15] . Enfin, la fonction de transfert A est appliquée à l'image rendue de la scène virtuelle afin qu'elle soit cohérente avec la réponse colorimétrique de la caméra de tournage. Le moteur de jeux *Unity* supporte de façon inhérente un système temps réel et optimisé IBL via l'utilisation de *shaders*.

13. *Hight Dynamic Range*
14. *Look Up Table*
15. *Image Based Lighting*

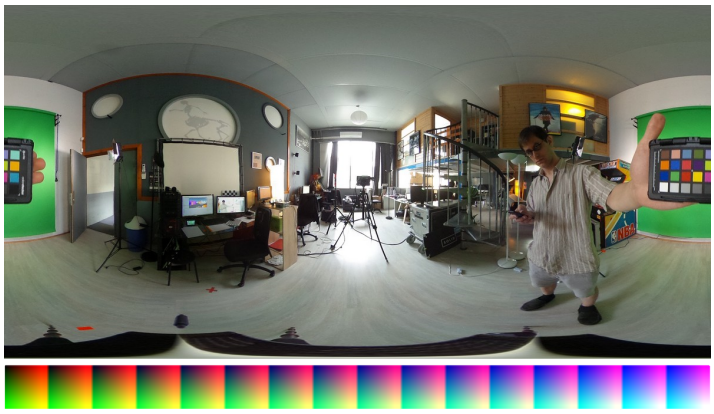


FIGURE 2.27 – Illumination et étalonnage automatique. Haut : échantillon d'une image à 360° HDR utilisée pour éclairer les éléments virtuels. Le damier utilisé pour la calibration de la couleur est visible de chaque côté de la photo. Bas : LUT stockée dans une texture 2D qui transfère l'espace couleur de l'image HDR vers un espace couleur neutre.

2.4.4 Interaction entre les éléments réels et virtuels

Les moteurs de jeux sont le plus souvent accompagnés d'un moteur physique comme c'est le cas avec *Unity*. Cela signifie que chaque élément virtuel est physiquement capable de se comporter comme un objet réel. Il est ainsi possible d'ajouter de la gravité, d'associer une masse pour chaque objet et d'incorporer un détecteur de collision. C'est grâce au moteur physique que les objets peuvent se repousser et s'attirer ou bien qu'un personnage virtuel peut interagir avec un objet virtuel par exemple. Dans notre cas, on s'intéresse à l'interaction entre les acteurs réels et les objets virtuels. Même s'il est techniquement possible de calculer la collision entre un objet virtuel et le maillage détaillé issu de la carte de profondeur, cela affecte les performances temps réel. Nous proposons de résoudre ce problème en utilisant une représentation 3D simplifiée du squelette de l'acteur. La détection de squelette humain est calculée avec une méthode décrite dans [SSK+13] détaillée dans la section 2.2. Lorsqu'un squelette 3D entre en collision avec un objet virtuel, le moteur de jeu déclenche un événement. Cet événement peut être relié à un script où chaque interaction physique peut être planifiée. Un exemple d'interaction simulant la destruction d'un mur de briques est illustré par la figure 2.28.

2.4.5 Sauvegarde des données et post-processing

La *previz on-set* est un outil permettant de démultiplier les potentiels de création durant le tournage. Ses potentiels s'étendent aux domaines de la post-produtiocn. Pour ce faire, la profondeur et le *camera tracking* sont enregistrés indépendamment pendant le tournage. Ces enregistrements différenciés permettent de rejouer les séquences de la scène en se dégageant de la contrainte du temps réel a posteriori.

Il est possible de ralentir un moteur de jeux afin d'offrir un temps de calcul plus important

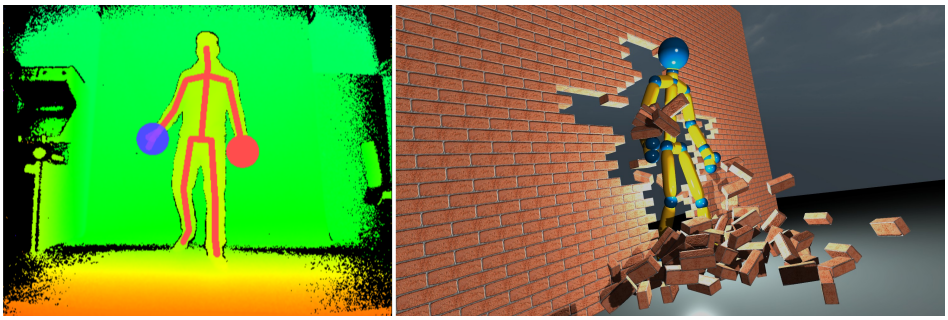


FIGURE 2.28 – Exemple d'interaction. Gauche : détection du squelette dans la carte de profondeur. Droite : représentation 3D du squelette dans l'éditeur du moteur de jeux vidéo avec détection de la collision entre le squelette et un mur de briques virtuel.

pour chaque image. Cela permet d'obtenir de meilleures performances de rendus. Pour prendre en compte les impératifs de notre problématique nous avons décidé de convertir nos données dans une séquence d'images *openEXR*[16]. Ce format est initialement développé par *Illuminated Light and Magic* et représente à l'heure actuelle un standard dans l'industrie du film. Ce format supporte le multi-canaux, et jusqu'à 32 bits de précision par canal. Il est également possible d'ajouter des informations personnalisées dans les en-têtes de fichier, telle que la position 3D de la caméra. De cette façon, les données de la *previz* sont stockées dans un format standardisé, manipulable par les équipes de post-production. Enfin des logiciels dédiés, tels que *Nuke*[17] ou *Natron*[18], qui sont très utilisées dans les studios de post-production, permettent de réaliser le *compositing* des différentes images générées par le moteur de jeux vidéo.

Une équipe d'ingénieurs à Solidanim travaille sur ce sujet précis qui est la chaîne de post-traitement de la *previz*.

16. `http://www.openexr.com/`
17. `https://www.thefoundry.co.uk/products/nuke/`
18. `https://natron.inria.fr/`

2.5 Alignement spatial de l'image de la profondeur avec l'image couleur

Plusieurs traitements sont nécessaires afin de mettre en relation les données de l'image film avec celles de l'image de profondeur.

Le premier traitement des données consiste à effectuer un recalage temporel. Il s'effectue grâce au *timestamp* contenu dans les images issues du capteur de profondeur. En effet, contrairement à une caméra film, le flux n'est pas toujours constant. L'action de recaler les flux dans le temps permet d'associer à chaque image film une image de la profondeur. Cette étape est expliquée dans la section précédente.

Le second traitement consiste à recaler l'image film dans l'espace avec l'image de profondeur. Ces deux images ne se superposent pas pour plusieurs raisons : elles correspondent à un point de vue, à une densité de pixels, à un champ de vision différents. Les paramètres calculés lors de l'étape de calibration sont alors exploités (voir section 2.3). Cette section développe notre méthode pour obtenir une image de profondeur qui se superpose à l'image couleur de la scène réelle obtenue avec la caméra film.

2.5.1 Comparaison des focales et de la densité de pixels du capteur film et profondeur

Dans un premier temps on superpose les centres optiques du capteur de profondeur et de la caméra film. Il s'agit d'obtenir une première estimation de la surface utile du capteur de la caméra de profondeur (figue 2.29).

Pour obtenir cette estimation, on commence par calculer le rapport des focales, exprimées en pixels, entre la caméra film et le capteur de profondeur. Exprimer la focale d'un capteur en pixel permet d'incorporer à la fois la résolution du capteur et son champ de vision. On appelle ce rapport α :

$$\alpha = \frac{Focale_d}{Focale_f} \tag{2.8}$$

On obtient les dimensions $w_{d,utile} \times h_{d,utile}$ de la surface utile du capteur de profondeur en multipliant les dimensions du capteur de la caméra film avec le rapport α. Cela nous donne en pratique :

$$\begin{cases} w_{d,utile} = \alpha \times w_f \\ h_{d,utile} = \alpha \times h_f \end{cases} \tag{2.9}$$

Appliquons ces formules à notre prototype. Avec la caméra film nous utilisons un objectif qui présente une focale de $14mm$. En prenant en compte les dimensions physiques du capteur

FIGURE 2.29 – Surface utile du capteur de profondeur. Le rectangle rouge dans l'image de profondeur brute correspond à la surface utile puisqu'il s'agit de la surface qui correspond au champ de vision de la caméra film. Il s'agit ici d'une estimation puisque les centres optiques des capteurs de profondeur et caméra film ne sont, en réalité pas confondus.

de la caméra, cela correspond à une focale de 2150.47 pixels. Le module *Kinect* a une focale de 364.81 pixels. On obtient un rapport $\alpha = 0.17$, soit une surface utile de 326.4×183.6 pixels pour le capteur de profondeur. Il s'agit d'une surface environ 36 fois plus petite que l'image film. Il est donc nécessaire de faire un sur-échantilonnage de la carte de profondeur pour l'adapter à l'image couleur. La figure 2.30 illustre le rapport entre ces trois échelles.

Pour l'estimation de cette surface utile, le décalage entre les capteurs n'est pas pris en compte. Elle permet néanmoins de donner un ordre de grandeur dans le rapport d'échelle. La surface utile peut varier en fonction de la profondeur des éléments observés.

2.5.2 Méthode de recalage spatial

L'objectif est de répercuter l'information de profondeur dans le référentiel de l'image film. Comme nous l'avons développé dans la section 2.4, notre méthode se structure en deux étapes principales. La calibration permet de récupérer les paramètres intrinsèques de chaque caméra et de récupérer les paramètres extrinsèques des deux caméras. Grâce à ces paramètres la carte de profondeur est re-projeté en 3D afin de former une modélisation 3D de la scène réelle. Cette modélisation est observée avec le point de vue de la caméra film dans le but de recaler la carte de profondeur. La carte de profondeur obtenue est alignée avec l'image film de la scène. La figure 2.31 caractérise ces manipulations en détail.

Les coordonnées d'un pixel d'indice i dans l'image de profondeur D sont définies par le

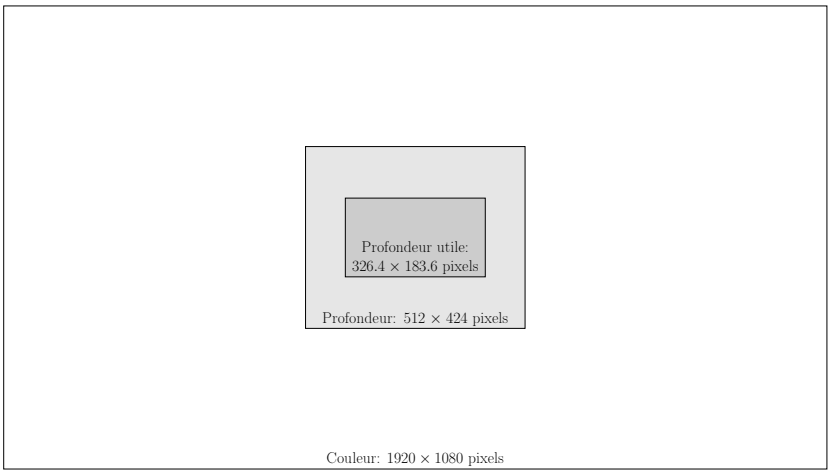


FIGURE 2.30 – Schéma illustrant le rapport d'échelle conservé entre la densité de pixels du capteur couleur, le capteur de profondeur et la surface utile du capteur de profondeur.

vecteur $\boldsymbol{u_{i,d}} \in R^2$. La profondeur $z_{i,d}$ du pixel i dans l'image de profondeur est récupérée en lisant la valeur de l'image de profondeur aux coordonnées $\boldsymbol{u_{i,d}}$. On a $z_{i,d} = D(\boldsymbol{u_{i,d}})$. En utilisant la matrice de projection inverse de la caméra de profondeur K_d^{-1} il est possible de calculer les coordonnées du point $\boldsymbol{p_{i,d}} \in R^3$ dans le référentiel de la caméra de profondeur correspondant au pixel i, tel que détaillé dans l'équation 2.10. Il est important de noter que $z_{i,d}$ correspond exactement à la troisième coordonnée du vecteur $\boldsymbol{p_{i,d}}$. Dans la suite, afin d'alléger la notation on ne notera plus l'indice i du pixel. On gardera cependant l'indice d ou f qui permet de bien visualiser le référentiel dans lequel on se se situe (référentiel de la caméra de profondeur ou référentiel de la caméra film).

$$\boldsymbol{p_d} = z_d K_d^{-1} \dot{\boldsymbol{u}}_{\boldsymbol{d}} \tag{2.10}$$

De cette façon, on obtient une représentation 3D de la scène sous forme d'un nuage de points. Désormais, ce nuage peut être observé avec le point de vue de la caméra film en utilisant les paramètres extrinsèques $T_{d\to f}$. On obtient les coordonnées 3D de chaque point $\boldsymbol{p_f} \in R^3$ dans le référentiel de la caméra film (équation 2.11).

$$\dot{\boldsymbol{p}}_{\boldsymbol{f}} = T_{d\to f} \dot{\boldsymbol{p}}_{\boldsymbol{d}} \tag{2.11}$$

La matrice de projection de la caméra film K_f permet de projeter le nuage de points afin d'obtenir l'image de profondeur alignée (équation 2.12).

$$z_f \dot{\boldsymbol{u}}_{\boldsymbol{f}} = \boldsymbol{K_f p_f} \tag{2.12}$$

Celle-ci correspond à la profondeur parcimonieuse de la scène observée par la caméra film. On écrit la valeur de la profondeur z_f au pixel de coordonnées $\boldsymbol{u_f}$ dans l'image de profondeur

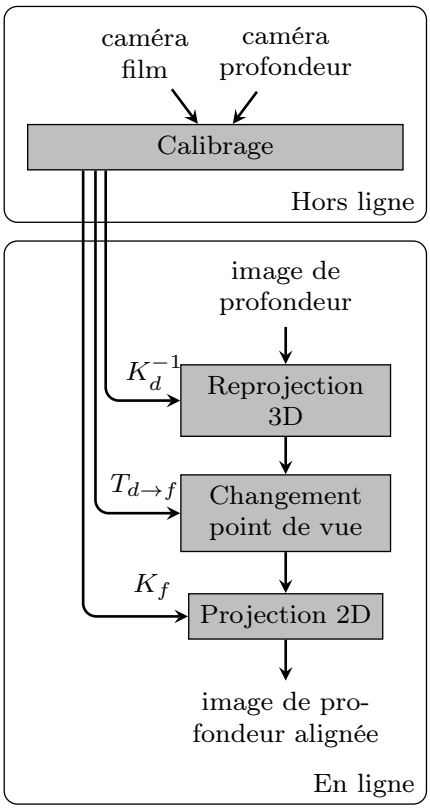


FIGURE 2.31 – Détail de la méthode d'alignement spatial de la carte de profondeur.

rectifiée. L'image de profondeur alignée a la même dimension $w_f \times h_f$ que l'image couleur cependant il ne s'agit pas d'une image dense. En effet, comme on l'a vu dans la sous-section précédente, la densité de pixels de la surface utile dans l'image de profondeur est très inférieure à la densité de pixels de l'image film. C'est pour cette raison que tous les pixels de l'image film ne possèdent pas une information de profondeur. La figure 2.32 permet de visualiser les différentes étapes de la méthode.

2.5.3 Implémentation dans le pipeline GLSL

Au moyen du moteur de jeux vidéo, l'alignement de l'image de profondeur et de l'image couleur est réalisé par l'intervention d'un *shader*. Il s'agit de l'implémentation de la méthode décrite dans la sous-section précédente. Dans notre cas particulier, le *shader* est utilisé pour reprojeter l'information 3D de l'image de profondeur dans le référentiel de la caméra film, et ceci est répété pour chaque image. L'image de profondeur est chargée dans une texture 2D dans le GPU. On envoie également à la carte graphique un maillage possédant autant de nœuds qu'il y a de pixels dans l'image de profondeur. Il s'agit d'une grille plane triangulée de dimensions $w_d \times h_d$. Chaque nœud v_i de la grille est associé à un pixel de l'image de profondeur, les coordonnées de ce pixel dans la texture 2D sont stockées dans un vecteur $\boldsymbol{u_{i,d}}$, communément appelé coordonnées uv. Pour chaque nœud envoyé au *vertex shader*, on identifie

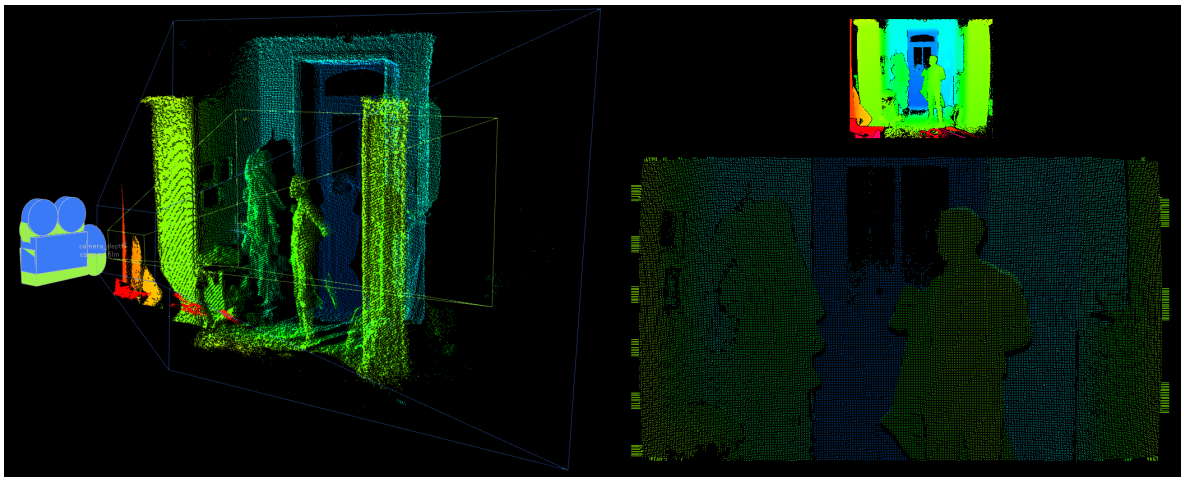


FIGURE 2.32 – L'image de profondeur initiale est reprojetée en 3D pour former un nuage de points. Ce nuage de points est alors observé par la caméra film. On projette les points à travers ce point de vue pour obtenir la carte de profondeur alignée à l'image film. L'image de profondeur obtenue est éparse.

la valeur de la profondeur $z_{i,d}$ en se référant aux coordonnées $\boldsymbol{u_{i,d}}$ de la texture de profondeur (équation 2.13).

$$z_{i,d} = tex(\boldsymbol{u_{i,d}}) \tag{2.13}$$

La prise en compte d'un maillage plutôt que celle d'un nuage de points 3D permet d'observer une surface dense. En effet, la carte graphique interpole une position 3D pour chaque pixel à l'intérieur d'un triangle du maillage. La reprojection de la carte de profondeur basse résolution devient alors une interpolation dense qui s'adapte à la résolution de la caméra film [dGB15]. Cette méthode permet donc de faire un sur-échantillonnage afin de créer une image de profondeur à la dimension de l'image couleur. La figure 2.33 illustre cette méthode. Il est à noter qu'il s'agit ici d'une interpolation linéaire de la profondeur au sein d'un triangle du maillage. D'autres méthodes d'interpolation, telle que bi-cubique, pourraient être envisagées.

Une illustration de la transformation d'une carte de éparse à une carte de profondeur dense dense est présentée dans la figure 2.34.

Il est à noter que le *shader* est codé de telle sorte que les nœuds associés à des pixels ne contenant pas d'information de profondeur, comme une zone trop lointaine ou trop proche, ou encore un matériau absorbant l'infrarouge, sont rejetés par la carte graphique. Ceci élimine certains triangles. Des trous apparaissent alors dans la carte de profondeur recalée spatialement. Les méthodes disponibles pour obstruer ces trous sont présentées dans l'annexe C.

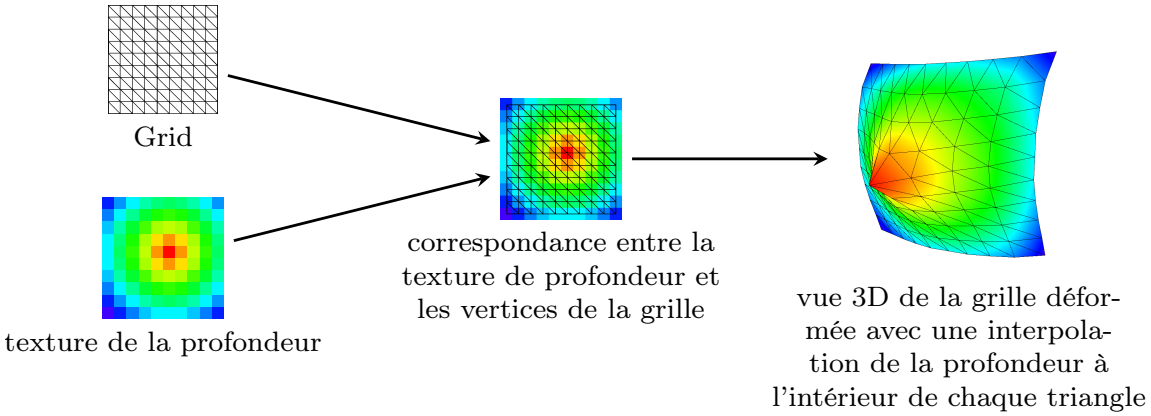


FIGURE 2.33 – Méthode de reprojection 3D de l'image de profondeur avec interpolation.

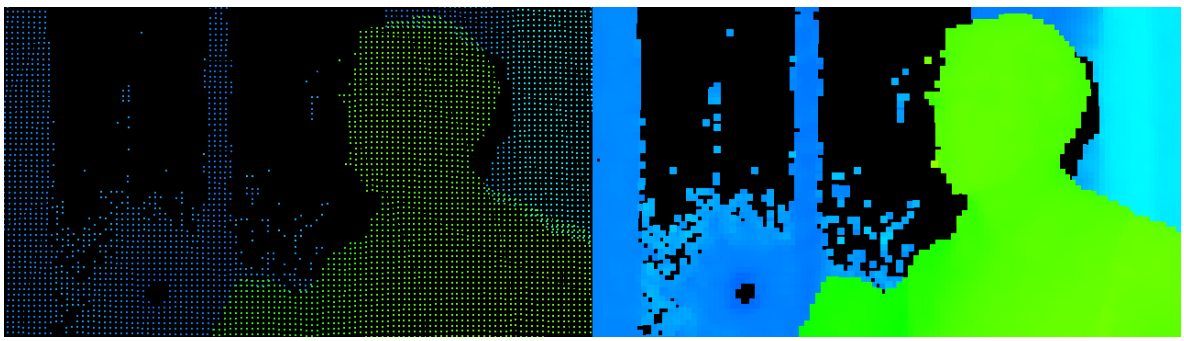


FIGURE 2.34 – Gauche : image de profondeur éparse. Droite : image de profondeur dense. Il est à noter que dans sur cette figure il ne s'agit pas exactement de la méthode d'interpolation décrite précédemment. Ici chaque pixel de profondeur est converti en *sprite* carré à la profondeur du pixel en question.

2.6 Gestion des occultations entre éléments réels et virtuels

2.6.1 Compositing basé sur la profondeur

L'étape finale de notre pipeline est le *compositing* de l'ensemble des éléments virtuels avec la scène réelle en gérant les occultations. Le point de vue du rendu de la scène virtuelle peut être défini soit manuellement pour un plan statique, soit calculé par un système de *tracking* de caméra pour un plan dynamique. Le but est que le point de vue du rendu, défini par une caméra virtuelle, corresponde exactement à la position et à la configuration optique de la caméra de tournage. De cette façon, l'image virtuelle et l'image issue de la caméra de tournage se superposent rigoureusement (voir figure 2.35).

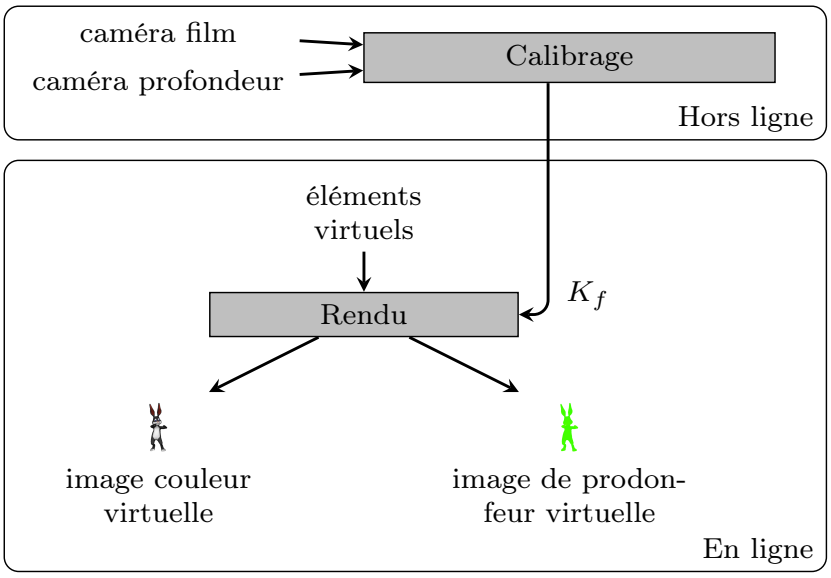


FIGURE 2.35 – Méthode de rendu des éléments virtuels. Les paramètres de la caméra film calculés pendant la phase de calibration servent à régler les paramètres de la caméra virtuelle qui génère le rendu des éléments virtuels. Le rendu génère une image couleur et une image de profondeur des éléments virtuels qui se superposent à l'image film.

Pour gérer les occultations, le moteur de jeux est configuré de façon à générer une carte de profondeur de la scène virtuelle D_v. La carte de profondeur réelle D_r est comparée à la carte de profondeur virtuelle D_v pour générer un masque binaire M_{rv} (voir équation 2.14.)

$$M_{rv}(\boldsymbol{u}) = \begin{cases} 1, & \text{si } D_r(\boldsymbol{u}) < D_v(\boldsymbol{u}) \\ 0, & \text{sinon} \end{cases} \tag{2.14}$$

Ce masque M_{rv} code si un pixel du rendu final I_{finale} provient de l'image couleur virtuelle C_v ou de l'image couleur réelle C_r. Le *compositing* final est produit en mélangeant les images I_v et I_r pondéré par le coefficient du masque M_{rv} à la manière de l'opérateur *over* [Bli94], tel qu'exprimé par l'équation 2.15.

$$I_{finale} = M_{rv} \times I_r + (1 - M_{rv}) \times I_v \tag{2.15}$$

La figure 2.36 illustre la méthode de *compositing*. Le masque peut être légèrement flouté de façon à atténuer les discontinuités abruptes entre le premier plan et l'arrière plan.

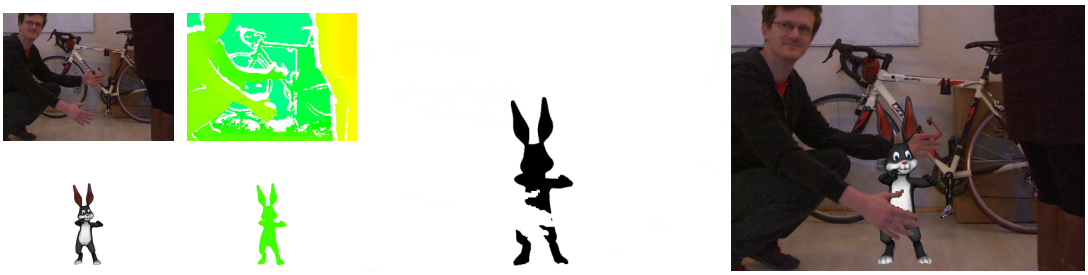


FIGURE 2.36 – Méthode de *compositing*. Gauche : première ligne : image couleur réelle I_r et image de profondeur réelle rectifiée D_r, seconde ligne : image couleur virtuelle I_v et image de profondeur virtuelle D_v. Milieu : masque binaire M_{rv}. Droite : *compositing* final I_{finale}.

2.6.2 Exemples d'applications

La figure 2.37 détaille le procédé mis en place sur une séquence d'images. L'image finale est obtenue en temps réelle. La méthode est implémentée dans le moteur de jeux *Unity 3D*.

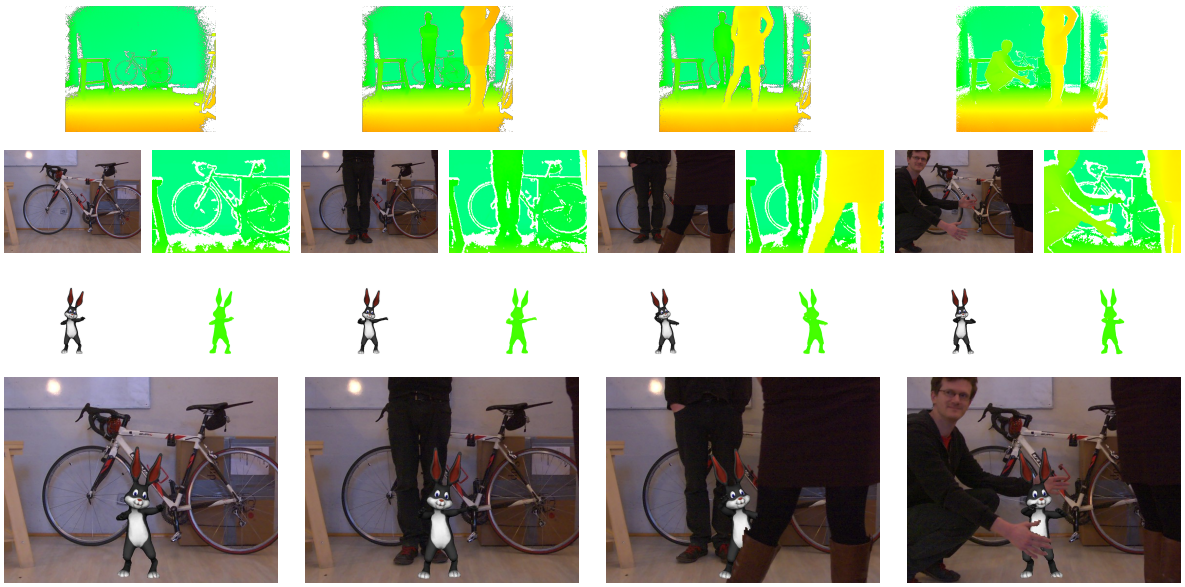


FIGURE 2.37 – Exemples de *compositing*. Première ligne : carte de profondeur réelle brute. Seconde ligne : image couleur réelle et profondeur réelle rectifiée. Troisième ligne : couleur virtuelle et profondeur virtuelle. Quatrième ligne : *compositing* final. Des occultations sont visibles sur la troisième et quatrième ligne.

Diverses applications comme l'interaction entre l'acteur réel et les éléments virtuels sont notamment possibles. La figure 2.38 illustre la destruction d'un mur virtuel avec gestion des occultations entre les briques virtuelles et l'acteur. La figure 2.39 montre une séquence où les objets virtuels sont en apesanteur. L'interaction est possible en utilisant la détection du squelette fourni par le SDK de la *kinect*.

FIGURE 2.38 – Exemple d'interaction réelle-virtuelle : destruction d'un mur de brique virtuel (depuis l'image en haut à gauche à celle en bas à droite). L'acteur recule quelques secondes en brisant le mur, il casse ensuite une autre partie avec son poignet droit. Filmé avec la caméra Blackmagic HD Studio et un objectif 14mm.

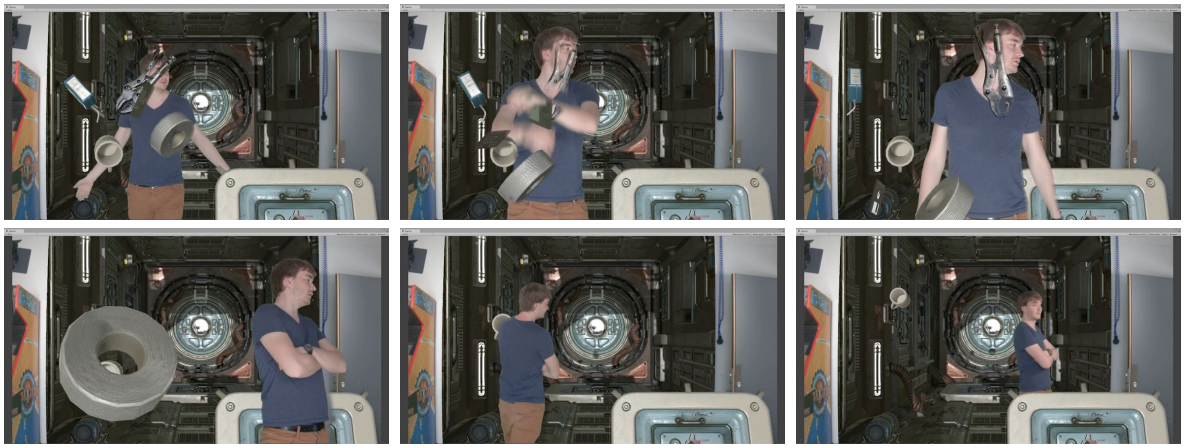


FIGURE 2.39 – Exemple d'interaction réelle-virtuelle : Déplacement d'objets virtuels en apesanteur (depuis l'image en haut à gauche à celle en bas à droite). Les élémnts 3D sont éclairés en utilisant une image HDR. Filmé avec la caméra Blackmagic HD Studio et un objectif 14mm.

La figure 2.40 met en jeu un système de *tracking* de caméra externe. Cela permet de mettre en mouvement la caméra film tout en conservant la synchronisation du rendu des éléments virtuels. La méthode mise en place permet de gérer les occultations entre les acteurs réels et le personnage virtuel.

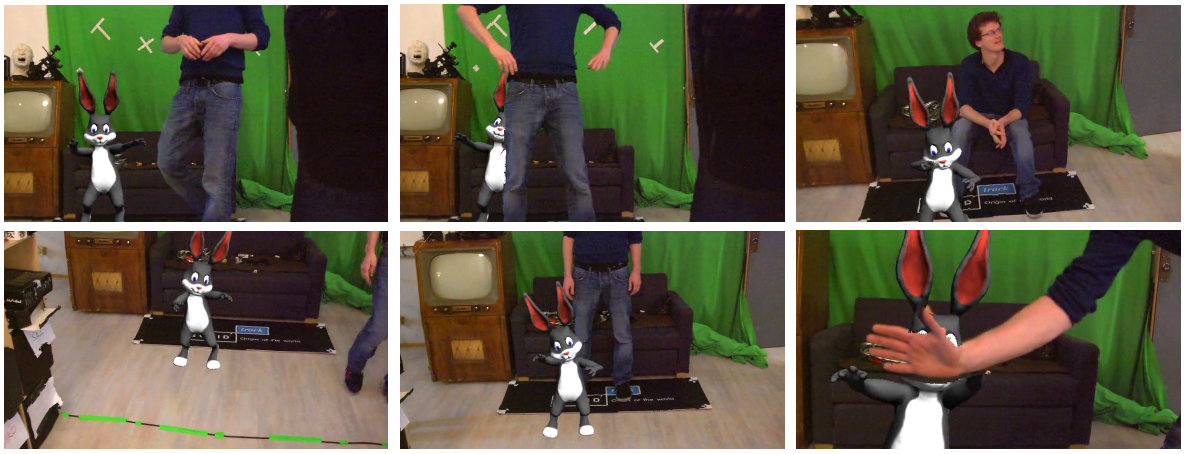


FIGURE 2.40 – Tracking de caméra et occlusions virtuelles-réelles. Filmé avec une caméra Panasonic AG-AF100A avec une lentille 18mm. Utilisation de la technologie *SolidTrack* pour le *tracking* de caméra temps réel.

Chapitre 3

Segmentation

Sommaire

3.1 Analyses des limites du système

Notre système basé uniquement sur la profondeur présente plusieurs limites qui dégradent la qualité visuelle du *compositing*. Celles-ci sont liées à la fois à des lacunes dans la carte de profondeur, à des phénomènes d'ombre et à une différence importante de résolution entre le capteur de la caméra de profondeur et celui de la caméra film. Nous détaillons chacune de ses limites dans la suite de notre propos.

3.1.1 Limites de la technologie ToF

La carte de profondeur produite par le module *Kinect* présente des lacunes. Il s'agit de zones où aucune information de profondeur n'est fournie. Plusieurs raisons peuvent expliquer ce phénomène. Tout d'abord, la plage de fonctionnement du module *Kinect 2* est limitée de $0,5$ à 5 mètres. Autrement dit, en dehors de ces bornes, le système n'est pas capable de fournir une information de profondeur. Il se peut également qu'un matériau présent dans la scène absorbe complètement les ondes infrarouges émises par le capteur ToF. Si c'est le cas, le module n'est pas en mesure de déduire une valeur de distance puisqu'aucune onde correspondant à cet endroit de la scène n'arrive en retour. De plus, aux contours des objets, les ondes infrarouges ne sont pas forcément renvoyées au capteur. C'est pour cette raison qu'une fine bande sans information apparaît autour des objets. Ces limites sont visibles sur la figure 3.1.

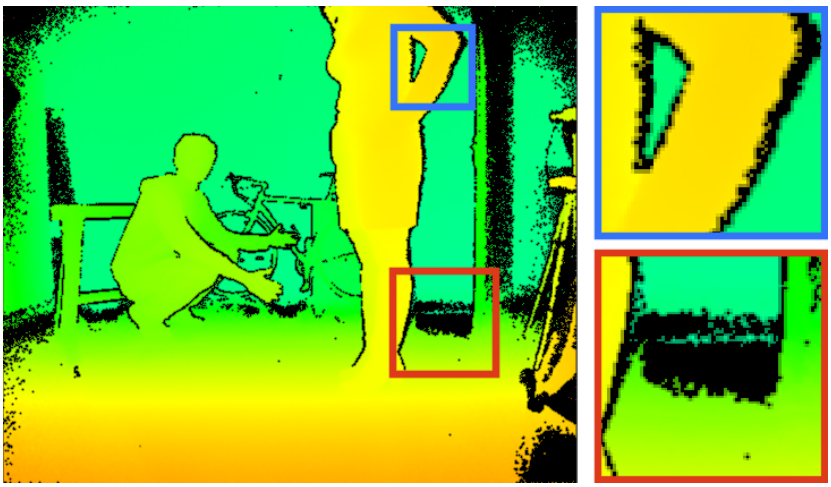


FIGURE 3.1 – Défauts de la carte de profondeur originale. Carré rouge : il n'y a pas d'information de profondeur dans cette zone car les ondes infrarouges ne sont pas renvoyées au capteur en cette zone (matière absorbante, la normale à la surface étant très éloignée de l'axe de vision du capteur). Carré bleu : une fine bande sans information apparaît aux contours des objets. On remarque que la bande est légèrement plus épaisse lors de la transition d'un objet proche à gauche (bras) à un objet lointain à droite (mur). La bande est d'autant plus épaisse que la distance est grande entre le premier et l'arrière plan.

Enfin, il n'y a pas d'information de profondeur aux contours de profondeur des objets dont

le mouvement est important. Ce phénomène est mis en évidence par la figure 3.2.

FIGURE 3.2 – Influence du mouvement sur la carte de profondeur. Il s'agit de cartes de profondeur directement issues du module *Kinect 2*. Haut : peu de mouvement. La carte de profondeur présente une fine bande sans information autour de la main. Bas : le mouvement est important. On note d'ailleurs un phénomène de flou de bougé au niveau de la main sur l'image couleur. Dans la carte de profondeur, une bande sans information de profondeur de plusieurs pixels de large apparaît autour de la main.

Ces zones sans information de profondeur posent problème puisqu'à ces endroits il n'est pas possible de comparer la profondeur de la scène réelle avec la profondeur de la scène virtuelle. L'application du *compositing* basée sur la profondeur est donc irréalisable. Néanmoins, il est possible d'établir des heuristiques telles que "*si pour un pixel de la scène réelle il n'y a pas d'information de profondeur alors ce pixel sera considéré comme provenant de l'image couleur réelle pour le compositing final*". En revanche, ce procédé ne fonctionne que pour un nombre restreint de scénarios.

3.1.2 Synchronisation temporelle et recalage spatial approximatifs

Les données de la carte de profondeur sont exprimées dans le référentiel de la caméra film afin d'être utilisées pour le *compositing*. Cependant, on remarque généralement un décalage entre l'image issue de la caméra film et la carte de profondeur recalée. Il peut s'agir d'une erreur d'alignement spatial causée par une mauvaise calibration des capteurs. Ce décalage peut également être causé par une synchronisation temporelle défaillante entre le flux des images films et le flux des images de profondeur.

3.1.3 Faible résolution de la carte de profondeur

Par ailleurs, même avec une calibration spatiale et une synchronisation temporelle correctes, les contours peuvent encore apparaître localement décalés. C'est la faible résolution du

capteur de profondeur qui est à mettre en cause. Celle-ci est interpolée pour s'accorder à celle de l'image film.

3.1.4 Parallaxe au sein du capteur *Kinect*

En outre, il faut noter qu'au sein du module *Kinect 2*, l'émetteur infrarouge est placé à gauche du récepteur. L'effet engendré par ce décalage horizontal, appelé aussi parallaxe, introduit une ombre au sein de la carte de profondeur. La parallaxe accentue l'épaisseur de la bande sans information sur certains contours de profondeur verticaux, spécifiquement concernant les contours de profondeur entre un objet proche et un objet lointain (objet proche à gauche et objet lointain à droite en regardant la scène). Ce phénomène est visible dans le carré bleu de la figure 3.1.

Pour comprendre ce phénomène, prenons l'exemple d'un objet placé devant un arrière plan. Plaçons une lampe devant l'objet afin de l'éclairer, on considère que c'est la seule source d'éclairage de la scène. En se plaçant au niveau de la lampe, un observateur voit une scène éclairée en chaque point. S'il se décale latéralement, par rapport au point de vue de la lampe, alors il voit une ombre apparaître sur le plan. La figure 3.3 illustre cette situation.

3.1.5 Parallaxe du *rig* bi-caméra

Le phénomène d'ombre, lié à la parallaxe entre le capteur de profondeur et la caméra film, est également présent lors du recalage spatial de la carte de profondeur dans le référentiel de la caméra film. Dans ce cas précis, la lampe de la figure 3.3 représente le capteur de profondeur tandis que l'observateur, qui est légèrement décalé, représente la caméra film. Le capteur de profondeur et la caméra film ne partagent pas le même point de vue sur la scène. Lorsque la caméra film se décale du capteur de profondeur, certaines surfaces de la scène deviennent visibles par la caméra film mais reste invisibles par le capteur de profondeur. Le recalage spatial de la carte de profondeur dans le référentiel de la caméra film, qui consiste finalement à faire une photographie de la carte de profondeur depuis le point de vue de la caméra film, met en évidence les zones d'ombres causées par le décalage inter-capteurs. Ces zones d'ombres, appelées également trous, apparaîssent lors d'un changement brut de profondeur suivant la direction définie par la position relative du capteur de profondeur vis-à-vis de la caméra film. La figure 3.4 illustre les effets du décalage inter-capteurs sur la carte de profondeur alignée. Il est bien entendu possible d'obstruer ces trous avec une méthode d'*inpainting* (voir annexe C).

L'ensemble des limites évoquées dans cette section se répercutent dans le *compositing* basé sur la profondeur et produisent un effet désagréable pour le spectateur puisque les objets de l'image film ne se mélangent pas de façon précise avec les éléments virtuels. Ces imprécisions liées au système sont illustrées dans la figure 3.5.

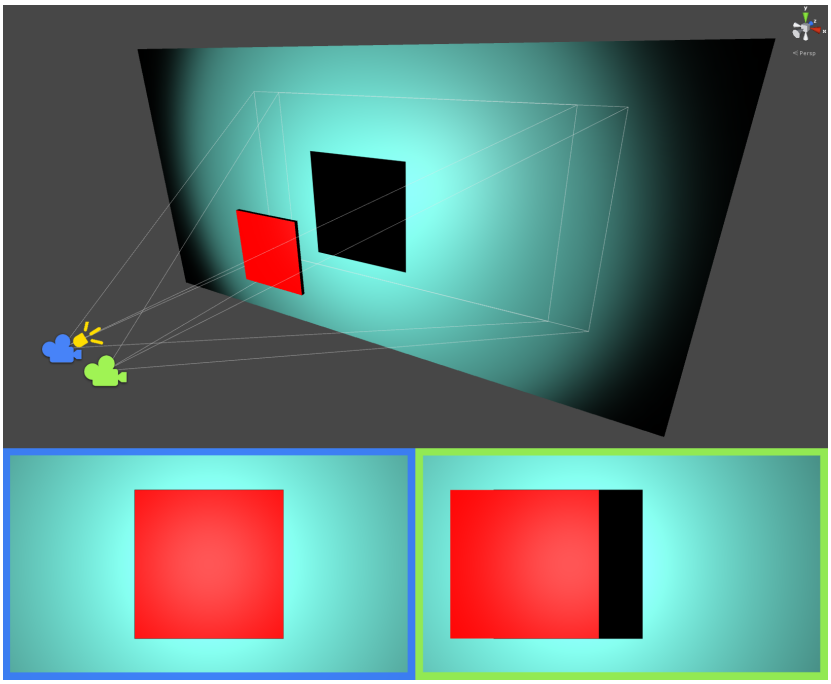


FIGURE 3.3 – Illustration des parallaxes qui provoque le phénomène d'ombre. La scène est composée d'un objet (illustré en rouge) placé devant un arrière plan (illustré en turquoise). La scène est éclairée par une source lumineuse ponctuelle (représentée ici par un spot jaune). Un observateur regarde la scène avec le même point de vue que la source lumineuse (caméra bleue) tandis qu'un autre observateur regarde la scène d'un point de vue légèrement décalé horizontalement (caméra verte). Les images de la scène depuis des points de vue différents sont représentées en bas de cette figure. On constate qu'il n'y a pas d'ompbre du point de vue de la caméra bleue, tandis qu'une ombre apparaît à droite de l'objet dans le point de vue de la caméra verte. Les caméras bleue et verte représentent successivement l'émetteur et le récepteur infrarouge de la caméra ToF, puis la caméra *Kinect* et la caméra film de notre système.

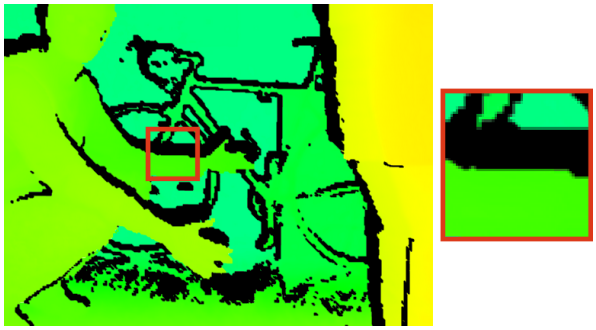


FIGURE 3.4 – Ombre dans la carte de profondeur alignée. La méthode de reprojection utilisée pour aligner spatialement la carte de profondeur à l'image film produit des trous. Ce sont des zones d'ombre qui résultent du décalage entre la caméra film et le capteur de profondeur.

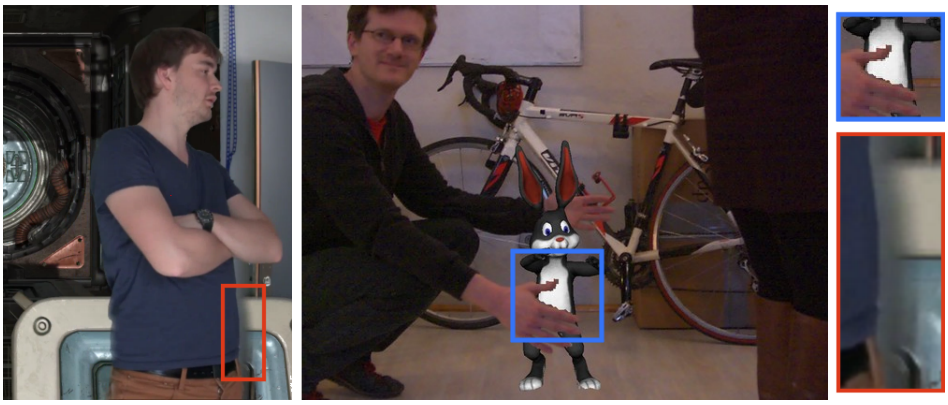


FIGURE 3.5 – Limites du *compositing* illustrées sur deux images. Gauche et centre : Décalage local de la carte de profondeur et de l'image film se répercutant dans le *compositing* de profondeur. Droite (haut et bas) : La faible résolution du capteur de profondeur produit des contours crénelés. Les contours de la carte de profondeur ne se superposent pas parfaitement aux contours de l'image couleur. Ce désalignement local est visible lors du *compositing*. Dans la vignette du bas, un flou gaussien est appliqué au masque de *compositing* pour atténuer le crénelage dans le *compositing*. On remarque cependant que le contour, qui est sensé suivre la silhouette de l'acteur, n'est pas bien localisé. Cela est dû au décalage entre l'image couleur et la carte de profondeur.

3.2 Détection de la silhouette

De nombreux scénarios mettent en jeu un acteur réel avec des objets virtuels. Pour cette raison, nous nous intéressons à la segmentation de la silhouette dans l'image couleur. C'est une situation courante lors du trucage d'un plan vidéo. Dans ce contexte, le terme segmentation signifie être en mesure de déterminer pour chaque pixel s'il appartient ou non à la silhouette de l'acteur dans l'image film. Obtenir un découpage propre de la silhouette permet un grand nombre d'applications telle que placer un acteur réel dans un décor virtuel en remplaçant l'arrière plan par exemple. Une application de remplacement d'arrière plan par un fond quelconque est fournie par *Microsoft*. La figure 3.6 donne un aperçu du résultat obtenu. Cette application met en évidence le fait que le découpage de la silhouette dans l'image couleur de la *Kinect* est globalement correct mais souffre localement d'imprécisions.

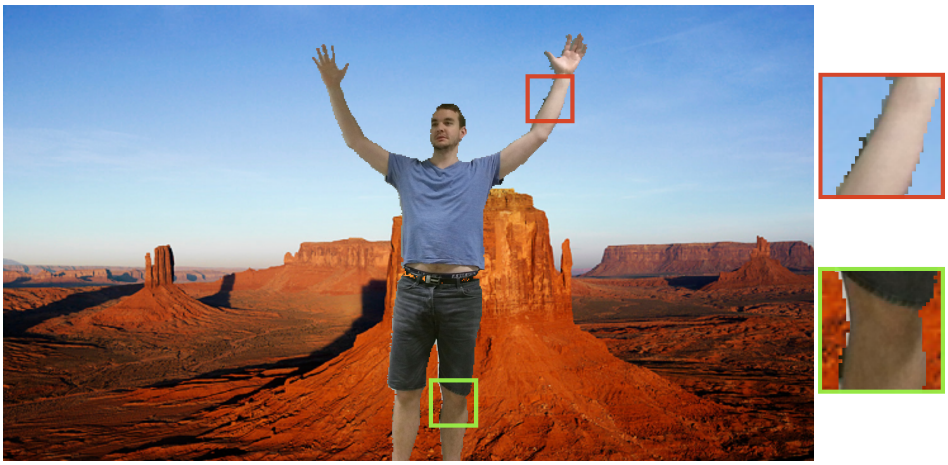


FIGURE 3.6 – Application de *compositing* d'arrière plan réalisé avec le logiciel *Coordinate Mapping Basics* de *Microsoft* en utilisant le module *Kinect*. On distingue que le découpage de la silhouette n'est pas parfait.

Ces imprécisions sont inhérentes aux limites listées précédemment dans cette section, à savoir la technologie ToF, les ombres, la calibration, la synchronisation temporelle, le calage spatiale et la faible résolution de l'image de profondeur par rapport à l'image couleur. Ces imprécisions se répercutent dans la segmentation de la silhouette et affectent la qualité du *compositing*.

Malgré ces imprécisions, ce genre d'application est possible puisqu'un algorithme (présent dans le SDK de la *Kinect*) est capable d'identifier la silhouette de l'acteur. Le fonctionnement de cet algorithme [SSK+13] a été détaillé dans la section 2.2.2 du chapitre précédent. Cet algorithme analyse l'image de profondeur et établit une segmentation de la silhouette en temps réel. L'image résultant de cette segmentation correspond à un masque binaire codant si un pixel de l'image appartient à la silhouette ou non. On appelle cette image le masque de la silhouette. La figure 3.7 illustre la détection de la silhouette sur une image de profondeur.

Dans ce chapitre nous proposons une méthode qui permet d'améliorer la segmentation de

FIGURE 3.7 – Détection de la silhouette à partir de l'image de profondeur. Gauche : image de profondeur. Droite : masque de la silhouette. La segmentation est réalisée en temps réel par l'algorithme présent dans le SDK de la *Kinect*.

la silhouette dans l'image couleur à partir de la segmentation de la silhouette fournie par le SDK de la *Kinect*. Dans notre cas, l'image couleur à laquelle on fait référence est l'image issue de la caméra film (et non celle du module *Kinect*). Le recalage spatial et temporel développé dans le chapitre 2 permet de mettre en correspondance le masque de la silhouette avec l'image film. Pour les raisons que nous avons évoquées précédemment, des imprécisions persistent localement dans la mise en correspondance des contours de l'image film et du masque de la silhouette. La figure 3.8 illustre ce décalage avec une image couleur issue de la caméra film.

FIGURE 3.8 – Décalage entre le masque silhouette recalé et l'image film. Le masque silhouette issue de l'image de profondeur est recalée spatialement avec l'image film en utilisant la méthode de reprojection. Le masque est superposé avec l'image film. On note localement des décalages de contours. Il est à noter qu'un filtrage gaussien permettrait de lisser le crénelage du masque, cependant celui-ci ne résout pas le décalage de contours avec l'image couleur.

Notre méthode propose de tenir compte des informations contenues dans l'image couleur afin d'améliorer la qualité de la segmentation de la silhouette. Pour fonctionner, cette méthode nécessite une segmentation initiale ainsi qu'une image couleur. Celle-ci est fournie par le masque de la silhouette généré le SDK de la *Kinect* puis recalé spatialement et temporellement. L'image couleur associée correspond à l'image film.

Notre méthode de segmentation de la silhouette est développée dans la section 3.5. Avant cela nous proposons un état de l'art des méthodes de segmentation qui utilisent conjointement la couleur et la profondeur dans la section suivante 3.3. Une description de la création de la base de données qui a servi à l'établissement de la méthode est donnée dans la section 3.4. Enfin une évaluation quantitative de la méthode est faite dans la section 3.6.

La qualité visuelle du *compositing* basé sur la profondeur, décrit dans le chapitre 2, n'est pas toujours probante. Ce niveau de qualité est inhérent aux limites du capteur de profondeur utilisé. Ces limites sont liées aux lacunes présentes dans la carte de profondeur ainsi qu'aux erreurs potentielles lors de la mise en correspondance des données de profondeur avec l'image couleur. Il en résulte que les contours des objets dans l'image film et l'image du capteur de profondeur ne sont pas toujours alignés. Ce constat est valable lorsque nous projetons le résultat de la segmentation de la silhouette issue de l'image de profondeur dans l'image couleur, il existe localement des décalages. Dans ce chapitre, nous proposons une méthode qui utilise les informations présentes dans l'image couleur dans l'objectif d'affiner la segmentation de la silhouette issue de l'image de profondeur. Par le biais de cette méthode, les contours du masque binaire représentant la silhouette sont alignés aux contours naturels de l'image couleur. Cette méthode est décrite dans la section 3.5.

3.3 État de l'art de la segmentation conjointe couleur et profondeur

Dans l'objectif de segmenter une image en utilisant les informations de couleur et de profondeur, plusieurs méthodes existent. La segmentation désigne l'action d'isoler un élément dans une image, ou de séparer l'image en zones homogènes selon certains critères. Isoler la silhouette d'un acteur dans une image afin de produire un masque correspond à une segmentation. Dans cette section nous dressons un état de l'art des méthodes relatives à la segmentation qui utilisent l'image couleur et l'image de profondeur. Un autre aspect qui n'est pas évoqué ici mais développé dans l'annexe C traite de l'amélioration des données de la carte de profondeur en s'appuyant sur l'image couleur associée.

Une méthode de segmentation basée sur la comparaison avec une image de référence est proposée dans [SBK+08]. L'auteur décrit un système complet pour réaliser le *compositing* entre des éléments virtuels et éléments réels. Un panorama à 360° de la scène où à lieu la captation est réalisé a priori. Ce panorama inclut une carte couleur et une carte de profondeur. Un appareil photo grand angle est utilisé pour la carte couleur, tandis qu'une caméra ToF montée sur une tête articulée est utilisée pour capter la profondeur. Il s'agit de réaliser une modélisation de l'arrière plan. Lors du traitement d'une séquence, le panorama est aligné et superposé à l'image courante. Les objets mobiles de la scène sont détectés en comparant la profondeur courante de la scène avec la profondeur de la carte du panorama. Cette comparaison permet d'obtenir une segmentation des objets au premier plan. Une amélioration de cette méthode est présentée dans [SK11]. Une acquisition a priori permet de décrire chaque pixel de l'arrière plan par un modèle multigaussiennes correspondant à la distribution de couleur et de profondeur. Lorsqu'un objet mobile se situe dans la scène, on mesure la distance entre le pixel courant et le modèle multigaussiennes. Si cette distance est inférieure à un seuil défini, cela signifie que le pixel courant appartient à l'arrière plan. Dans le cas contraire. cela signifie qu'il se situe au premier plan. Ce modèle permet d'obtenir des résultats plus convaincants qu'une comparaison de la profondeur ou que l'utilisation d'un modèle multigaussiennes uniquement ciblé sur la couleur. Cependant, l'auteur améliore encore la méthode en introduisant une notion de confiance dans la carte de profondeur. Une première méthode pour établir cette carte de confiance consiste à regarder la variance de la profondeur. La seconde méthode consiste à évaluer l'amplitude des rayons infrarouge qui reviennent à la caméra ToF. En effet, l'amplitude du signal reçu par la caméra de profondeur est un indicateur fiable de la confiance dans la mesure. Cette carte de confiance permet d'influencer la segmentation du premier plan et de l'arrière plan. Cette méthode fonctionne en temps réel mais nécessite une connaissance a priori de l'arrière plan.

Dans [DDH+15], l'auteur segmente une image de façon interactive en dessinant manuellement des segments sur l'image. La couleur d'un segment correspond à une classe à segmenter telle que arrière plan ou premier plan. Ces segments sont appelés *scribbles*. La méthode utilise les informations de couleur et de profondeur pour générer une segmentation. La profondeur est considérée comme le quatrième canal d'une image couleur classique RGB. Les *scribbles* permettent de générer un modèle d'apparence pour chaque classe. Ce modèle est basé sur trois

critères : une influence volumétrique (3D), une influence d'intensité couleur et une influence de profondeur. La méthode permet aussi d'avoir un retour sur l'influence d'un *scribble* sur la segmentation, ceci dans le but de limiter la non uniformité de distribution de *scribbles* dans l'image. La méthode définit un problème d'optimisation convexe en fonction du modèle d'apparence dont la résolution est effectuée avec l'algorithme primal-dual [CP11]. Cette méthode n'est pas temps réel puisque le temps de calcul pour une image de taille 640×480 pixels est d'environ 1 seconde, l'algorithme est cependant capable de gérer plus de deux classes pour la segmentation.

La méthode décrite dans [GJRW15] utilise également des *scribbles* afin de définir les régions à segmenter. Contrairement à la méthode précédente capable de gérer plusieurs classes de segmentation, cette méthode se limite à la segmentation de seulement deux classes. L'algorithme utilisé est le *Graph Cut* hiérarchique (une description du *Graph Cut* est donnée dans la section 3.5.6). Le terme qui définit la pénalité d'un pixel d'appartenir à l'une ou l'autre de ses classes prend en compte la couleur et la profondeur. La pénalité de couleur correspond à la probabilité d'un pixel d'appartenir à une classe en fonction de sa couleur. C'est en fait la ressemblance de couleur d'un pixel avec l'histogramme de la distribution de couleur de l'avant et l'arrière plan. Concernant la pénalité basée sur la profondeur il s'agit du ratio entre la distance géodésique d'un pixel à une classe et la somme des distances géodésiques du pixel à chacune des classes. Par le biais de cette méthode, des images en haute définition sont segmentées en $113ms$, cette segmentation nécessite tout de même une intervention afin de définir les *scribbles*. L'algorithme *graph cut* est également utilisé dans [DCSCO12] pour segmenter une image couleur et profondeur à partir de *scribbles*.

Dans l'une des méthodes précédemment citée, la profondeur est ajoutée comme le quatrième canal d'une image couleur. Un pixel de l'image est défini dans un espace à 6 dimensions : 2 dimensions pour la position du pixel dans l'image XY, 3 dimensions pour la couleur RGB et 1 dimension pour la profondeur D. Dans [BW09], l'auteur utilise l'algorithme *Mean Shift* dans cet espace à 6 dimensions afin de segmenter l'image. Il est important de noter que l'espace couleur utilisé pour cette segmentation est l'espace LUV. Afin de pondérer l'utilisation de la profondeur une carte de confiance est générée à l'instar d'une méthode précédente [SK11]. Une image du bruit de profondeur est générée par soustraction de l'image de profondeur originale à sa version lissée par un filtre bilatéral. Un pixel dont le bruit est faible indique une confiance importante dans la profondeur. Cette méthode est temps réel. Dans [DMZC10], l'auteur considère également pour chaque point de l'image un espace à 6 dimensions. Les 3 premières dimensions sont allouées pour la couleur, sachant que la méthode utilise l'espace normalisé CIELab. Ensuite, les 3 dimensions restantes sont utilisées pour les coordonnées 3D assignées au point considéré. Le pixel est projeté dans l'espace 3D en utilisant la matrice de projection inverse de la caméra et l'information de profondeur. Pour chaque point de l'image est associé un vecteur de dimension 6. Ce vecteur est normalisé en divisant chaque terme par l'écart type calculé sur toute l'image de la grandeur qu'il représente. Enfin, un algorithme de classification du type *k-means* est utilisé pour effectuer la segmentation. La méthode présentée dans [WFFD11] inclue également la couleur et la profondeur, cependant c'est la dérivée de la profondeur qui est utilisée. La segmentation est ensuite réalisée par la médiation d'un algorithme appelé *superparamagnetic clustering*.

Les méthodes que nous avons décrites fonctionnent pour une image. D'autres méthodes sont spécialement conçues pour traiter des séquences vidéos. Dans [GKHE10], l'auteur procède au préalable par une sur-segmentation de la séquence vidéo puis agrège les régions entre elles afin de générer la segmentation. Les régions sont générées en volume en considérant la similarité entre deux pixels voisins. Chaque pixel est connecté à ses 8 voisins au sein de l'image, mais il est également connecté à ses 18 voisins (9+9) dans l'image précédente et suivante de la vidéo. Cette méthode n'est pas temps réel. La méthode utilisée dans [APP+12] fait référence au modèle de *Potts* et à l'algorithme *Metropolis*. La segmentation d'une vidéo prend en compte la cohérence temporelle. La première image est segmentée complètement puis cette segmentation est transmise comme point de départ pour les image suivantes. Le flot optique est pris en compte pour le transfert de la segmentation d'une image à l'autre. A l'initialisation de l'algorithme, chaque pixel se voit attribué une valeur aléatoire de label. La profondeur et la couleur sont utilisées lors de la définition du terme qui représente l'interaction entre 2 pixels voisins. Plus le terme est faible, plus les pixels sont similaires ; a contrario plus le terme est fort plus les pixels présentent des différences. Si dans l'image de profondeur ces pixels sont trop éloignés (différence de profondeur supérieur à un seuil) alors ce terme prend la valeur maximale. De manière itérative, l'algorithme modifie la valeur du label de chaque pixel en fonction de l'ensemble des termes qui définit l'interaction entre les pixels. La profondeur joue aussi un rôle lors du transfert de la segmentation d'une image à l'instant t à l'instant $t+1$. Si la différence de profondeur entre un pixel à l'instant t et son pixel apparié via le flot optique à l'instant $t+1$ est supérieure à un seuil, alors le transfert de label n'a pas lieu. De même le transfert n'a pas lieu si la valeur de profondeur d'un pixel à l'instant $t+1$ est trop différente de la valeur de profondeur moyenne de tous les pixels appartenant à cette région à l'instant t. Les pixels qui n'ont pas reçu de transfert sont initialisés avec une valeur de label aléatoire.

La méthode [HBEC14] permet également de segmenter une vidéo en utilisant la couleur et la profondeur. Au lieu de construire un seul graphe dont les liens sont conjointement calculés à partir des valeurs de la profondeur et de la couleur de chaque pixel, cette méthode est basée sur la construction de deux graphes successifs. Le premier graphe consiste à effectuer une segmentation basée sur la profondeur et le mouvement uniquement. Chaque pixel représente un nœud du graphe. Chaque nœud est connecté à ses 8 voisins dans l'image courante (lien spatial) ainsi qu'à ses 18 voisins dans l'image $t-1$ et $t+1$ (lien temporel). Le lien temporel s'effectue en prenant en compte le flot optique. Une segmentation a lieu au sein du graphe en prenant en compte les liens entre les nœuds du graphe. Le deuxième graphe correspond à une sur-segementation basée uniquement sur la couleur et le mouvement en respectant les limites imposées par la première segmentation. Le lien entre deux nœuds de ce graphe correspond à la valeur absolue de la distance euclidienne des couleurs dans l'espace RGB. Le poids associé à un lien entre deux nœuds dont les zones sont différentes suite à la première segmentation sont étendus à l'infini. Ensuite, un graphe hiérarchique de la segmentation basé sur les histogrammes est créé. Pour chaque région, appelée *super-toxel*, un vecteur caractéristique est calculé en se basant sur la distribution de la couleur, de la 3D et le flot optique. Enfin, un *bipartite graph matching* est réalisé en mobilisant les 8 images précédentes afin d'assurer la cohérence temporelle de la segmentation.

Parmi les méthodes évoquées, celles qui fonctionnent en temps réel nécessitent un a priori fort tel que la modélisation de l'arrière plan. Dans notre cas nous ne disposons pas de cette information. D'autres méthodes utilisent une représentation de type graphe et cherche la coupe optimale permettant de séparer l'image en différentes zones. Cependant ces méthodes ne sont pas temps réel. Pour ces raisons, nous avons développé notre propre méthode. Celle-ci peut être implémentée à l'aide de *shaders* afin de bénéficier de la puissance du GPU tout en gardant une homogénéité avec le moteur de jeux vidéo. A l'instar de [HBEC14] notre méthode utilise successivement la profondeur puis la couleur. La profondeur permet d'extraire une première segmentation de la silhouette, puis la couleur sert à affiner la segmentation.

3.4 Constitution d'une base de données

Dans l'objectif de développer et de tester une méthode d'affinage des contours, nous utilisons des séquences enregistrées avec le système décrit dans le chapitre 2 afin de constituer une base de données. Dans cette section nous décrivons la production de ces séquences et le pré-traitement mis en œuvre pour obtenir des séquences d'images synchronisées spatialement et temporellement. Le but de ce chapitre et d'analyser les séquences afin de déterminer une méthode de segmentation adéquate. Le critère du temps réel n'est pas nécessaire pendant cette phase d'analyse. C'est pour cette raison que nous utilisons un logiciel tel que *Nuke* pour la préparation des données. Tous ces traitements sont à terme implémentés en temps réel au sein du moteur de jeux vidéo. Enfin, nous décrivons comment est créée la vérité terrain dont nous avons besoin pour comparer nos résultats. Celle-ci est produite à l'aide du logiciel *Sensarea* développé au sein du laboratoire.

3.4.1 Description des données

Un total de 56 clips a été enregistré, dont 7 séquences sont utilisées pour la calibration du système. Sur les 49 clips restants, les 8 clips les plus représentatifs ont été sélectionnés. Chacun de ces clips dure 5 secondes et présente une scène avec un acteur. Les conditions d'éclairage peuvent varier pendant la séquence et la caméra peut être fixe ou mobile selon le clip. Le tableau 3.1 donne un aperçu du contenu des 8 clips sélectionnés. Une illustration de ces clips est présentée dans l'annexe D. Il est à noter que pendant la prise de vue l'éclairage était de piètre qualité, engendrant un bruit d'acquisition important pour l'image film et beaucoup d'ombres bouchées. Ceci laisse à penser que sur un plateau, les résultats de segmentation RGBD seraient meilleurs.

Clip	**Caméra**	**Lieu**	**Acteur**	**Arrière plan**	**Remarques**
1	fixe	salle à manger	Pascal	bow window	prend un bonbon dans la boîte, chemise
2	mobile	salle à manger	Tim	piano, bow window	passage, pull blanc, panoramique cam
3	fixe	salle à manger	Amyrose	entrée salle à manger	avance vers la caméra, tshirt rose
4	fixe	salle à manger	Amyrose	piano	danse, mouvement des bras et mains, tshirt rose
5	mobile	salle à manger	Amyrose	piano	danse, tourne, tshirt rose
6	fixe	salon	Tim	cheminée, canapé	se lève, allume lampe, tshirt rayé
7	fixe	entrée	Pascal	porte entrée	met un manteau
8	fixe	cuisine	Amyrose	cuisinière	prend marmite, tshirt blanc

Tableau 3.1 – Description des 12 séquences sélectionnées pour constituer la base de données et évaluer une méthode d'affinage des contours.

Pour enregistrer ces séquences, un logiciel permettant de capturer le flux vidéo de la caméra film (via la carte d'acquisition SDI) et le flux des données du module *Kinect* a été développé. Le flux issu de la caméra film est capturé dans un fichier vidéo au format *.avi*. Ce fichier vidéo est ensuite converti en une séquence d'images par le biais de la librairie *ffmpeg* via un script. Le flux du module *Kinect* est capturé dans un fichier *.xef*. Il s'agit du format de fichier officiel proposé par le SDK de *Microsoft*. Un module a été développé en utilisant ce même SDK pour extraire a posteriori du fichier brut *.xef* les différents flux (profondeur, infrarouge,

silhouette) acquis par la *Kinect* sous forme de séquences d'images. La figure 3.9 illustre le processus d'acquisition des données et la conversion en séquences d'images.

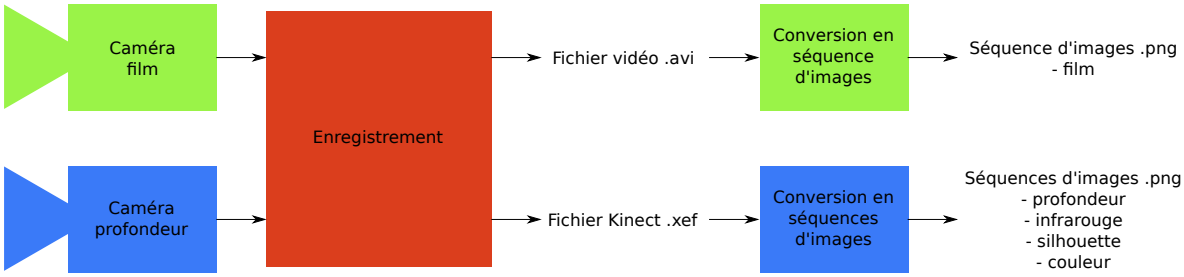


FIGURE 3.9 – Acquisition des données pour la constitution de la base de données. Un logiciel a été développé pour simultanément enregistrer la vidéo issue de la caméra film et le flux de la *Kinect*. Ensuite, la vidéo est convertie en une séquence d'images au format *png*. Un autre logiciel a été créé afin de transformer le fichier brut *xef* comportant les flux *Kinect* en des séquences d'images appropriées au format *png*.

Au total, pour chaque clip, on obtient 5 séquences d'images dont la liste figure dans le tableau 3.2. Il est à noter que bien que nous enregistrons le flux couleur de la *Kinect*, celui-ci n'est pas utilisé.

Flux	**Capteur**	**Dimension** (pixels)	**Précision** (bits)	**Canal**
film	Blackmagic HD Studio	1920×1080	8	3
profondeur	Kinect 2	512×480	16	1
infrarouge	Kinect 2	512×480	16	1
silhouette	Kinect 2	512×480	8	1
couleur	Kinect 2	1920×1080	8	3

Tableau 3.2 – Différents flux enregistrés lors de la captation.

3.4.2 Synchronisation temporelle

Le module *Kinect* enregistre dans le fichier *.xef* un *timestamp* pour chacune des images qu'il capture. Lors de l'acquisition des séquences, le programme d'enregistrement note le *timestamp* du déclenchement de la caméra film et du module *Kinect*. En combinant ces informations il est possible d'horodater chaque image de chaque flux. A chaque image issue du flux caméra film on associe l'image de profondeur, l'image infrarouge, l'image silhouette et l'image couleur dont le *timestamp* est le plus proche. Les images de chaque flux sont numérotées pour être synchronisées temporellement. La figure 3.10 illustre un exemple d'images pour chacun des flux enregistrés en respectant les proportions.

FIGURE 3.10 – Exemple de données extrait des différents flux. Haut : image extraite du flux film. Milieu gauche : image de la profondeur issue du flux *depth* (illustration en fausses couleurs pour permettre de distinguer les niveaux de profondeur). Milieu centre : image infrarouge issue du flux *infrared* (histogramme modifié pour permettre la visualisation). Milieu droite : masque binaire correspondant à la silhouette extrait du flux silhouette. Bas : image couleur de la *Kinect* issue du flux *color*.

3.4.3 Recalage spatial

Le flux d'images silhouettes est obtenu en utilisant l'algorithme décrit dans [SSK+13] fournie par le capteur *Kinect*. Une description du fonctionnement de cet algorithme est donnée dans la section 2.2.2 du chapitre 2. La silhouette est détectée dans le référentiel de la caméra de profondeur. L'objectif du recalage spatial est de transformer cette image pour obtenir la silhouette dans le référentiel de la caméra film. Il est important d'obtenir cette image puisqu'elle sert de point d'entrée pour l'algorithme d'affinage des contours. Pour cela nous utilisons la méthode de reprojection explicitée dans le chapitre 2. Les paramètres intrinsèques et extrinsèques, nécessaires à la reprojection, sont extraits à partir des séquences d'images enregistrées spécialement pour la calibration. Cette méthode de reprojection est appliquée aux séquences d'images en utilisant le logiciel *Nuke*. En effet, le logiciel *Nuke* possède des modules de gestion de la 3D qui permettent d'implémenter ce procédé. La figure 3.11 illustre l'utilisation du logiciel.

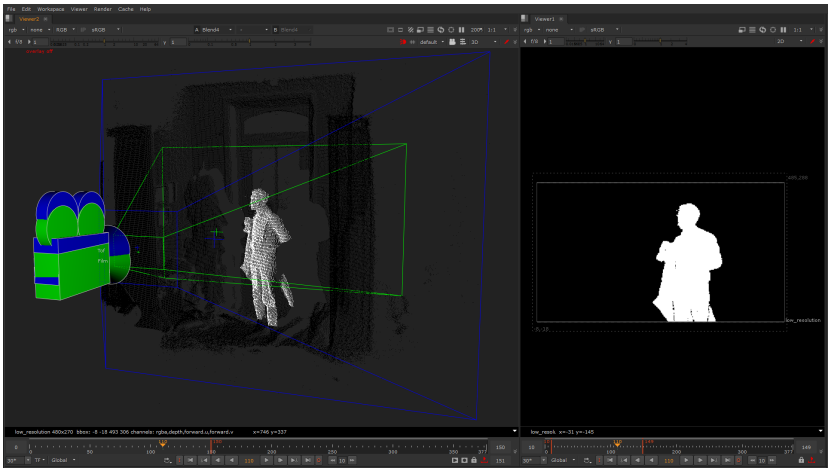


FIGURE 3.11 – Capture d'écran du logiciel *Nuke*. Partie gauche : Vue 3D, on distingue la caméra Tof (bleue) ainsi que la caméra film (verte). La scène est projetée sous forme d'un nuage de points 3D à partir de l'image de profondeur et des paramètres intrinsèques de la caméra ToF. Partie Droite : le nuage de point est projeté dans le point de vue de la camera film. Ici il s'agit du rendu de la silhouette.

En s'appuyant sur cette méthode, nous générons également les séquences d'images de profondeur et d'infrarouge recalées spatialement avec la caméra film. La figure 3.12 permet de visualiser une image de chaque flux synchronisé spatialement.

3.4.4 Vérité terrain avec *Sensarea*

L'image de la silhouette est un masque binaire brut. Ce masque est synchronisé temporellement et spatialement avec l'image film. Il ne se superpose pas parfaitement à la silhouette présente dans l'image film. Notre méthode, décrite dans la section suivante, permet d'adapter et d'affiner le contour du masque brut pour l'adapter aux contours réels du personnage dans

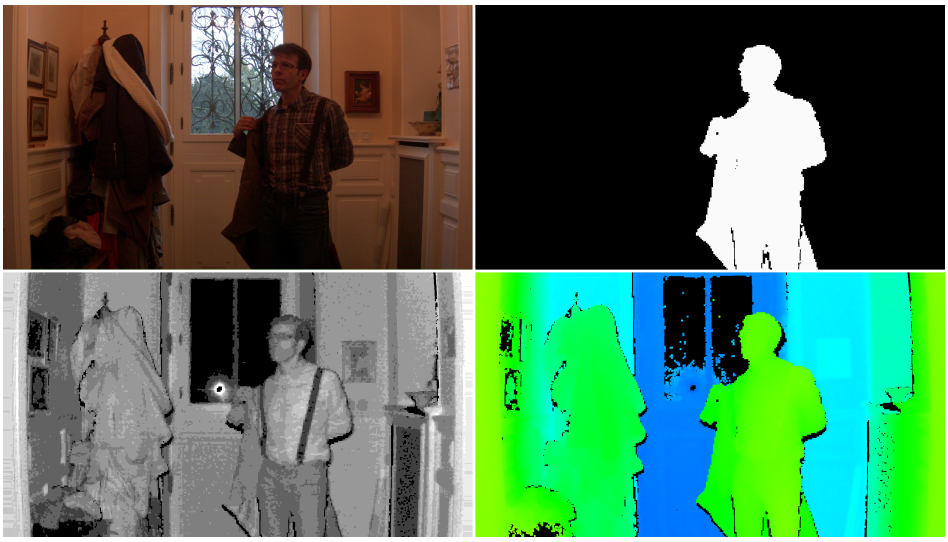


FIGURE 3.12 – Exemple de données alignées temporellement et spatialement. Haut gauche : image film. Haut droite : masque binaire correspondant à la silhouette, recalé spatialement. Bas gauche : image infrarouge recalée spatialement (histogramme modifié pour permettre la visualisation). Bas droite : image de la profondeur recalée spatialement (illustration en fausses couleurs pour permettre de distinguer les niveaux de profondeur).

l'image film HD. Dans l'objectif d'évaluer la qualité des masques produits, une référence est nécessaire. Pour cette raison une vérité terrain a été établie. Dans chacune des 8 séquences sélectionnées nous avons créé les masques binaires correspondant à la silhouette de l'acteur. La méthode consiste à dessiner manuellement sur l'image couleur le masque correspondant à la silhouette. Il appartient à l'utilisateur de juger où se situe le contour dans l'image film. Pour effectuer cette manipulation, nous utilisons le logiciel *Sensarea* [Ber14] développé au laboratoire GIPSA-lab. Ce logiciel permet d'accélérer le processus puisqu'il permet assez souvent de détecter automatiquement les contours. Le logiciel utilise l'algorithme de *Canny* [Can86] pour détecter des contours dans l'image. Lorsque l'utilisateur déplace sa souris approximativement autour de la silhouette, le logiciel sélectionne le contour de *Canny* le plus proche. Lorsqu'il n'y a pas de contour de *Canny* disponible l'utilisateur doit manuellement dessiner le contour. *Sensarea* permet d'accélérer le processus de création de la vérité terrain et nous assure que la silhouette est délimitée au pixel près à un contour naturel de l'image. L'utilisation de *Canny* permet de localiser précisément le contour sur la norme maximale du gradient et ainsi optimiser la précision du masque en réduisant l'erreur de manipulation de l'outil de découpage utilisé par l'utilisateur.

Dans chacun des clips sélectionnés nous avons extrait un échantillonnage d'images film à traiter. Cet échantillonnage nous sert à produire une vérité terrain en s'assurant que l'image silhouette associée est valide. En effet l'image silhouette est l'entrée de notre algorithme d'affinage des contours. Il peut arriver que l'algorithme de détection de la silhouette ait échoué. Cela se produit par exemple si la différence de profondeur entre la silhouette et l'arrière plan de la scène n'est pas suffisamment marquée, ou bien si la silhouette est connectée physiquement à un autre objet dans l'image de profondeur. Nous avons écarté ces images de notre sélection.

La figure 3.13 présente une capture d'écran du logiciel *Sensarea*.

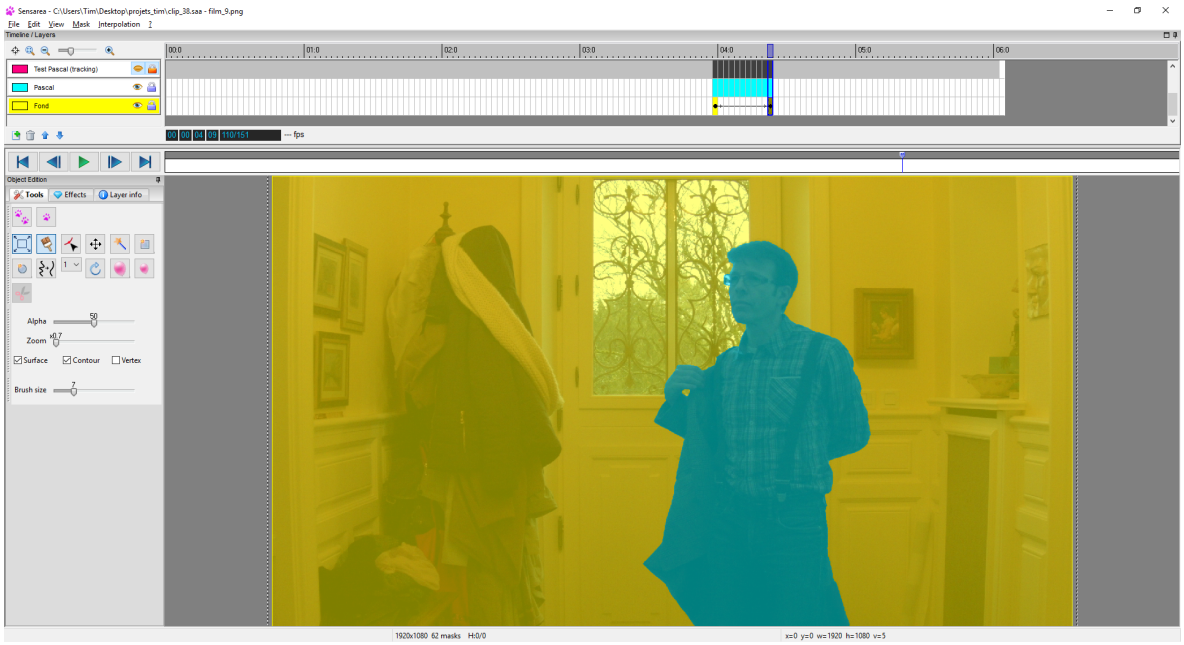


FIGURE 3.13 – Capture d'écran du logiciel *Sensarea*. L'utilisateur définit manuellement les contours des zones à segmenter. Afin d'accélérer la création de la segmentation, le logiciel recherche automatiquement le contour de *Canny* le plus proche de la souris.

La base de données qui est constituée comprend 8 clips de 5 secondes. Chacun de ses clips comprend un acteur dans une situation particulière. L'éclairage peut varier pendant le clip. La caméra est fixe ou mobile. Un clip est composé de plusieurs séquences d'images qui sont prétraitées afin d'être synchronisées temporellement et recalées spatialement, l'image film étant le référentiel maître. On trouve pour chaque clip les séquences d'images suivantes : film, profondeur, infrarouge, silhouette et couleur. La vérité terrain de la segmentation de la silhouette de l'acteur est produite pour une dizaine d'images dans chacun de ces clips. Celle-ci est réalisée par le logiciel *Sensarea*. Cette base de données sert à expérimenter et mesurer la qualité de la méthode de segmentation de la silhouette qui est décrite dans la section suivante.

3.5 Segmentation de la silhouette

Nous disposons d'une segmentation grossière de la silhouette établie à partir de la carte de profondeur de la *Kinect*. Cette segmentation est synchronisée temporellement et recalée spatialement dans le référentiel de la caméra film. L'image du masque de la silhouette à laquelle correspond cette segmentation est appelée segmentation brute. Lorsqu'on isole la silhouette dans l'image film en utilisant la segmentation brute on s'aperçoit que des décalages et des imprécisions subsistent localement (voir figure 3.14).

FIGURE 3.14 – *Compositing* brut. Gauche : segmentation brute de la silhouette. Droite : *compositing* de l'image film avec la segmentation brute. On note un mauvais alignement local des contours sur certaines portions de la silhouette.

Dans cette section nous décrivons une méthode qui utilise l'image couleur afin d'affiner la segmentation brute de la silhouette. Cette méthode se structure en deux étapes. La première étape est la création de deux masques représentant la silhouette et l'arrière plan sur lesquels nous allons appliquer une diffusion anisotropique guidée par l'image couleur. La seconde étape consiste à calculer une probabilité d'appartenance au masque de la silhouette à partir des images générées par la diffusion. Cette probabilité permet de déterminer une segmentation affinée de la silhouette. Nous allons commencer par dépeindre de manière générale le principe de la méthode au travers de ces deux étapes avant de rentrer davantage dans les détails.

Comme nous le verrons, la diffusion anisotropique est réalisée par filtrage guidé. Ce type d'approche est intéressant dans notre contexte car ce filtrage peut être implémenté en temps réel. Nous évaluons 4 filtres différents pour effectuer cette opération. Pour cette raison, nous détaillons chacun de ces filtres dans la suite de cette section.

Enfin, la méthode que nous décrivons est locale, c'est ce qui la différencie d'autres méthodes évoquées dans la section 3.3 telle que la méthode du *Graph Cut*. Une méthode de type *Graph Cut* a l'avantage d'être globale, c'est-à-dire qu'elle optimise une fonction d'énergie sur l'ensemble de l'image afin de proposer une segmentation optimale. Cet avantage a souvent lieu au détriment du temps de calcul, ce qui l'éloigne du temps réel. Néanmoins, nous implémentons une variante de notre méthode qui utilise le *Graph Cut* afin d'avoir un point de comparaison en terme de qualité. Le résultat obtenu par le *Graph Cut* peut être considéré la plupart du temps comme la qualité maximale qui peut être obtenue avec une méthode de filtrage locale.

3.5.1 Principe de la méthode

A partir de la segmentation grossière fournie par la *Kinect* nous créons 2 masques binaires qui serviront à initialiser la diffusion anisotropique. Ces deux masques sont obtenus par des opérations morphologiques sur l'image de la segmentation brute. Il s'agit du masque brut du premier plan, appelé $M_{brut,fg}$ et le masque brut de l'arrière plan $M_{brut,bg}$. Dans notre cas le premier plan représente l'objet à segmenter, la silhouette, tandis que l'arrière plan correspond à tout ce qui n'est pas la silhouette. Tout d'abord, une fermeture morphologique est effectuée afin d'enlever les trous de quelques pixels de large dans la segmentation grossière. A partir de cette image, une érosion permet d'obtenir $M_{brut,fg}$, tandis que $M_{brut,bg}$ correspond au négatif de la dilatation de cette image. Le figure 3.15 illustre les deux masques obtenus. Pratiquer une érosion et une dilatation permet finalement de définir une bande d'incertitude dans laquelle peut se trouver le contour définitif.

FIGURE 3.15 – Opérations morphologiques sur l'image de la segmentation brute. Gauche : image de la segmentation brute de la silhouette. Centre : masque du premier plan $M_{brut,fg}$ obtenu par une érosion. Droite : masque brut de l'arrière plan $M_{brut,bg}$ qui correspond au négatif de la dilatation.

Une diffusion anisotropique du masque $M_{brut,fg}$ ainsi que du masque $M_{brut,bg}$ est réalisée pour adapter au mieux le masque initial $M_{brut,fg}$ (respectivement $M_{brut,bg}$) aux contours de la silhouette (resp du fond) dans l'image couleur. Cette diffusion est influencée par les informations présentes dans l'image film. En combinant les cartes de diffusion (celle du premier plan et celle de l'arrière plan) nous pouvons établir en chaque pixel une probabilité d'appartenance à la silhouette. Ce point est détaillé plus loin.

La diffusion anisotropique du masque $M_{brut,fg}$ permet de créer une image J_{bg} représentant pour chaque pixel de l'image haute résolution (correspondant à l'image film) la probabilité qu'il a d'appartenir au premier plan. De la même manière, une image J_{fg} est créée par la diffusion anisotropique du masque $M_{brut,bg}$. La valeur d'un pixel de l'image J_{fg} quantifie la probabilité qu'il a d'appartenir à l'arrière plan.

La diffusion anisotropique est réalisée par un filtrage des masques bruts ($M_{brut,fg}$ et $M_{brut,bg}$) guidé par l'image couleur I_{col}. Des filtres issus de la famille des *edge preserving filter* [KCLU07] sont utilisés puisqu'ils ont la capacité de filtrer une image en respectant les contours présents dans une autre image, appelée image guide. Quatre filtres sont mis en concurrence :

- filtre bilatéral joint (**BF**) [KCLU07]
- filtre guidé (**GF**) [HST13]

- domain transform filter (**DT**) [GO11]
- adaptive manifold filter (**AM**) [GO12]

Dans notre contexte les masque bruts $M_{brut,fg}$ et $M_{brut,bg}$ sont filtrés en étant guidé par les contours réels de la silhouette présents dans l'image couleur. Soit f l'application générique qui réalise le filtrage (équation 3.1). Cette application f s'appuie sur l'un des quatre filtres précédemment cité pour réaliser la diffusion.

$$\begin{cases} J_{fg} = f(M_{brut,fg}, I_{col}) \\ J_{bg} = f(M_{brut,bg}, I_{col}) \end{cases} \tag{3.1}$$

Comme nous l'avons précisé, la diffusion est basée sur les contours présents dans l'image couleur. Deux paramètres permettent d'ajuster cette diffusion, il s'agit de σ_s et σ_r. Le détail de chacun des filtres et de leurs paramètres est donné dans la suite de cette section, néanmoins nous allons décrire brièvement l'influence de ces paramètres. Le paramètre σ_s définit la taille de la fenêtre spatiale (en pixel) dans laquelle deux pixels auront une influence l'un sur l'autre pour le filtrage. Il s'agit en quelque sorte du rayon d'action de la diffusion. Le paramètre σ_r traduit le seuil de tolérance acceptable entre deux pixels en terme d'intensité. Il s'agit du rayon d'intensité qui définit si deux pixels sont semblables. Ce paramètre conditionne l'utilisation d'un pixel pour influencer la diffusion, il règle la force de la diffusion. Une valeur forte de σ_r (valeur proche de 1) revient à effectuer un flou gaussien, tandis qu'une valeur faible (proche de 0) permet une diffusion localisée aux zones homogènes de l'image guide.

A partir des images de diffusion J_{fg} et J_{bg} il est possible de calculer une probabilité d'appartenance à la silhouette en chaque pixel d'indice i, cette probabilité est notée $P_{silhouette}(i)$ et s'établit comme suit :

$$P_{silhouette}(i) = \frac{J_{fg}(i)}{J_{fg}(i) + J_{bg}(i)} \tag{3.2}$$

Le masque binaire affiné M_{affine} correspondant à la segmentation affinée de la silhouette est construit en évaluant la probabilité $P_{silhouette}(i)$ pour chaque pixel. La valeur de M_{affine} est 1 si $P_{silhouette}(i) \geqslant 0.5$, elle est de 0 autrement. Cela se traduit mathématiquement dans l'équation 3.3.

$$M_{affine}(i) = \begin{cases} 1, & \text{si } P_{silhouette}(i) \geqslant 0.5 \\ 0, & \text{sinon} \end{cases} \tag{3.3}$$

Dans la suite de cette section nous présentons en détail chacun des filtres utilisés pour réaliser la diffusion anisotropiquue. Pour chaque filtre nous expliquons le principe et présentons un exemple des cartes de diffusion obtenues. Ces filtres sont mis en concurrence dans la section 3.6 afin de déterminer celui qui offre les meilleurs résultats pour la segmentation de la silhouette dans les séquences de tests présentées dans la section 3.4.

3.5.2 Filtre bilatéral joint

Le filtrage bilatéral [TM98] [YTA09] est basé sur la convolution d'un noyau de filtrage avec chaque pixel de l'image à filtrer. Le filtre bilatéral permet de lisser les zones homogènes d'une image, contrairement à un filtre gaussien qui filtre uniformément l'image sans tenir compte de l'intensité des pixels. Les valeurs des coefficients du noyau sont dépendantes de la position dans le noyau ainsi que de la valeur de l'intensité des pixels filtrés. Le noyau est donc calculé pour chaque pixel de l'image à traiter, de sorte qu'il n'est pas constant comme l'est le noyau d'un filtre gaussien. Cette évaluation systématique rend son implémentation coûteuse. L'utilisation du filtre bilatéral appliquée à une image est illustrée dans la figure 3.16.

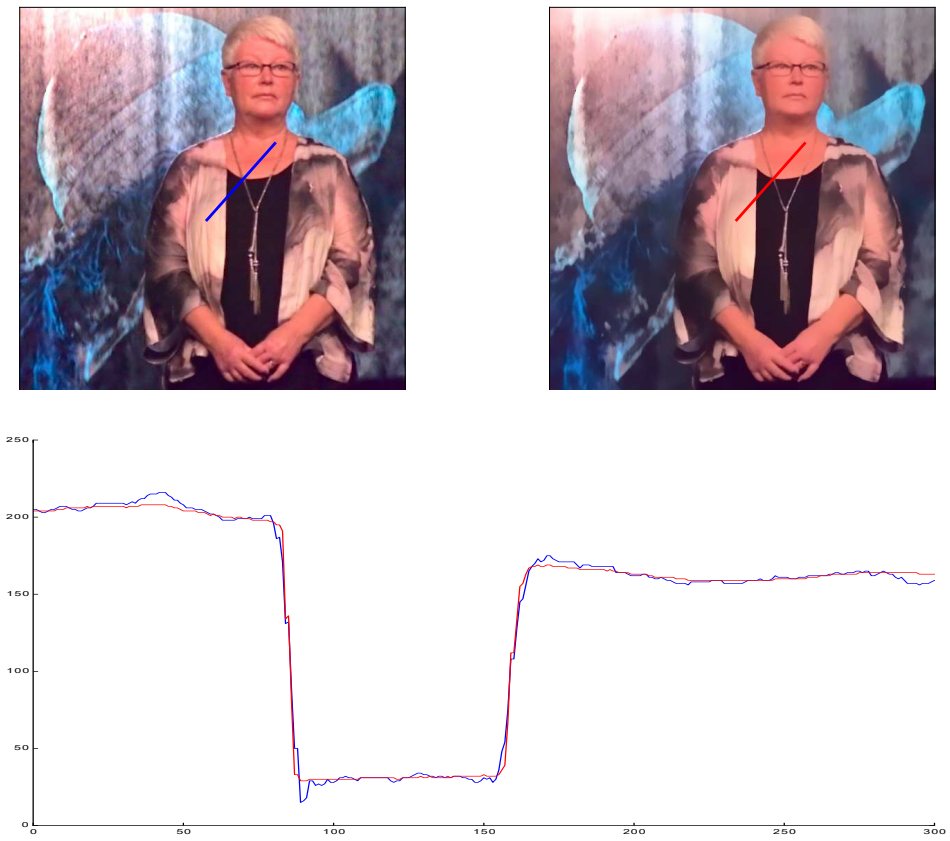


FIGURE 3.16 – Filtre bilatéral appliqué sur l'image de Lena. On peut voir en bas le profil de l'intensité le long d'un segment de coupe. En bleu le profil le long du segment sur l'image original. En rouge le profil le long du segment sur l'image après le filtrage bilatéral. Le filtre bilatéral lisse les zones homogènes tout en conservant les variations importantes.

L'équation 3.4 décrit mathématiquement le filtrage bilatéral au pixel de coordonnées i d'une image I.

$$J(i) = \frac{1}{W(i)} \sum_{j \in \omega_i} g_s(i,j,\sigma_s) \times g_r(I(i), I(j), \sigma_r) \times I(i) \tag{3.4}$$

avec :

$$g_s(i,j,\sigma_s) = \exp(-\frac{\|i-j\|_2^2}{2\sigma_s^2}) \tag{3.5}$$

$$g_r(I(i), I(j), \sigma_r) = \exp(-\frac{\|I(i)-I(j)\|_2^2}{2\sigma_r^2}) \tag{3.6}$$

$$W(i) = \sum_{j \in \omega_i} g_s(i,j,\sigma_s) \times g_r(I(i), I(j), \sigma_r) \tag{3.7}$$

$I(i)$ représente la valeur du pixel i dans l'image originale, tandis que $J(i)$ représente la valeur filtrée. L'ensemble des coordonnées des pixels de la fenêtre centrée sur i est noté ω_i. L'indice j représente l'indice d'un pixel dans la fenêtre ω_i centrée sur i. La notation $\|.\|_2^2$ définit la distance euclidienne au carré. Le terme $g_s(i,j,\sigma_s)$ permet de calculer un poids quantifiant l'influence spatiale. Il s'agit d'une gaussienne de variance σ_s^2 évaluant la distance entre le pixel i et le pixel j. Le terme $g_r(I(i), I(j), \sigma_r)$ permet de calculer un poids quantifiant l'influence de l'intensité (*range*). Il s'agit également d'une fonction gaussienne dont la variance est σ_r^2. Ce poids permet de limiter l'influence des pixels dont la valeur $I(j)$ est trop éloignée de celle de $I(i)$. Enfin le terme $W(i)$ permet de normaliser le résultat du filtrage bilatéral.

Le filtre bilatéral joint [KCLU07] est une évolution du filtre bilatéral. Il consiste à évaluer le poids d'intensité g_r non pas sur l'image à filtrer I mais sur une image différente, appelée image guide $\tilde{I}$. L'équation du filtre bilatéral joint est donnée en 3.8.

$$J(i) = \frac{1}{W(i)} \sum_{j \in \omega_i} g_s(i,j,\sigma_s) \times g_r(\tilde{I}(i), \tilde{I}(j), \sigma_r) \times I(i) \tag{3.8}$$

On utilise cette équation pour calculer la diffusion du premier plan J_{fg} et de l'arrière plan J_{bg} (équations 3.9) afin d'appliquer la méthode décrite précédemment. Les images à filtrer sont $M_{brut,fg}$ et $M_{brut,fg}$ et l'image qui sert de guide est l'image couleur I_{col}.

$$\begin{cases} J_{fg}(p) = \frac{1}{W(i)} \sum_{j \in \omega_i} g_s(i,j,\sigma_s) \times g_r(I_{col}(i), I_{col}(j), \sigma_r) \times M_{brut,fg}(i) \\ J_{bg}(p) = \frac{1}{W(i)} \sum_{j \in \omega_i} g_s(i,j,\sigma_s) \times g_r(I_{col}(i), I_{col}(j), \sigma_r) \times M_{brut,bg}(i) \end{cases} \tag{3.9}$$

Un exemple de diffusions anisotropiques obtenues avec le filtre bilatéral est présenté sur la figure 3.17.

FIGURE 3.17 – Cartes de diffusion issues du filtre bilatéral joint. Gauche : carte de diffusion de l'arrière plan J_{bg}. Droite : carte de diffusion du premier plan J_{fg}.

3.5.3 Filtre guidé

Le filtre guidé est introduit dans [HST13]. A l'instar du filtre bilatéral joint, ce filtre permet de filtrer une image en utilisant une image guide comme modèle. La particularité de ce filtre est qu'il préserve le gradient de l'image filtre dans l'image filtrée. Il est par exemple utilisé dans [LGL12] pour filtrer des images de profondeur. Des travaux présentés dans [WBH11] décrivent une implémentation au GPU.

Le point clé de ce filtre est de considérer que localement l'image filtrée est une fonction affine de l'image guide. On considère que dans une fenêtre ω_k centrée sur le pixel k, la valeur $J(i)$ du pixel à l'indice i de l'image filtrée J obéit à un modèle linéaire local par rapport à la valeur $\tilde{I}(i)$ du pixel d'indice i dans l'image guide $\tilde{I}$. On a l'équation 3.10 :

$$J(i) = a_k \tilde{I}(i) + b_k, \forall i \in \omega_k \tag{3.10}$$

Les termes a_k et b_k du modèle linéaire sont choisis de façon à minimiser la quantité E qui est définie dans l'équation 3.11. Il s'agit de minimiser l'écart entre la valeur du pixel filtré $J(i)$ et la valeur du pixel associé dans l'image source $I(i)$. Le paramètre ϵ est un terme de régularisation qui sert à définir le seuil au delà duquel les variations de l'image guide seront préservées.

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k \tilde{I}(i) + b_k - I(i))^2 + \epsilon a_k^2) \tag{3.11}$$

Résoudre l'équation 3.11 permet de déterminer une valeur pour a_k et b_k pour la fenêtre ω_k. Le pixel à la position i est impliqué dans toutes les fenêtres ω_k qui contiennent i. En prenant la moyenne de toutes les solutions a_k et b_k dans laquelle i est impliqué, on obtient la solution $\bar{a_i}$ et $\bar{b_i}$. Ainsi, le pixel filtré $J(i)$ est défini comme tel :

$$J(i) = \bar{a_i} \tilde{I}(i) + \bar{b_i}, \tag{3.12}$$

Les formules mathématiques pour calculer $\bar{a_i}$ et $\bar{b_i}$ sont données dans [HST13]. Deux paramètres permettent d'ajuster le comportement du filtre guidé. Il s'agit du paramètre ϵ et de la taille de la fenêtre ω_k. Il est possible de relier ces deux paramètres à σ_r et σ_s. Le paramètre σ_s permet de définir la taille de la fenêtre ω_k, tandis que le paramètre σ_r permet d'ajuster le paramètre ϵ. La figure 3.18 illustre les cartes de diffusion obtenues avec le filtre guidé.

FIGURE 3.18 – Cartes de diffusion issues du filtre guidé. Gauche : carte de diffusion de l'arrière plan I_{bg}. Droite : carte de diffusion du premier plan I_{fg}.

Il est intéressant de noter qu'une des applications du filtre guidé décrites dans [HST13] est le *matting*. Cette application est valable lorsque le masque binaire est déjà parfaitement aligné avec l'image coleur, ce qui la différencie de notre contexte, où nous cherchons justement à aligner le masque binaire à l'image couleur.

3.5.4 Filtre *domain transform*

Ce filtre est décrit dans [GO11]. La caractéristique de ce filtre est de conserver les propriétés du filtre bilatéral tout en diminuant la complexité du calcul.

Pour une image 2D, le filtrage est effectué en plusieurs itérations. Chaque itération contient un filtrage 1D horizontal et un filtrage 1D vertical. La valeur du paramètre de filtrage est réduite de moitié à chaque itération afin de réduire les artefacts. En pratique 3 itérations sont suffisantes.

La première étape de ce filtre consiste à transférer le signal à filtrer de son domaine original Ω dans un domaine particulier Ω_w. Dans ce domaine, l'écart entre deux points représente à la fois la distance spatiale qui existe entre ces deux points dans le domaine original et également la distance en terme d'intensité. Dans le domaine particulier Ω_w le signal est spatialement filtré avec un noyau gaussien. Puisque la distance spatiale entre chaque point encode désormais également l'écart en intensité, ce filtrage spatial permet de conserver les variations brutes dans le domaine initial. La dernière étape consiste à transférer le signal filtré dans le domaine initial Ω. Les variations importantes du signal sont conservées tandis que les variations faibles sont lissées. Le principe de ce filtrage est illustré sur la figure 3.19 [1] .

1. Images tirées de la vidéo de présentation liée aux travaux de [GO11]. `http://www.inf.ufrgs.br/~eslgastal/DomainTransform/`

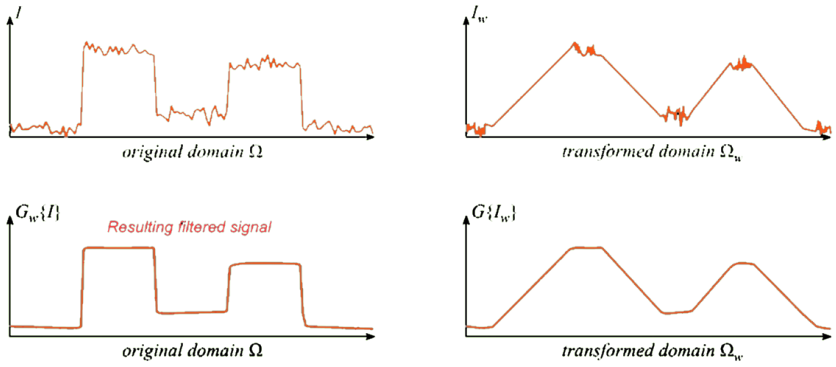


FIGURE 3.19 – Principe du filtre *domain transform* sur un signal en 1 dimension. Le signal est transformé dans un domaine Ω_w et filtré spatialement dans ce domaine. Le signal filtré est ensuite reporté dans le domaine original. Cette opération filtre le signal en conservant les variations importantes. Elle est efficiente en terme de complexité et coût mémoire.

Lors d'une itération du filtrage d'une image 2D, un filtre 1D a lieu sur les lignes horizontales. Afin de mieux comprendre la transformation de domaine qui est opérée lors de ce filtrage, considérons une ligne de pixels horizontale dans une image en niveaux de gris. Lorsque nous traçons la courbe de l'intensité des pixels en fonction de leur position le long de la ligne il s'agit d'un signal en 2 dimensions. Le filtre *domain transform* va transformer ce signal dans un domaine en 1 dimension. La transformée est une isométrie qui va prendre en compte les distances (verticale et horizontale) qui existent entre deux points successifs dans le domaine original afin de définir l'écart entre ces deux points dans le domaine transformé. La distance entre deux points successifs dans le domaine transformé est égale à la somme des distances verticale et horizontale dans le domaine initial. On peut visuellement représenter cette transformation comme le déploiement de la courbe (représentant le signal en 2 dimensions) afin de la mettre à plat (1 dimension). Dans ce nouveau domaine le signal est filtré par une convolution avec un filtre 1D dont la réponse diminue avec la distance tel qu'un filtre gaussien. Dans cette méthode le support du filtre est constant, c'est le signal qui s'adapte au filtre afin de contrôler la force du filtrage. Dans notre cas cette adaptation est réalisée avec les paramètres σ_r et σ_s qui vont contrôler la contraction ou d'étirement du signal original avant le passage dans le domaine transformé.

Pour obtenir un filtrage guidé avec ce type de filtre, on calcule le changement de domaine en utilisant les données de l'image guide $\tilde{I}$, puis le filtrage dans le domaine transformé s'effectue sur les données de l'image I.

Un exemple de filtrage de cartes d'influence est illustré sur la figure 3.20.

FIGURE 3.20 – Cartes de diffusion anisotropique issues du filtre guidé. Gauche : carte de diffusion anisotropique de l'arrière plan I_{bg}. Droite : carte de diffusion anisotropique du premier plan I_{fg}.

3.5.5 Filtre *Adaptive manifolds*

Ce filtre qui préserve également les contours est décrit dans [GO12]. L'originalité de ce filtre consiste à utiliser un nombre N de *manifolds* afin d'interpoler un noyau de filtrage proche de celui obtenu avec un filtre bilatéral classique. On peut concevoir un *manifold* comme une image caractéristique de l'image originale. Chaque image caractéristique est filtrée avec un filtre gaussien. Le principe du filtre *adaptive manifolds* est de considérer que la valeur en chaque pixel de l'image filtrée est une combinaison linéaire des pixels dans les images caractéristiques.

Pour du filtrage guidé les manifolds sont construits à partir de l'image guide $\tilde{I}$. Généralement 3 *manifolds* sont suffisants pour filtrer une image I. Le premier *manifold* M_1 correspond à l'image guide $\tilde{I}$ filtrée avec un filtrage moyen. Le second *manifold* M_2 est construit par un filtrage moyen des pixels de l'image guide $\tilde{I}$ en ne considérant que ceux dont la valeur est supérieure à celle du pixel équivalent dans le premier *manifold* M_1. Le troisième *manifold* M_3 est construit de manière équivalente à M_2 mais en considérant dans ce cas seulement les pixels de l'image guide $\tilde{I}$ dont la valeur est inférieure.

La comparaison d'un pixel de l'image guide avec chacun des pixels associés dans les *manifolds* permet de définir en chaque pixel un nombre N de poids. Ces poids sons calculés par le biais d'une fonction gaussienne (contrôlée par le paramètre σ_r). l'image à filtrer I est multipliée par chacun des poids obtenus afin d'obtenir N images. Celles-ci sont lissées avec un filtre gaussien (contrôlé par le paramètre σ_s). L'image filtrée J est obtenue par une combinaison linéaire des N images filtrée pondérées par les poids calculés précédemment.

La figure 3.21 permet d'illustrer le principe de ce filtre [2].

Nous utilisons ce procédé afin d'appliquer une diffusion anisotropique aux masques $M_{brut,fg}$ et $M_{brut,bg}$. Le paramètre σ_r permet d'ajuster le calcul des poids tandis que σ_s définit la taille de la fenêtre du filtre moyen. Les cartes de diffusion obtenues avec ce filtre sont visibles sur la

2. Images tirées de la vidéo de présentation. `http://inf.ufrgs.br/~eslgastal/AdaptiveManifolds/` liée aux travaux [GO12]

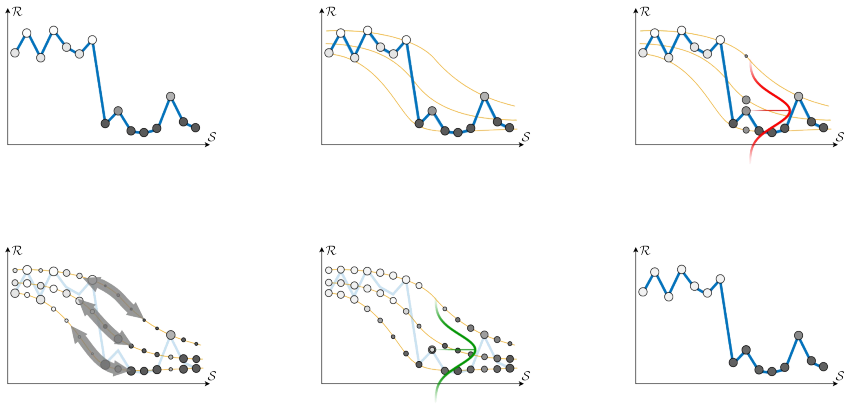


FIGURE 3.21 – *Adaptive Manifolds Filter*. La première étape consiste à construire des *manifolds* adaptés au signal à filtrer. On calcule ensuite le poids de chaque point du signal sur ces manifolds. Les poids sont lissés sur chacun des manifolds. Enfin on calcule la valeur du nouveau point du signal en faisant une combinaison linéaire de la valeur des manifolds pondérée par le poids du manifold.

figure 3.22.

FIGURE 3.22 – Cartes de diffusion issues du filtre guidé. Gauche : carte de diffusion de l'arrière plan I_{bg}. Droite : carte de diffusion du premier plan I_{fg}.

La figure 3.23 permet de visualiser les cartes de diffusion ainsi que la silhouette obtenue pour chacun des filtres énoncés.

3.5.6 Algorithme *Graph Cut*

L'algorithme *Graph Cut* nous permet d'obtenir une référence en terme de segmentation, c'est pour cette raison que nous décidons de l'utiliser.

L'algorithme *Graph Cut* [BV06], très utilisé dans le domaine de la vision par ordinateur, permet de produire une segmentation binaire. Une application utilisant le *Graph Cut* pour

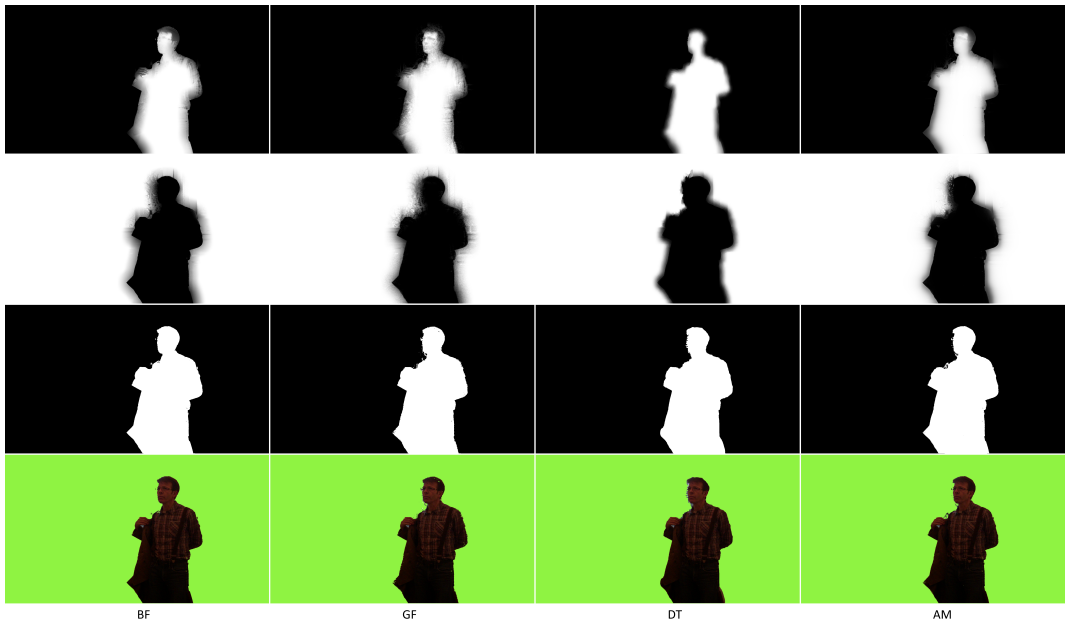


FIGURE 3.23 – Comparaison de l'utilisation des différents filtres pour la méthode d'affinage de la silhouette. Les deux premières lignes illustrent les cartes de diffusion du premier plan et de l'arrière plan. La troisième ligne illustre la silhouette obtenue. Enfin, la dernière ligne affiche le *compositing* avec l'image couleur. Dans l'ordre des colonnes : Filtre bilatéral joint (BF), Filtre guidé (GF), Filtre *domain transform* (DT), Filtre *adaptive manifold* (AM).

séparer l'arrière plan du premier plan nécessite des indications de la part de l'utilisateur. Généralement, il s'agit de *scribbles* pour indiquer des zones précises. Un *scribble* appliqué à un pixel permet d'étiqueter sa classe d'appartenance. De cette façon, l'utilisateur indique à l'algorithme un sous-ensemble de pixels appartenant à la classe du premier plan ainsi qu'un sous-ensemble de pixels appartenant à la classe de l'arrière plan. Les pixels qui n'ont pas reçu d'indication sont étiquetés en tant que pixel appartenant à la classe indéterminée. Pour chacun des pixels de cette dernière classe, l'algorithme déterminera s'il appartient à la classe de l'arrière plan ou celle du premier plan.

Dans notre cas, on utilise les masques $M_{brut,fg}$ et $M_{brut,bg}$ afin de générer automatiquement l'étiquetage de chaque pixel. Un pixel de valeur 1 dans le masque $M_{brut,fg}$ indique que le pixel est étiqueté à la classe du premier plan, tandis qu'un pixel de valeur 1 dans $M_{brut,bg}$ indique que le pixel est étiqueté à la classe de l'arrière plan. En combinant ces cartes brutes, on peut définir une carte comportant 3 régions distinctes : l'arrière plan, le premier plan et une bande d'incertitude. Cette carte des zones est aussi appelée *trimap*. L'algorithme *Graph Cut* détermine ensuite pour chaque pixel de la zone indéterminée s'il appartient à l'avant ou l'arrière plan. La figure 3.24 illustre la *trimap* originale ainsi que la *trimap* affinée après l'algorithme *Graph Cut*.

Cet algorithme construit un graphe où chaque pixel de l'image représente un nœud (figure 3.25). Si deux pixels sont voisins dans l'image, alors les nœuds les représentant sont reliés. Ce graphe est également composé de deux nœuds terminaux représentant respectivement la

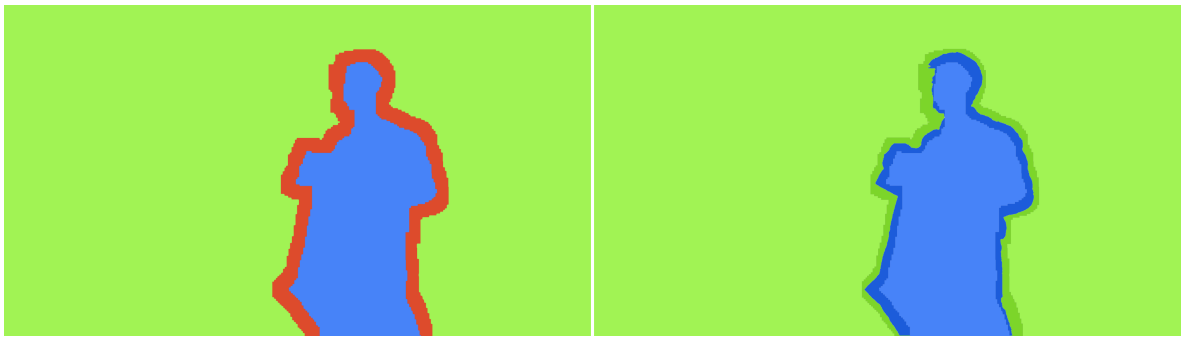
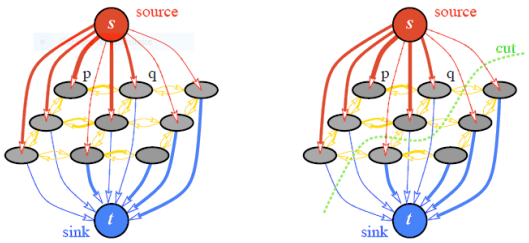


FIGURE 3.24 – *Graph Cut trimap*. Gauche : *trimap* originale construite à partir de $M_{brut,fg}$ et $M_{brut,bg}$. En vert l'arrière plan, en bleu le premier plan et en rouge la zone indéterminée. Droite : l'algorithme *Graph Cut* décide pour chaque pixel de la zone indéterminée à quelle classe il appartient. Les pixels appartenant au premier plan sont en bleu foncé, ceux de l'arrière plan en vert foncé.

classe premier plan et la classe arrière plan. Chacun des nœuds représentant les pixels est relié aux deux nœuds terminaux. En fonction des indications de l'utilisateur, un poids est affecté sur le lien reliant un nœud pixel à un nœud terminal. Ce poids est fort si la probabilité de ce pixel d'appartenir à la classe représentée par le nœud terminal est forte et inversement. Un poids est également affecté au lien reliant deux nœuds pixels. La valeur de ce poids varie en fonction de la similarité entre la valeur de ces deux pixels. Deux pixels dont la valeur est similaire produit un poids fort, tandis que deux pixels dont la valeur est très différente produit un poids faible. L'algorithme du *Graph Cut* s'attache à rechercher dans le graphe la coupe qui minimise la somme des poids des liens coupés. Après cette coupe, chaque nœud pixel est lié à seulement un nœud terminal. Le pixel prend alors le label du nœud terminal auquel il est relié.



(a) A graph $\mathcal{G}$ (b) A cut on $\mathcal{G}$

FIGURE 3.25 – Illsutration de la construction et de la coupe du graphe de l'algorithme *Graph Cut*. Image tirée de [BV06].

Le résultat du découpage avec utilisation du masque fabriqué par le *Graph Cut* est visible sur la figure 3.26. Ce résultat montre que la correspondance des contours est de meilleure qualité avec le *Graph Cut*. L'algorithme du *Graph Cut* a l'avantage d'être global dans le sens où il optimise une fonction d'énergie sur l'ensemble de l'image. En ce sens, il n'est pas gêné par les minimas locaux.

FIGURE 3.26 – Exemple de résultat obtenu. Gauche : masque binaire brut. Droite : mélange du masque affiné avec l'image film. Le masque binaire obtenu avec l'algorithme *Graph Cut* épouse mieux la forme de la silhouette dans l'image film.

3.6 Résultats

Nous appliquons la méthode d'affinage du masque de la silhouette brute sur les séquences évoquées dans la section 3.4. La vérité terrain est fournie par *Sensarea*. Une dizaine d'images successives sont traitées dans chacune des séquences. Afin de comparer les masques obtenus avec ceux de la vérité terrain, plusieurs mesures sont réalisées : l'erreur quadratique moyenne (RMSE), la *F-measure* et l'erreur de *Yasnoff* [YMB77, MBC11].

3.6.1 Erreur quadratique moyenne RMSE

La première mesure à laquelle nous nous intéressons correspond à la racine carrée de l'erreur quadratique moyenne entre l'image binaire de la vérité terrain et l'image binaire obtenue par l'une des méthodes proposées. Cette mesure est aussi appelée erreur RMSE (*root-mean-square error*). La mesure d'erreur est effectuée pixel à pixel. La formule est donnée dans l'équation 3.13.

$$RMSE = \sqrt{\frac{\sum (I(p) - I_{GT}(p))^2}{N}} \tag{3.13}$$

La figure 3.27 permet d'illustrer cette erreur pour une image pour chacune des méthodes. Il s'agit de la valeur absolue de la différence entre l'image du masque de la vérité terrain et l'image du masque proposé par une des méthodes.

FIGURE 3.27 – Visualisation de l'erreur RMSE pour une image du clip 7 ($\sigma_r = 20$ et $\sigma_s = 0.1$). Dans l'ordre : segmentation grossière issue de l'algorithme proposée par la *Kinect* (Raw), *Graph Cut* (GC), filtre bilatéral (BF), filtre guidé (GF), *domain transform* (DT), *adaptive manifolds* (AM)

3.6.2 Erreur *F-measure*

L'erreur RMSE est une information qui permet de comptabiliser les pixels qui différent entre deux images. Cependant, pour la comparaison de masques de segmentation il peut également être intéressant d'affiner cette information. Parmi les pixels erronés on trouve les faux négatifs (FN), c'est-à-dire les pixels considérés à tort comme appartenant à l'arrière plan, et les faux positifs (FP), c'est-à-dire les pixels considérés à tort comme appartenant au premier plan. La *F-measure* tient compte de cette différenciation à travers deux grandeurs : le rappel et la précision.

FIGURE 3.28 – Visualisation de l'erreur F-measure pour une image du clip 7 ($\sigma_r = 20$ et $\sigma_s = 0.1$). En blanc les vrais positifs VP, en noir les vrais négatifs VN, en jaune les faux positifs FP, en rouge les faux négatifs FN. Dans l'ordre : segmentation grossière issue de l'algorithme proposée par la *Kinect* (Raw), *Graph Cut* (GC), filtre bilatéral (BF), filtre guidé (GF), *domain transform* (DT), *adaptive manifolds* (AM)

Soit A les pixels représentant la classe silhouette dans l'image vérité terrain. Soit B les pixels représentant la classe silhouette dans l'image dont on cherche à évaluer l'erreur. Le rappel, noté R, correspond à la grandeur définie par le nombre de pixels correctement détectés comme appartenant à la classe silhouette, c'est-à-dire $A \cap B$, divisé par le nombre de pixels appartenant effectivement à la silhouette, c'est-à-dire A. Il s'agit également du rapport entre les vrais positifs VP et la somme des vrais positifs VP et des faux négatifs FN.

$$R = \frac{A \cap B}{A} = \frac{VP}{VP + FN} \tag{3.14}$$

La précision P correspond au ratio entre le nombre de pixels correctement détectés comme appartenant à la classe silhouette, c'est-à-dire $A \cap B$, et le nombre total de pixels total considérés comme étant la silhouette, c'est-à-dire B. Il s'agit également du rapport entre les vrais positifs VP et la somme des vrais positifs VP et des faux positifs FP.

$$P = \frac{A \cap B}{B} = \frac{VP}{VP + FP} \tag{3.15}$$

La *F-measure* prend en compte ces deux paramètres R et P afin d'évaluer la qualité de la segmentation :

$$F_{measure} = 2 \times \frac{P \times R}{P + R} \tag{3.16}$$

Plus le score se rapproche de la valeur 1, plus les masques binaires sont similaires. La figure 3.28 permet d'illustrer cette erreur pour une image tirée d'un des clips étudiés.

3.6.3 Erreur de *Yasnoff*

Contrairement aux autres mesures d'erreur qui font une comparaison pixel à pixel entre le masque calculé et la vérité terrain, la mesure de *Yasnoff* s'intéresse au voisinage. Cette mesure calcule la distance euclidienne $d(p)$ entre le pixel courant de la segmentation obtenue et le pixel de la même classe le plus proche dans la vérité terrain. L'erreur finale correspond à la RMSE de la carte des distances 3.17.

$$e_{Yasnoff} = \sqrt{\frac{\sum d(p)^2}{N}} \tag{3.17}$$

La figure 3.29 permet d'illustrer cette erreur pour une image tirée d'un des clips étudiés.

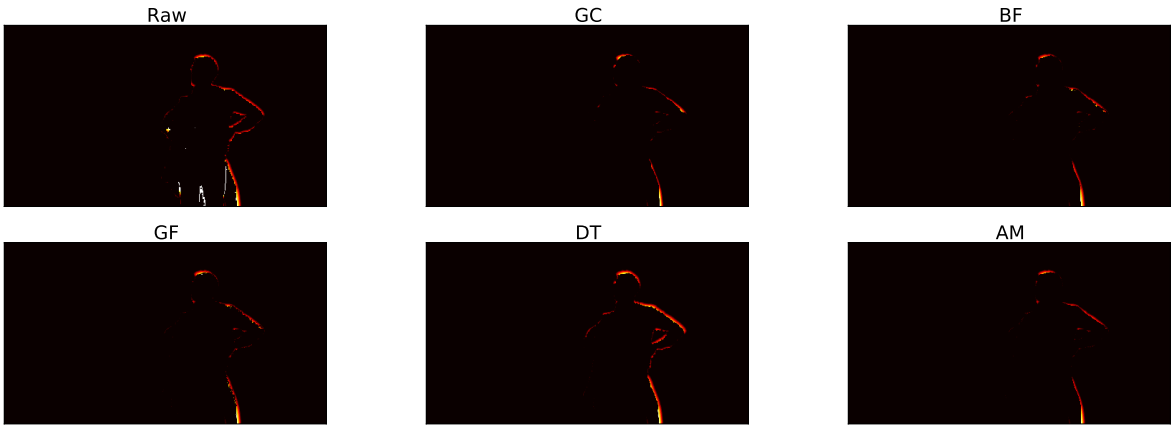


FIGURE 3.29 – Visualisation de l'erreur de *Yasnoff* pour une image du clip 7 ($\sigma_r = 20$ et $\sigma_s = 0.1$). La distance est codée par une échelle de couleur, en noir une distance nulle, en blanc la distance maximale. Dans l'ordre : segmentation grossière issue de l'algorithme proposée par la *Kinect* (Raw), *Graph Cut* (GC), filtre bilatéral (BF), filtre guidé (GF), *domain transform* (DT), *adaptive manifolds* (AM)

3.6.4 Choix des paramètres σ_r et σ_s

Afin d'évaluer objectivement chaque méthode, nous les appliquons à chaque image en faisant varier les paramètres σ_r et σ_s. Ces deux paramètres sont les seuls réglages possibles. Il est à noter que pour le graph cut, seul le paramètre σ_s est utile puisqu'il définit la largeur de la bande d'incertitude dans la *trimap*. Pour chaque méthode, nous déterminons le jeu de paramètres σ_r et σ_s qui produit l'erreur minimale pour chaque clip. Avec $\sigma_r \in [5, 10, 15, 20, 25, 30, 40]$ (en pixel) et $\sigma_s \in [0.1, 0.2, 0.3, 0.4, 0.5]$ (l'intensité est normalisée entre 0 et 1)). L'erreur pour chaque clip correspond à la moyenne des erreurs sur l'ensemble des images dont nous disposons de la vérité terrain.

Pour chaque clip traité par chaque méthode nous dressons une carte de l'erreur en fonction des paramètres. La figure 3.30 illustre l'erreur de *Yasnoff*.

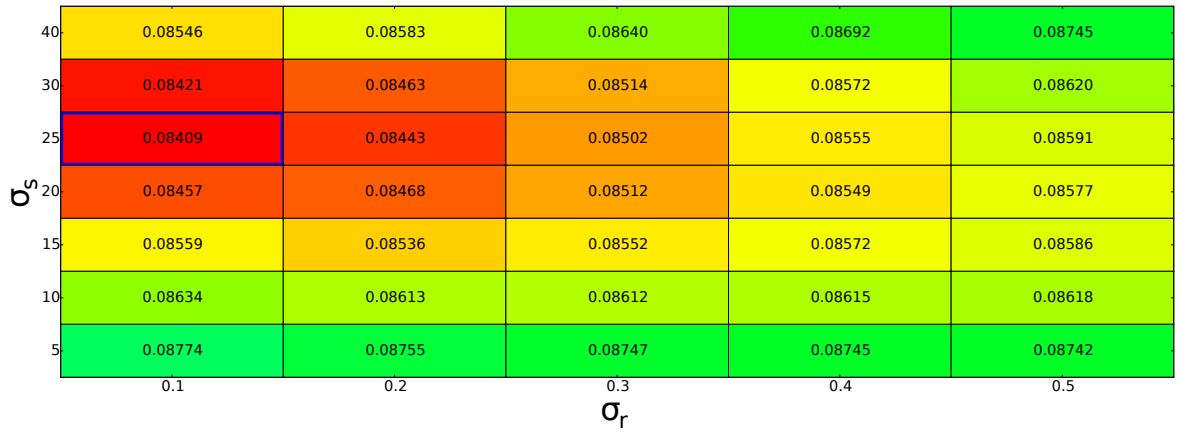


FIGURE 3.30 – Carte de l'erreur de *Yasnoff* moyenne avec la méthode basée sur le filtre bilatéral en fonction des paramètres σ_r et σ_s pour le clip 3. Les résultats sont présentés sous la forme d'une carte de chaleur. L'erreur minimale tend vert le rouge tandis que l'erreur maximale tend vers le vert. Ici l'erreur minimale est encadrée en bleue, elle est obtenue avec $\sigma_r = 0.1$ et $\sigma_s = 25$.

3.6.5 Analyse des résultats

Nous allons comparer les résultats de la segmentation pour chacune des méthodes utilisées. Les méthodes comparées sont celles basées sur la diffusion anisotropique (comprenant l'ensemble des filtres présentés précédemment) ainsi que la méthode utilisant le *Graph Cut*. Nous comparons également avec la segmentation brute (dénommée *Raw*). Les erreurs RMSE, *F-Measure* et *Yasnoff* sont calculées pour chacune de ces méthodes en comparant le masque obtenu avec la vérité terrain. Le tableau 3.3 établit l'erreur moyenne [3] optimale sur l'ensemble des clips pour chacune des méthodes mises en place.

3. Il est à noter que le calcul de la F-Measure moyenne n'est pas tout à fait juste. Il faut en effet d'abord considérer la moyenne du rappel et de la précision sur l'ensemble des clips pour enfin reconstituer la F-Measure finale.

Méthode	RMSE	F-Measure	Yasnoff
Segmentation brute **Raw**	0.1135	0.9519	0.1023
Graph Cut **GC**	**0.0787**	**0.976**	**0.0862**
Filtre bilatéral **BF**	**0.0871**	**0.9708**	0.0875
Filtre guidé **GF**	0.0897	0.9693	0.0886
Filtre *domain transform* **DT**	0.1005	0.9621	0.0924
Filtre *adaptive manifolds* **AM**	0.0913	0.9679	**0.0872**

Tableau 3.3 – Erreur moyenne pour l'ensemble des clips avec chaque méthode. La valeur de la meilleure erreur est minimale pour RMSE et *Yasnoff*, elle est maximale pour la *F-Measure*. En bleu le meilleur score, en rouge le second meilleur résultat.

Comme on pouvait s'y attendre la méthode qui utilise le *Graph Cut* obtient à chaque fois le meilleur score. Il s'agit en quelque sorte de la borne à atteindre pour une méthode locale temps réel. C'est pour cette raison que cette méthode, différentes des autres, a été choisie dans le comparatif. En fonction de l'erreur considérée, la méthode qui arrive en seconde position varie. La méthode avec le filtre bilatéral arrive en seconde position pour les erreurs RMSE et *F-Measure*. Si l'on considère l'erreur de *Yasnoff*, c'est la méthode du filtre *Adaptive Manifolds* qui arrive en seconde position, cependant la méthode du filtre bilatéral est troisième et vraiment proche.

Les figures 3.31, 3.32 et 3.33 permettent de visualiser le détail du score de chaque méthode pour chaque type d'erreur considéré. On remarque alors que 2 fois sur 7, si l'on regarde l'erreur de *Yasnoff* (figure 3.33), le filtre bilatéral (BF) est même meilleur que la méthode avec le *Graph Cut* (GC). Cela s'explique par le fait que mal initialisé, le GC ne peut pas s'adapter correctement alors que BF dispose d'une plus grande latitude. Cependant, sur l'ensemble des clips c'est la méthode du *Graph Cut* qui arrive en tête.

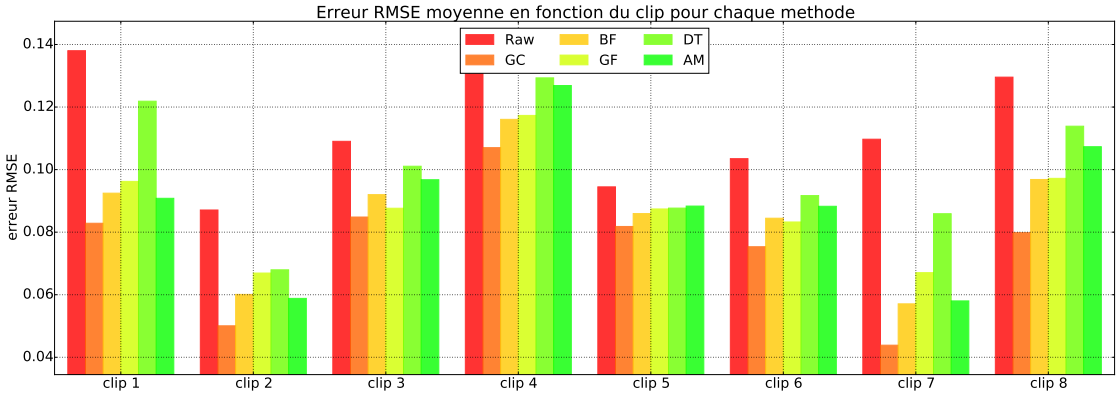


FIGURE 3.31 – RMSE moyenne en fonction du clip pour chaque méthode.

La méthode basée sur les images de diffusion fonctionne de façon probante lorsque l'initialisation de la silhouette brute est spatialement correcte et lorsque l'élément à segmenter se détache nettement de l'arrière plan. Si dans l'image couleur il n'y a pas de frontière nette entre le premier plan et l'arrière plan alors la limite se positionne au centre de la bande d'incertitude.

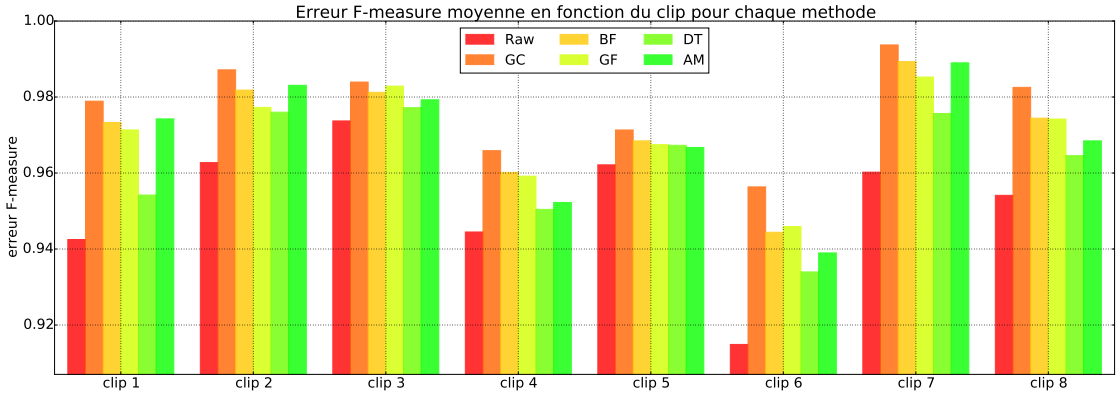


FIGURE 3.32 – F-Measure moyenne en fonction du clip pour chaque méthode

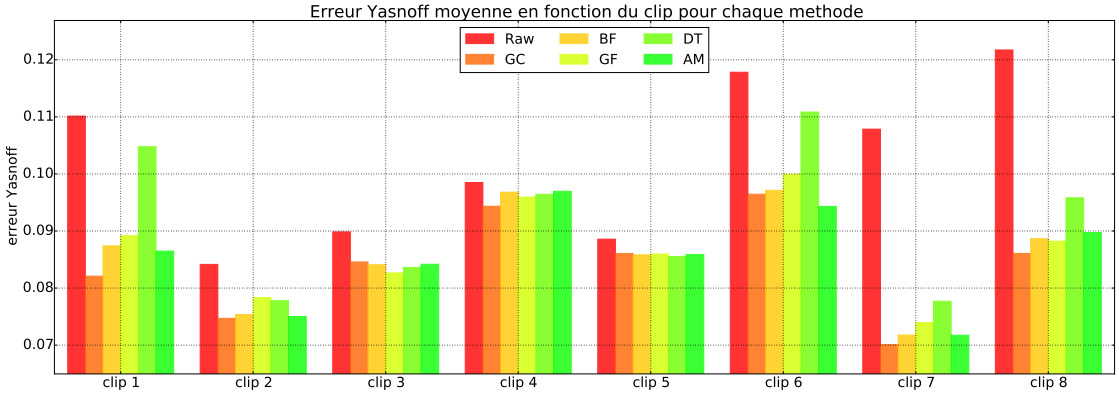


FIGURE 3.33 – Erreur Yasnoff moyenne en fonction du clip pour chaque méthode

De même, si l'initialisation du masque brut n'est pas correcte spatialement ou que la largeur de la bande d'incertitude n'est pas suffisamment large, alors la frontière entre le premier plan et l'arrière plan ne peut pas être déterminée de façon exacte.

Le clip 7 est un plan fixe où un acteur, habillé avec des vêtements dont la couleur est bien différente de l'arrière plan, enfile son manteau. Il présente une configuration performante pour la méthode basée sur les cartes de diffusion anisotropique (figure 3.34). Le masque brut est synchronisé de façon correcte avec l'image couleur, à la fois spatialement et temporellement. Visuellement, la silhouette de l'acteur se détache de l'arrière plan. La méthode proposée est alors capable d'affiner le masque binaire afin de correspondre aux contours de la silhouette dans l'image film. On peut voir sur les figures 3.31, 3.32 et 3.33 que l'écart d'erreur entre le masque brut et les méthodes proposées est important pour le clip 7.

Lorsque les mouvements sont rapides, le décalage spatial causé par une mauvaise synchronisation temporelle entre le capteur de profondeur et la caméra film est important. L'image brute fournie à l'algorithme pour calculer un masque affiné est une mauvaise initialisation. On observe ce phénomène sur le clip 4 où l'acteur exécute des mouvements rapides avec ses bras face à la caméra.

FIGURE 3.34 – Alignement du masque affiné avec l'image couleur pour une des images du clip 7.

Enfin, il est à noter qu'une segmentation correcte n'est pas possible lorsque la largeur de la bande d'incertitude est de l'ordre de grandeur de la largeur d'un élément à segmenter. Il faut trouver un compromis dans le réglage des paramètres pour obtenir une bande d'incertitude suffisamment large afin de corriger le décalage spatial ; et également étroite pour ne pas entrer en conflit avec la taille des éléments à segmenter.

FIGURE 3.35 – Alignement du masque affiné avec l'image couleur pour une des images du clip 4. Gauche : superposition du masque de segmentation brut avec l'image film. Droite : superposition du masque de segmentation obtenu avec la méthode du filtre bilatéral avec l'image film. La segmentation au niveau de la partie gauche du buste est affinée. Ce n'est pas le cas au niveau des cheveux. La segmentation affinée s'est alignée sur le contour entre le front et les cheveux et non l'arrière plan. Enfin, la segmentation entre les jambes n'est pas correcte, une mauvaise initialisation est à mettre en cause comme on peut le voir sur l'image de gauche.

3.7 Implémentation dans le moteur de jeux vidéo

La comparaison des différents filtres montre que le filtre bilatéral offre de bonnes performances. C'est ce filtre que nous implémentons dans le moteur de jeux. L'implémentation est réalisée avec des *shaders* dans le moteur de jeux *Unity*. Cette section décrit les détails de l'implémentation.

3.7.1 Utilisation de la grille bilatérale

Nous allons utiliser la grille bilatérale [CPD07] pour l'implémentation car cette représentation se prête bien à une utilisation avec le GPU et notamment avec des *shaders*. La complexité du filtre bilatéral est de $O(N \times (\sigma_s)^2)$, avec N le nombre de pixels dans l'image. En utilisant la représentation de la grille bilatérale le domaine spatial est quantifié avec S sous-images réparties sur 2 dimensions, on a $S = N/\sigma_s^2$. L'intensité est également quantifiée sur R niveaux, c'est la troisième dimension de la grille bilatérale. Ainsi la grille bilatérale est composée de $S \times R$ cellules. La taille de chaque cellule de cette grille est définie par $\sigma_s \times \sigma_s \times \sigma_r$. Avec l'utilisation de cette grille, la complexité du filtrage bilatéral devient $O(S + \frac{S \times R}{(\sigma_s)^2 \times (\sigma_r)^2})$. C'est une représentation adaptée pour les valeurs de σ_s et σ_r élevées puisque la complexité diminue lorsque celles-ci augmentent.

Le filtrage d'une image s'effectue en 3 étapes par le biais d'une grille bilatérale. La première étape consiste à placer l'image dans la grille (*splat*). Pour une image en niveaux de gris, il s'agit de repérer dans quelle cellule de la grille chaque pixel de l'image est associé. On affecte à chaque cellule la moyenne des valeurs des pixels qui lui sont associés. La seconde étape consiste à effectuer un filtrage gaussien en 3 dimensions sur l'ensemble de la grille (*blur*). Enfin, la dernière étape consiste à extraire une image à partir de la grille (*slice*). Pour cela, on lit la valeur de la cellule de la grille à laquelle chaque pixel est associé. Cette valeur définie la valeur du pixel dans l'image filtrée. La figure 3.36 illustre le principe de la grille bilatérale avec un signal en 1 dimension.

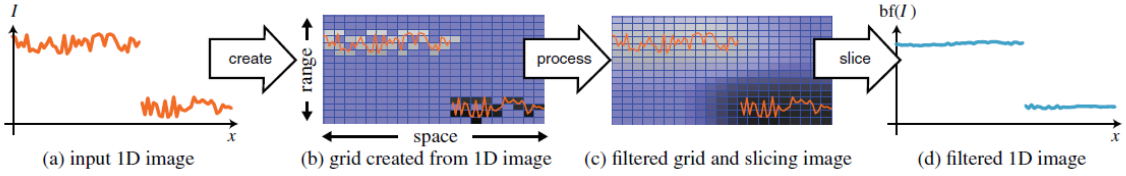


FIGURE 3.36 – Principe de la grille bilatérale appliquée sur un signal à 1 dimension. La grille bilatérale est créée autour de ce signal. Chaque valeur du signal en entrée est reliée à une case de la grille bilatérale (*splat*). La grille bilatérale est ensuite filtrée (*blur*). Enfin, un signal filtré est extrait de la grille (*slice*). Image tirée de [CPD07].

Dans notre cas, on utilise la luminance de l'image couleur afin de définir la cartographie de l'association entre un pixel de l'image et une cellule de la grille. Le contenu d'une cellule est défini par la valeur des pixels des masques $M_{brut,fg}$ et $M_{brut,bg}$. On effectue le filtrage gaussien

sur la grille $3D$, puis on construit les images J_{fg} et J_{bg} en affectant à chaque pixel la valeur présente dans la cellule qui lui est associée. La création de la grille, le filtrage gaussien et l'extraction de l'image filtrée sont réalisés au GPU par l'intermédiaire de shader au sein du moteur de jeux vidéo.

Nous filtrons deux images $M_{brut,fg}$ et $M_{brut,bg}$, cependant l'image guide est identique. Aussi, la construction de la grille a lieu une seule fois pour ces deux images. Cela permet de mutualiser les ressources. Chaque cellule possède deux canaux qui sont filtrées indépendamment. Le premier canal est utilisé pour les valeurs de $M_{brut,fg}$, tandis que le deuxième est utilisé par $M_{brut,bg}$. La valeur de chaque pixel de l'image de diffusion J_{fg} (respectivement J_{bg}) est obtenue par la lecture du premier canal (respectivement du deuxième canal) de la cellule associée à ce pixel d'après la cartographie établie par l'image couleur. Enfin, un *shader* permet de combiner J_{fg} et J_{bg} afin d'obtenir le masque de la segmentation affiné M_{affine}.

Avec une silhouette affinée il est possible de remplacer l'arrière plan de la scène par un fond virtuel sans avoir besoin de recourir à un fond vert. Dans *Unity*, le rendu d'une scène similaire à l'application de remplacement d'arrière plan (telle que présentée dans la figure 3.6) avec la méthode de segmentation affinée, prend 27 millisecondes avec un ordinateur portable équipé d'un processeur Intel i7 à 2.8 GHz, 16Go de RAM et une carte graphique Nvidia Quadro K4100M.

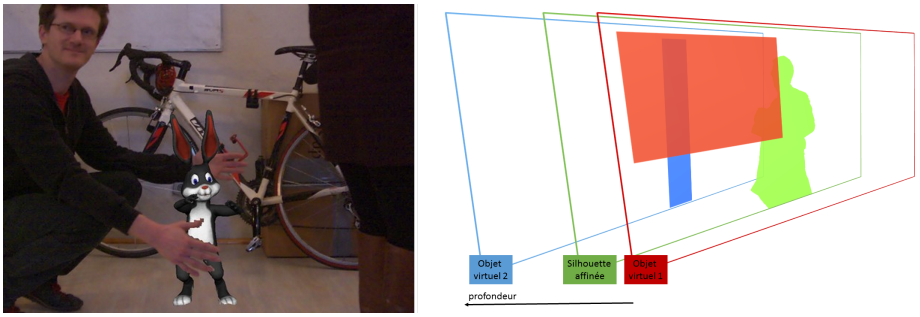


FIGURE 3.37 – Méthode de *compositing* basée sur la profondeur par rapport au principe du *compositing* avec une segmentation de silhouette affinée. Gauche : scénario possible avec la méthode de *compositing* basée sur la profondeur. Un acteur peut mettre un bras derrière et un bras devant un objet virtuel. En revanche la qualité du découpage est imprécise. Droite : principe du *compositing* avec une segmentation de la silhouette affinée. La silhouette est un calque possédant une profondeur unique pour tous les pixels qu'elle contient. Le découpage est plus précis, cependant la logique de calques restreint les possibilités d'occultations.

La finalité est le *compositing*. L'estimation de la profondeur moyenne de la silhouette en temps réel ouvre la voie à des scénarios tels qu'une scène avec des objets virtuels situés à différents niveaux de profondeur entre lesquels un acteur réel pourrait se déplacer. Lorsque le scénario implique une silhouette à intercaler entre des objets virtuels à différentes profondeur notre méthode est valide. Cependant, certains scénarios tels que présentés à la fin du chapitre 2 impliquent une segmentation au sein même de la silhouette. Notamment celui où un acteur place une main devant un objet virtuel et une autre main derrière. La figure 3.37 illustre les deux approches. En comparant pixel à pixel la profondeur, les possibilités d'occultations

entre virtuel et réel sont plus nombreuses mais les imprécisions liées aux limites du capteur sont importantes. En s'attachant à la segmentation d'un object en particulier, tel que la silhouette on améliore la qualité du *compositing* mais celui-ci est limité à un nombre restreint de possibilités.

Avec la segmentation affinée de la silhouette, le *compositing* obéit à une logique de calques. La segmentation de la silhouette est un calque qui fait face à la caméra intercalé avec d'autres calques. La profondeur est utilisée pour positionner les calques les uns par rapport aux autres. Nous ne possédons pas pour chaque pixel du masque de la silhouette affinée une mesure de profondeur comme l'illustre la figure 3.38.

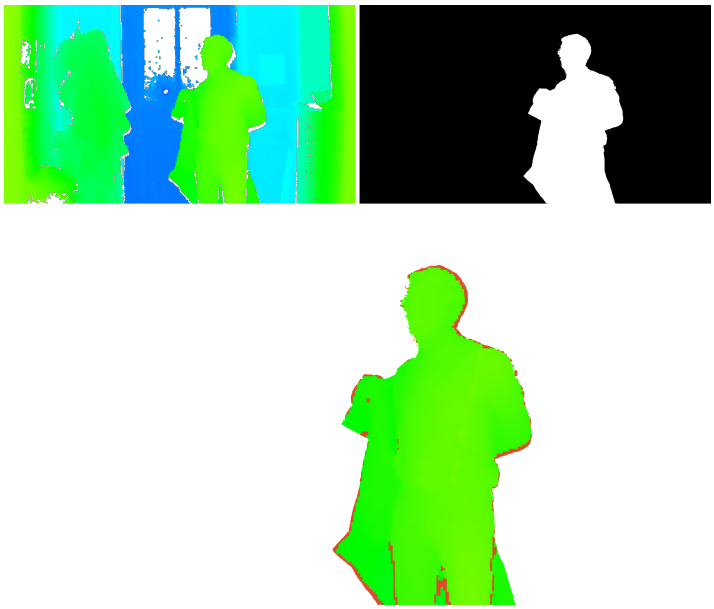


FIGURE 3.38 – Illustration des zones sans profondeur dans le masque de segmentation affinée de la silhouette. Haut gauche : profondeur brute de la scène réelle. Haut droit : segmentation affinée de la silhouette obtenue avec l'utilisation de l'image couleur. Bas : image de profondeur de la silhouette. Les zones en rouge indiquent les pixels appartenant au masque de la silhouette mais dont la profondeur n'est pas connue. On ne peut pas appliquer la méthode de *composting* basée sur la profondeur dans ces zones.

Plusieurs solutions sont possibles. Utiliser les techniques d'*inpainting* (voir annexe C) afin de combler les trous avec une estimation de la profondeur, puis appliquer la méthode de *compositing* basée sur la profondeur. Une autre approche, évoquée dans la section suivante, est une extension de notre méthode d'affinage de la segmentation de la silhouette. Celle-ci permet de gérer plus finement les occultations entre éléments virtuels et réels tout en améliorant la qualité du découpage. Le principe est d'identifier grossièrement dans l'image film les zones qui sont au premier plan du *compositing* en comparant les images de profondeur réelle et virtuelle. Cette segmentation grossière est ensuite affinée avec une méthode similaire à celle utilisée pour affiner la silhouette de l'acteur.

3.7.2 Extension de la méthode de segmentation

La méthode de *compositing* basée sur la profondeur décrite à la fin du chapitre 2 repose sur la création d'un masque binaire M_{rv} codant l'influence de l'image couleur réelle I_r et celle de l'image couleur virtuelle I_v dans le *compositing* final. Ce masque binaire est construit en comparant pixel à pixel les images de profondeur de la scène réelle D_r et de la scène virtuelle D_v (voir figure 2.36).

Dans la méthode d'affinage de la segmentation de la silhouette décrite dans ce chapitre, le point de départ est le masque de segmentation grossier fourni par le SDK de la *Kinect*.

Dans cette section l'étape de segmentation est faite conjointement avec le *compositing*. Ici nous allons utiliser le masque M_{rv} pour l'initialisation de la méthode d'affinage des contours. Deux masques binaires sont construits à partir du masque M_{rv}. Ceux-ci sont construits de façon similaire à ceux décrits dans la section 3.5. Le premier masque M_r est une érosion du masque M_{rv} tandis que le second masque M_v correspond au négatif de la dilatation du masque M_{rv}. Nous limitons les pixels valides de ces masques à la zone où l'objet virtuel est défini. Nous appelons α l'image qui définit là où sont les pixels valides (3.18). Un pixel $\alpha(i)$ est valide si un élément virtuel est visible dans l'image I_v au pixel i, c'est-à-dire si une distance de profondeur est définie au pixel i dans l'image D_v.

$$\alpha(i) = \begin{cases} 1, & \text{si } D_v(i) \text{ est définie} \\ 0, & \text{sinon} \end{cases} \tag{3.18}$$

Deux cartes de diffusion J_r et J_v sont créées en appliquant un filtre bilatéral sur M_r et M_v guidé par l'image couleur film.

$$J_v(i) = \alpha(i)\frac{1}{W(i)}\sum_{j\in\omega_i}\exp(-\frac{\|i-j\|_2^2}{\sigma_s^2})\exp(-\frac{\|I_{col}(i)-I_{col}(j)\|_2^2}{\sigma_r^2})M_v(j) \tag{3.19}$$

$$J_r(i) = \alpha(i)\frac{1}{W(i)}\sum_{j\in\omega_i}\exp(-\frac{\|i-j\|_2^2}{\sigma_s^2})\exp(-\frac{\|I_{col}(i)-I_{col}(j)\|_2^2}{\sigma_r^2})M_r(j) \tag{3.20}$$

Cette méthode est multimodale puisqu'elle s'appuie à la fois sur la comparaison des distance des cartes de profondeur réelle et virtuelle, et également sur l'image film. La figure 3.39 illustre l'utilisation de cette méthode de *compositing* multimodale. Nous avons choisi d'insérer un *donut* virtuel puisqu'il s'agit d'un objet virtuel dont la profondeur varie. L'objet est à la fois devant et derrière l'acteur. Les cartes de profondeur de la scène virtuelle et de la scène réelle sont comparées afin d'isoler la zone de l'image où le réel est devant le virtuel. La segmentation de cette zone est ensuite affinée en utilisant les contours présents dans l'image couleur.

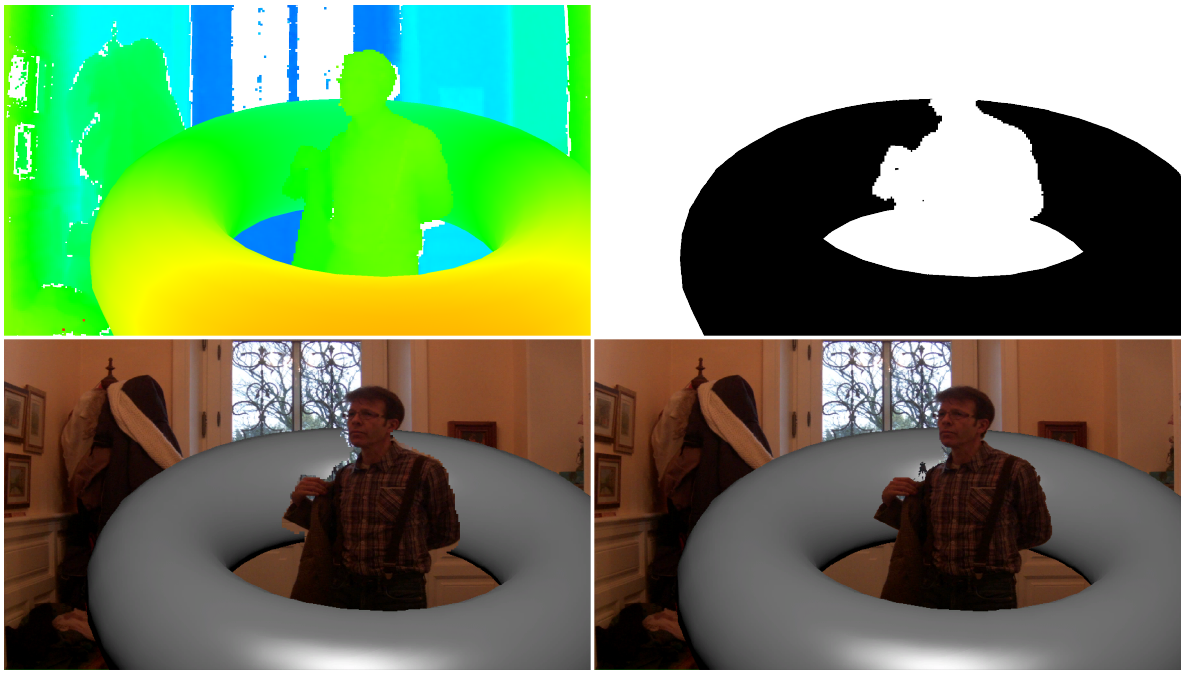


FIGURE 3.39 – Méthode multimodale de *compositing* basé sur la profondeur. Haut gauche : profondeur de la scène avec éléments virtuels et réels mélangés. Haut droite : masque M_{rv} obtenu en comparant la profondeur de la scène virtuelle et de la scène réelle. Bas gauche : méthode classique en comparant la profondeur réelle et virtuelle. Bas droite : méthode multimodale qui utilise l'image film afin d'affiner le découpage.

La méthode développée est étendue à une méthode de *compositing* multimodale. Le but de cette méthode est de produire un masque binaire codant l'influence de l'image couleur réelle et l'image couleur virtuelle dans le *compositing* final. Le *compositing* basé sur la profondeur décrit dans le chapitre précédent utilise seulement les images de profondeur virtuelle et réelle afin de produire ce masque. Cette méthode multimodale utilise en plus l'image couleur réelle afin d'affiner les contours du masque binaire.

Conclusion et perspectives

Nous avons proposé dans cette thèse des outils pour la résolution de problèmes de prévisualisation en temps réel. Par le biais d'un démonstrateur matériel, nous avons formalisé une solution intégrée dans un moteur de jeux vidéo. Les expérimentations menées montrent dans un premier temps des résultats encourageants pour le *compositing* en temps réel. Nous avons observé une amélioration des résultats suite à l'introduction de la méthode de segmentation conjointe. La principale force de ce travail réside dans la mise en place du démonstrateur qui nous a permis d'obtenir des algorithmes efficaces dans le domaine de la *previz on-set*.

Rappel du contexte

Nous nous intéressons à la prévisualisation appliquée au domaine de l'audiovisuel et plus particulièrement du cinéma. La prévisualisation est une opération préalable dont l'intérêt est de pouvoir anticiper et préparer le tournage. Dans ce travail de recherche, nous nous sommes focalisés sur la prévisualisation qui a lieu pendant le tournage, appelée aussi la *previz on-set*. Il s'agit de montrer au réalisateur une vue assemblée de son plan, c'est le *compositing*. Cette vue comprend à la fois les éléments réels, tels que les acteurs, qu'il est en train de filmer, et les éléments dits virtuels. Ces éléments virtuels correspondent à des éléments réalisés en image de synthèse. Il peut s'agir d'un arrière plan existant reconstitué numériquement car il est par exemple impossible de tourner dans ce lieu pour des questions pratiques ou économiques. Ces éléments virtuels peuvent également être des éléments qui n'existent pas dans la réalité tels que des créatures fantastiques ou des éléments de science-fiction. Tous ces éléments virtuels appartiennent à la scène virtuelle. Celle-ci est modélisée en amont du tournage. Par ailleurs, cette scène est observée selon un angle de vue qui est défini au moment du tournage, autrement dit l'affichage des éléments virtuels doit être calculé en temps réel. C'est justement cette condition précise qui pose problème. Tout d'abord, le temps réel nécessite que l'image mélangée, contenant les éléments réels et virtuels, soit produite en quelques millisecondes. Durant ce laps de temps, il nous faut être capable de déterminer quel est l'angle de vue de la caméra du plateau de tournage, afin d'être en mesure de répercuter cet angle dans la génération du rendu de la scène virtuelle. Cela nécessite aussi de hiérarchiser le plan de manière spatiale, c'est-à-dire de distinguer les éléments réels ou virtuels qui seront au premier plan, de ceux qui seront à l'arrière plan. Cette compréhension de la hiérarchie spatiale du plan est au cœur du sujet de ce travail de recherche.

Contributions

Dans le cadre de notre travail, nous avons mis en place un démonstrateur matériel. Celui-ci se compose d'une caméra de tournage instrumentée par un capteur de profondeur afin

d'obtenir une image de la distance des éléments réels de la scène par rapport à la caméra. Cette instrumentation pose des problèmes techniques importants. Pour obtenir l'information de profondeur, le système que nous avons mis en place utilise un dispositif à temps de vol, celui-ci est le module *Kinect 2*. Il est possible d'associer un tel capteur sur tout type de caméra. Si la polyvalence est un avantage, elle entraîne également un inconvénient non négligeable. Là où des fabricants de caméras peuvent intégrer un capteur de profondeur dans la conception de leur caméra et ainsi gérer de façon matérielle la synchronisation spatiale et temporelle, notre système doit compenser de façon logicielle ces limites. En effet, lorsque le capteur de profondeur est placé à côté du capteur couleur de la caméra de tournage on obtient un décalage spatial entre les deux référentiels des différents capteurs. Dans le but de rendre les données compatibles, il faut ajuster les données de l'un des capteurs pour les exprimer dans le référentiel de l'autre. Il s'agit d'un processus de reprojection développé dans le chapitre 2 de ce manuscrit. La première étape de ce processus est la calibration, elle consiste à calculer la transformation entre les capteurs. La seconde étape correspond à l'alignement. Il s'agit d'utiliser les données de la calibration pour faire correspondre les données de profondeur avec l'image couleur. Pour effectuer une synchronisation temporelle entre le flux de données du capteur de profondeur et le flux d'images de la caméra de tournage, nous utilisons les informations temporelles stockées dans chaque image. Chaque image issue de la caméra de tournage est appariée avec l'image de la caméra de profondeur dont le code temporel est le plus proche.

Nous avons aussi démontré qu'il est possible de réaliser la partie logicielle du démonstrateur au sein d'un moteur de jeux vidéo. Cet outil est adapté pour l'application de vision que nous avons réalisée. Il s'agit d'une innovation dans le domaine du cinéma puisque nous n'avons pas trouvé de publications traitant spécifiquement de la *previz on-set* via l'utilisation d'une telle plateforme. Nous utilisons les données de profondeur recalées dans le référentiel de la caméra de tournage afin de les comparer aux données issues de la scène virtuelle. Cette comparaison est effectuée au sein d'un moteur du jeux vidéo où la scène virtuelle est préparée. Le moteur de jeux vidéo permet précisément de produire un rendu temps réel d'une scène virtuelle. De plus, l'interface du moteur de jeux offre un cadre de travail malléable et peut être configurée pour développer notre application. Nous avons également démontré le potentiel du moteur de jeux pour des applications de type réalité augmentée sur les plateaux de tournage avec des interactions possibles du réel sur les éléments virtuels de la scène.

Des imprécisions subsistent dans le *compositing*. Celles-ci sont notamment liées au décalage entre l'image de profondeur recalée et l'image couleur, tel que démontré dans le chapitre 3. Nous avons développé une méthode logicielle qui permet d'atténuer ces imprécisions en utilisant conjointement les informations de l'image de profondeur et de l'image couleur. Nous avons pris le parti de nous intéresser spécifiquement à l'atténuation des imprécisions autour de la silhouette de l'acteur puisqu'il s'agit d'un objet fréquemment segmenté en post-production pour produire un *compositing* final. Il est possible d'obtenir une segmentation grossière de l'acteur à partir de l'image de profondeur. Cette segmentation est alignée spatialement dans le référentiel de l'image couleur. Il est alors possible d'établir une *trimap* comprenant la zone de la silhouette, la zone de l'arrière plan, et une zone intermédiaire sans information. En utilisant des filtres guidés qui préservent les contours, nous pouvons construire des images de diffusion issues de la zone silhouette et de la zone de l'arrière plan. Les cartes de diffusion

sont guidées par les contours présents dans l'images couleur. Celles-ci sont ensuite comparées pour calculer la probabilité que chaque pixel a d'appartenir à la silhouette. Cette opération nous permet d'établir une segmentation affinée de la silhouette dans l'image. Parmi les filtres expérimentés, le filtre bilatéral joint présente des résultats performants. Une implémentation de cette méthode combinée avec un filtre bilatéral joint, via une grille bilatéral, permet d'obtenir une segmentation en temps réel.

Le démonstrateur réalisé est *low-cost* et configurable au sein du moteur de jeux vidéo. De plus, il est possible d'enregistrer les flux issus de la caméra de tournage et du capteur de profondeur dans leur format de fichier respectif pendant le tournage. Ces flux peuvent être réédités en séquences d'images et être importés dans un logiciel de *compositing* tel que *Nuke* pour la post-production. C'est ce logiciel que nous avons utilisé pour effectuer le recalage spatial de l'image de profondeur avec l'image de la caméra film lors de la création de notre base de données. La création d'une base de données de plusieurs clips tournés avec notre démonstrateur constitue par ailleurs une autre contribution de ce travail. Une dizaine d'images a été annotée manuellement dans chacun des clips afin de segmenter la silhouette de l'acteur présent.

Limites et perspectives

Le capteur de profondeur constitue une limite de ce système. D'une part la technologie employée ne permet pas d'obtenir la profondeur pour des scènes au delà de quelques mètres. D'autre part, la résolution du capteur reste très inférieure à celle de la caméra de tournage. Enfin, le débit des images est irrégulier et il n'est pas possible de le synchroniser temporellement avec celui de la caméra de tournage.

Des écarts apparaissent localement entre l'image de profondeur recalée spatialement et l'image couleur issue de la caméra de tournage. Notre méthode d'affinage de la segmentation permet de réduire ces écarts. Pour s'initialiser, la méthode nécessite une segmentation grossière de l'objet à segmenter. L'avantage de cette méthode est qu'elle fonctionne en temps réel et qu'elle s'intègre dans le moteur de jeux vidéo. Un inconvénient demeure : la segmentation grossière qui sert à l'initialisation ne doit pas être trop éloignée de la réalité. En effet, l'objectif de cette méthode est d'affiner la segmentation grossière en s'appuyant sur les contours naturels présents dans l'image couleur. Si l'initialisation n'est pas adéquate, le risque d'avoir une segmentation alignée sur un mauvais contour augmente. De surcroît, si l'image couleur ne présente pas de contour naturel, alors la méthode n'aura comme effet que de dilater la segmentation grossière.

Un aspect qui n'a pas été abordé dans cette thèse mais qui est extrêmement important pour augmenter la qualité du *compositing* dès lors que la segmentation de la silhouette est précise est le *matting*. Des solutions existent mais celles-ci ne sont pas forcément temps-réel, de plus elles nécessitent une puissance de calcul importante. Ce domaine constitue une piste de recherche intéressante pour la poursuite de ce travail.

Le démonstrateur matériel que nous avons développé est opérationnel. En effet, la preuve de son concept a été démontrée dans cette thèse. Nous avons développé plusieurs variantes de notre système et les résultats expérimentaux obtenus ont montré que la qualité que fournit notre méthode n'est pas suffisante pour la généraliser en production. Néanmoins, s'il reste à ce stade peu propice à une utilisation en production, il ouvre la voie à de tels dispositifs. Dans la continuité de notre travail nous pourrions adopter une approche avec un système hybride mêlant un capteur de profondeur et un fond vert. En effet, nous pensons qu'il est pertinent de conserver le fond vert afin d'obtenir une segmentation parfaite de l'acteur. Le capteur de profondeur nous fournit une information concernant la position dans l'espace de l'acteur afin de réaliser le *compositing* avec des éléments virtuels.

Si aujourd'hui la *previz on-set* est un outil de conception dans le domaine du cinéma, le milieu de la télévision espère aller plus loin. En effet, les professionnels du *broadcast* sont intéressés par l'utilisation de tels systèmes pour la diffusion d'émissions en direct. La qualité du rendu et la précision du *compositing* en temps réel sont donc des éléments essentiels. Dans cette optique, l'utilisation de la *previz on-set* est potentiellement amenée à devenir un outil standard dans les années à venir.

Annexe A

Autres méthodes d'acquisition de la profondeur

A.1 Variation de la mise au point

Les dispositifs précédemment présentés utilisent plusieurs caméras, ou associe une caméra avec un projecteur. Si l'on connait tous les paramètres optiques de la lentille utilisée , il est possible de déterminer la profondeur d'une scène à partir d'une seule caméra. Il s'agit généralement d'une optique avec une faible profondeur de champ. L'estimation de la profondeur à partir de la mise au point se divise en deux familles de méthodes : le *depth from defocus* et le *depth from focus*.

La première méthode *depth from defocus* [EL93] consiste à modéliser la lentille de façon précise pour interpréter la génération du flou dans l'image en fonction de la distance des objets. La société *Cyclopus* en partenariat avec l'ONERA propose une technologie [TCLBI13] capable d'extraire une carte de profondeur à partir d'une image (voir figure A.1). La technologie mise en œuvre repose sur l'utilisation d'un capteur couleur et d'une optique qui ne corrige pas l'aberration chromatique longitudinale. Ainsi, en fonction de la longueur d'onde, la mise au point ne se situe pas exactement à la même distance du capteur. Ceci permet de lever des ambiguïtés sur la résolution du problème de *depth from defocus*.

La seconde méthode *depth from focus* [XS93, JY12] consiste à analyser un ensemble d'images capturées en utilisant une mise au point différenciée. Le plan de mise au point balaie toute une plage de distance par rapport au capteur. Dans chaque image, pour chaque pixel, on évalue la mise au point, ce qui revient à déterminer si le pixel est net ou non. En fonction de l'image, on obtient pour chaque pixel, une courbe représentant sa netteté. La profondeur de ce pixel est alors liée au maximum de la courbe précédemment obtenue.

Les deux méthodes prennent en compte une modélisation de caméra bien plus précise que le simple *pinhole camera model* (voir section 2.3). On considère l'objectif comme une lentille convergente dont on peut faire varier l'ouverture et la mise au point. L'ouverture correspond au paramètre lié au diamètre de l'iris laissant entrer la lumière, tandis que la mise au point correspond au paramètre lié à la position du plan image. La figure A.2 permet d'illustrer un modèle de lentille convergente.

La profondeur de champ dépend de 3 paramètres : l'ouverture du diaphragme liée au

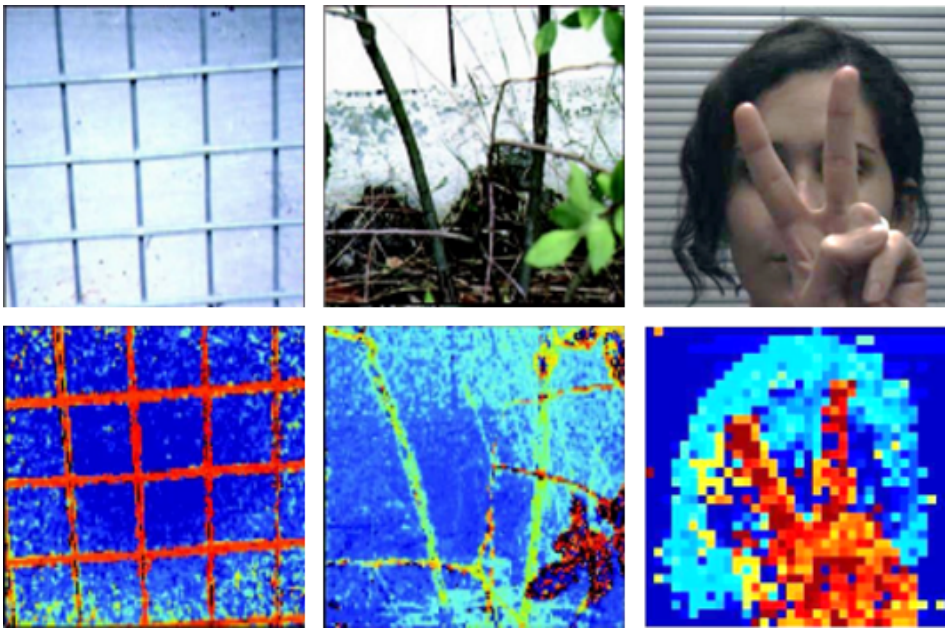


FIGURE A.1 – Exemple de cartes de profondeur obtenues avec la technologie développée par Cyclopus. Image tirée de *http ://www.cyclopus.net/*

paramètre *f-number* noté N, la focale f et la distance de mise au point s. Il est possible de calculer un intervalle de distance (borne minimum D_n et maximum D_f) dans lequel tout objet est observé avec une netteté jugée acceptable [1]. Cette mesure tient compte du cercle de confusion c lié au capteur utilisé. Les équations suivantes donnent le détail des calculs (d'après [Gre50]). Dans un premier temps, on détermine une grandeur appelée distance hyperfocale H qui est nécessaire pour la suite. Cette grandeur correspond à la distance maximale de mise au point. Autrement dit, avec une mise au point à la distance hyperfocale la borne supérieure D_f est infinie. La borne D_n correspond à la moitié de H.

$$H = \frac{f^2}{N \cdot c} + f \tag{A.1}$$

On peut ensuite calculer les bornes D_n et D_f.

$$D_n = \frac{s \cdot (H - f)}{H + s - 2f} \tag{A.2}$$

$$D_f = \frac{s \cdot (H - f)}{H - s} \tag{A.3}$$

1. On trouve des outils en ligne pour calculer automatiquement ces distances tel que `http://www.dofmaster.com/dofjs.html`.

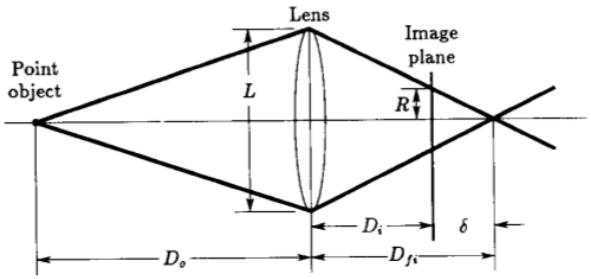


FIGURE A.2 – Modèle d'une lentille convergente utilisée dans la méthode *depth from focus*. La distance L est le paramètre lié à l'iris et défini la profondeur de champ. La distance D_i est le paramètre lié à la mise au point. Image tirée de [EL93].

On constate que la profondeur de champ décroît lorsque l'ouverture et la focale augmentent et lorsque la distance de mise au point diminue. Ainsi, pour obtenir une profondeur de champ faible correspondante à un objet éloigné, il faut choisir un objectif disposant d'une grande ouverture et d'une focale importante. L'inconvénient d'avoir une focale importante est que le champ de vision est faible, ce qui restreint la vision globale de la scène. De plus, le prix de tels objectifs est relativement élevé.

Enfin, pour obtenir une estimation de la profondeur en temps réel avec la méthode de *depth from focus* il faut capturer plusieurs images en modifiant les réglages de mise au point. Cela nécessite une cadence d'acquisition élevée et une instrumentation mécanique de la lentille pour obtenir un flux en temps réel. Cela rend le dispositif complexe à mettre en œuvre. De plus, il n'est pas possible de déterminer la profondeur sur les zones homogènes.

A.2 Caméra plénoptique

Un autre moyen d'estimer la profondeur à partir d'un seul capteur est d'utiliser un capteur plénoptique [NLB+05]. Ce capteur reprend l'architecture classique d'un capteur optique en glissant un réseau de micro-lentilles entre la dalle de photosites et l'optique (voir figure A.3).

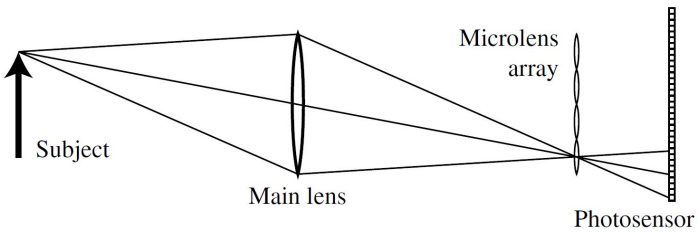


FIGURE A.3 – Principe du capteur plénoptique. Un réseau de micro-lentille est inséré entre l'optique et le capteur. Image tirée de [NLB+05].

Un capteur plénoptique ne capture pas seulement le point d'arrivée d'un rayon sur l'image (coordonnées i et j d'un pixel) mais il capture également la direction du rayon (angle vertical et horizontal). De cette façon, il n'y a pas besoin d'effectuer de mise au point lors de la prise de vue. Celle-ci peut être faite a posteriori. Il est possible de générer une carte de profondeur à partir de ce type de capteur. La figure A.4 présente une photo prise avec l'appareil Lytro (première génération). La carte de profondeur associée est également présentée.

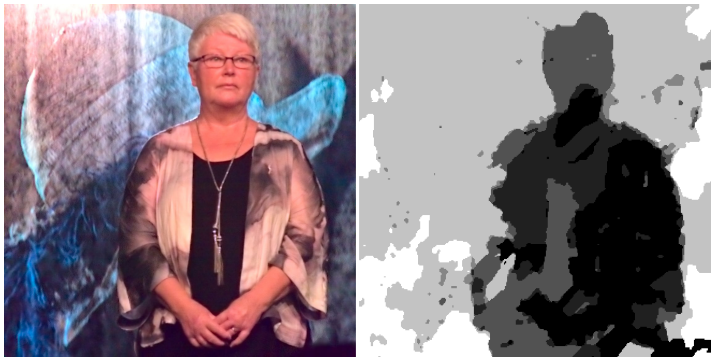


FIGURE A.4 – Exemple d'image couleur et d'une carte de profondeur capturées avec un appareil *Lytro* de première génération. Il s'agit d'une photo de Lena prise à Québec lors de la conférence ICIP en 2015.

La société *Lytro*, qui a commencé par commercialiser des appareils photos plénoptiques grand public, a conçu une caméra dédiée au cinéma, visible sur la figure A.5. La caméra est associée à un logiciel qui permet de refaire le point a posteriori, de modifier la position du point de vue (il s'agit d'une manipulation utile pour générer une vidéo stéréoscopique), ou encore de supprimer l'arrière plan afin d'isoler un acteur par exemple.

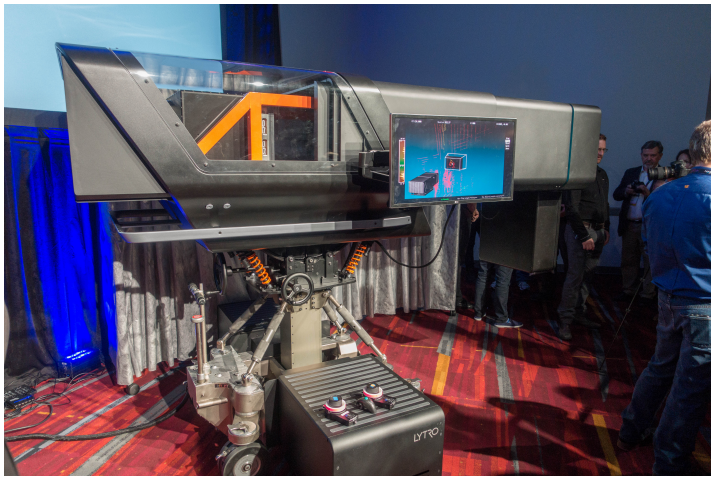


FIGURE A.5 – Caméra plénoptique *Lytro Cinema* présentée au salon NAB 2016 de Las Vegas par la société *Lytro* (source `http://www.dpreview.com/`).

Annexe B

Notes sur la calibration

B.1 Utilisation de l'information de profondeur pour déterminer la matrice de projection de la Kinect

Grâce aux mesures de distance, il est possible de déterminer la position du capteur de profondeur par rapport au damier. A partir de l'observation des coins du damier dans l'image infrarouge et d'une estimation de la matrice de projection il est possible de construire le nuage de points $3D$ correspondant aux coins du damier. Le barycentre de ce nuage de points nous donne la translation entre le capteur et le centre du damier. Ensuite, si on applique une décomposition en valeur singulière (SVD) à ce nuage de points centrés (on recentre chaque point en soustrayant le barycentre), il devient possible de déduire la matrice de rotation qui est appliquée au damier. Chaque colonne de la matrice de rotation est en fait un vecteur propre de la SVD effectuée précédemment. On utilise la transformation calculée pour projeter dans l'image les coins d'un damier idéal. On calcule l'erreur de reprojection e entre les points observés et les points projetés (voir équation B.1). Une méthode d'optimisation itérative permet de déterminer la matrice de projection $\boldsymbol{K_d}$ du capteur de profondeur qui minimise cette erreur.

$$e = \sqrt{\frac{\sum_{i=0}^{N-1} \|\boldsymbol{u_i} - \tilde{\boldsymbol{u}}_{\boldsymbol{i}}\|_2^2}{N}} \tag{B.1}$$

Minimiser e revient à minimiser le numérateur de l'équation définie ci-dessus. On appelle ce numérateur le résidu.

$$res = \sum_{i=0}^{N-1} \|\boldsymbol{u_i} - \tilde{\boldsymbol{u}}_{\boldsymbol{i}}\|_2^2 \tag{B.2}$$

Ou en notation matricielle :

$$res = \left\|\boldsymbol{U} - \tilde{\boldsymbol{U}}\right\|_2^2 \tag{B.3}$$

Avec $\boldsymbol{U}$ le vecteur colonne de dimension $2N \times 1$ contenant les coordonnées $2D$ des coins du damier observés dans toutes les images ; et $\tilde{\boldsymbol{U}}$ le vecteur colonne de même dimension que

$\boldsymbol{U}$ contenant les coordonnées $2D$ de la projection des coins du damier idéal, on peut réécrire l'équation sous la forme suivante :

$$res = \|\boldsymbol{U} - M\boldsymbol{k}\|_2^2 \tag{B.4}$$

avec $\boldsymbol{k} = \begin{pmatrix} f \\ c_x \\ c_y \end{pmatrix}$ le vecteur de dimension 3×1 contenant les paramètres de projection.

et $M = \begin{pmatrix} \frac{X_i}{Z_i} & 1 & 0 \\ \frac{Y_i}{Z_i} & 0 & 1 \\ & \vdots & \\ \frac{X_{N-1}}{Z_{N-1}} & 1 & 0 \\ \frac{Y_{N-1}}{Z_{N-1}} & 0 & 1 \end{pmatrix}$ la matrice de dimension $2N \times 3$ contenant les coordonnées $3D$ des coins du damier idéal.

On peut décomposer le vecteur de projection $\boldsymbol{k}$ de la façon suivante :

$\boldsymbol{k} = \boldsymbol{k_0} + \boldsymbol{\delta k}$

où $\boldsymbol{k_0}$ est le vecteur de projection initiale. Les données techniques du capteur fournies par le constructeur permettent d'obtenir une estimation de ce vecteur. $\boldsymbol{\delta k}$ représente l'incrément à ajouter au vecteur de projection initiale afin d'obtenir une meilleure estimation du vecteur de projection. C'est cet incrément que l'on cherche à estimer. Ce processus est répété de façon itérative jusqu'à trouver le vecteur de projection optimal. Le résidu correspond alors à :

$$res = \|\boldsymbol{U} - M\boldsymbol{k_0} - M\boldsymbol{\delta k}\|_2^2 \tag{B.5}$$

$$res = \left\|\boldsymbol{U} - \tilde{\boldsymbol{U}}_{\boldsymbol{0}} - M\boldsymbol{\delta k}\right\|_2^2 \tag{B.6}$$

Ou en généralisant à une itération i

$$res_i = \left\|\boldsymbol{U} - \tilde{\boldsymbol{U}}_{\boldsymbol{i-1}} - M\boldsymbol{\delta k_i}\right\|_2^2 \tag{B.7}$$

Cela équivaut à un problème des moindres carrés. On cherche la solution $\boldsymbol{x}$ à une équation du type $A\boldsymbol{x} = b$ tel que la norme $\|b - A\boldsymbol{x}\|_2$ soit minimisée. Dans notre cas on a $\boldsymbol{x} = \boldsymbol{\delta k_i}$, $A = M$ et $b = \boldsymbol{U} - \tilde{\boldsymbol{U}}_{\boldsymbol{i-1}}$.

Annexe C

Amélioration de la carte de profondeur

C.1 Obstruer les trous de la carte de profondeur

Nous présentons ici les méthodes existantes de l'état de l'art afin d'améliorer la qualité des cartes de profondeur. Le but est d'effectuer un pré-traitement sur ces images afin de les rendre exploitables par d'autres algorithmes, tel que le *compositing* temps réel dans notre cas.

La première étape du traitement des cartes de profondeur consiste à obstruer les éventuels trous présents dans l'image. L'absence de données sur la profondeur peut être liée à différents facteurs. Un élément de la scène constitué d'un matériau absorbant l'infrarouge ne renverra aucune onde vers le capteur, produisant alors un trou dans la carte de profondeur. Il existe également un décalage physique entre la source d'émission de l'onde infrarouge et le capteur de réception. Ce décalage introduit une légère ombre à la frontière de deux éléments de profondeurs différentes.

Afin de convenablement filtrer la carte de profondeur, il est important de faire disparaître ces trous. Ceci permet d'éviter les effets de bords qui pourraient survenir à la frontière des zones sans information.

La technique de l'*inpainting*, développée à l'origine pour les images d'intensité et introduite par [BSCB00], permet précisément de restaurer des zones d'une image sans information en prenant en compte l'information contenue dans les pixels voisins. On peut distinguer plusieurs approches détaillées dans la suite de cette annexe.

C.1.1 Méthodes basées matrices de convolution

L'article [OBMC01] décrit un algorithme d'*inpainting* rapide basé sur l'utilisation d'un filtre. La méthode est itérative, de sorte que le filtre est appliqué successivement jusqu'à la disparition du trou (voir figure C.1). Cette méthode fonctionne de manière satisfaisante pour les zones homogènes. En revanche, elle induit rapidement du flou dans les zones contenant des hautes fréquences. La solution évoquée dans cette article consiste à définir manuellement des limites de diffusion pour empêcher les effets de flous.

Une autre technique explicitée par [KW93] exploite la méthode de corrélation normalisée (filtre gaussien). Une implémentation GPU de cette méthode est évoquée dans [WBH11]. Ce

```
initialize Ω;
for (iter =0; iter < num_iteration; iter++)
  convolve masked regions with kernel;
```

a	b	a
b	0	b
a	b	a

c	c	c
c	0	c
c	c	c

FIGURE C.1 – **(haut)** Pseudo-code pour l'algorithme d'*inpainting* rapide. **(bas)** Deux exemples de noyaux de diffusion utilisés par l'algorithme (a = 0.073235, b = 0.176765, c = 0.125). Figure tirée de [OBMC01].

procédé est également itératif jusqu'à ce que tous les trous soient obstrués. Les équations suivantes donnent un aperçu de la méthode.

$$q_i = \frac{\sum_{j \in \omega_i} p_j \, a(i,j) \, m(j)}{\sum_{j \in \omega_i} a(i,j) \, m(j)} \tag{C.1}$$

$$a(i,j) = \exp\left(-\frac{\|i-j\|_2^2}{\sigma_s^2}\right) \tag{C.2}$$

$$m(i) = \begin{cases} 1, & \text{if } p_i \text{ is valid,} \\ 0, & \text{otherwise} \end{cases} \tag{C.3}$$

$$\tilde{q_i} = p_i \, m(i) + q_i \, (1 - m(i)) \tag{C.4}$$

Enfin, l'article [SBX14] utilise un filtre médian dans un *framework* pyramidal. La figure C.2 montre le résultat de cet algorithme sur une image de profondeur. Un filtre médian est appliqué à chaque étage de la pyramide de l'image de profondeur. Ensuite, l'image filtrée est reconstruite de façon itérative en mélangeant l'étage n et l'étage $n-1$ de la pyramide jusqu'à arriver à la résolution initiale de l'image à traiter.

C.1.2 Méthodes basées sur l'optimisation

L'approche globale de l'optimisation consiste à propager la profondeur dans les trous de l'image avec comme critère principal la continuité de la dérivée seconde de la profondeur. C'est-à-dire que la variation de la courbure est minimisée. Il est en effet démontré dans [WTRF09] que ce critère affiche de meilleurs résultats qu'un critère du premier ordre pour la restauration d'une carte de profondeur. Il s'agit d'un a priori usuel pour les formes naturelles. Ce critère est utilisé dans les travaux suivants [JKJ+13, THMR13, HKH+13]. Le principe est décrit dans l'équation C.5 avec $\tilde{\mathbf{Z}}$ la carte de profondeur optimisée, h un opérateur laplacien $[1, -2, 1]$, $*$ l'opérateur de convolution, et enfin $\|.\|_F^2$ la norme de Frobenius au carré. On contraint également la solution telle que $\tilde{\mathbf{Z}}_{\mathbf{xy}} = \mathbf{Z}_{\mathbf{xy}}$, si le pixel initial $\mathbf{Z}_{\mathbf{xy}}$ est valide.

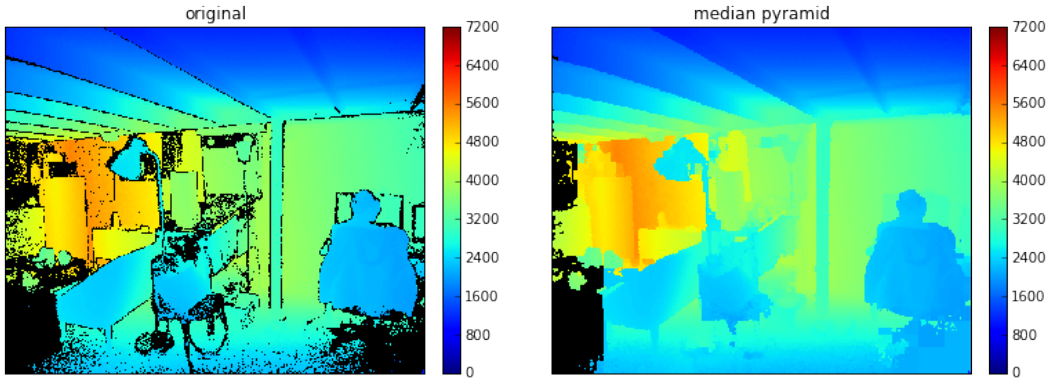


FIGURE C.2 – **(a.)** Image de profondeur originale. **(b.)** Image de profondeur avec une grande partie des trous obstrués grâce à une itération d'un filtre médian dans un *framework* pyramidal.

$$\tilde{\mathbf{Z}} = \underset{\tilde{\mathbf{Z}}}{\operatorname{argmin}} \left\| h * \tilde{\mathbf{Z}} \right\|_F^2 + \left\| h^{\mathsf{T}} * \tilde{\mathbf{Z}} \right\|_F^2 \tag{C.5}$$

Soit une image de profondeur de $h \times w = n$ valeurs. On considère le vecteur $\mathbf{z}$ qui contient en colonne toutes les valeurs de profondeur mises les unes à la suite des autres. On reformule l'équation précédente C.5 sous la forme de l'équation C.6.

$$\tilde{\mathbf{z}} = \underset{\tilde{\mathbf{z}}}{\operatorname{argmin}} \left\| (\mathsf{I} - \mathsf{W})\tilde{\mathbf{z}} - \mathbf{z} \right\|_2^2 \tag{C.6}$$

Avec I la matrice identité et W la matrice de dimension $n \times n$ contenant les poids pour le filtre laplacien (-2 sur la diagonale et 1 à l'intersection d'une ligne et d'une colonne de deux pixels voisins). Les lignes contenant un pixel valide sont mises à 0 puisque $\mathbf{z}$ sert de contrainte à l'optimisation.

Afin d'améliorer la solution , il est possible de rajouter des termes dans l'optimisation. Par exemple dans [MFL+12] l'auteur fait la différence entre les régions homogènes et les régions sur les arrêtes.

Il est intéressant de noter que ces méthodes se rapprochent pertinemment du problème posé dans [LLW04] où l'objectif est de diffuser de la couleur dans une image en niveaux de gris.

C.1.3 *Inpainting* basé sur les lignes isophotes

La méthode décrite dans [BSCB00] prend en considération les lignes isophotes (ligne d'intensité lumineuse identique) qui arrivent à la frontière $\delta\Omega$ du trou Ω (voir figure C.3).

Un terme de continuité est calculé le long d'une ligne isophote afin de calculer une esti-

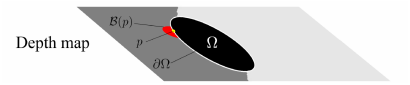


FIGURE C.3 – La zone inconnue est notée Ω, le pourtour de cette zone est la bande de pixel notée $\delta\Omega$. Le pixel à traiter situé sur la bande $\delta\Omega$ est p, son voisinage est noté $\beta(p)$. (Illustration tirée de [LGL12])

mation de la valeur du pixel inconnu situé à la frontière $\delta\Omega$ et dans la direction de la ligne isophote. Une étape de régularisation intervient en effectuant une diffusion anisotropique sur l'image à l'intervalle de plusieurs itérations successives de cette méthode d'*inpainting*,

La figure C.4 illustre l'utilisation de cette méthode appliquée à une image de profondeur.

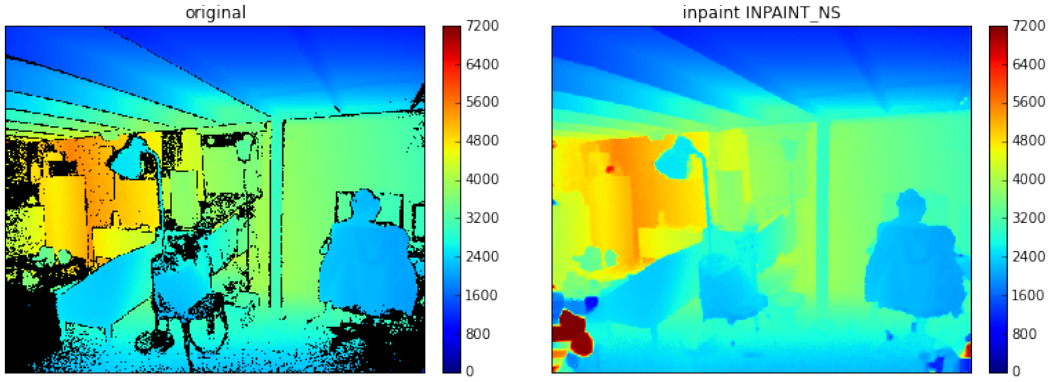


FIGURE C.4 – **a.** Image de profondeur originale. **b.** Image corrigée avec la méthode d'*inpainting* décrite dans [BSCB00].

C.1.4 *Fast Marching Method*

L'application de la méthode d'*inpainting* basée sur un filtre convolutif uniformise la vitesse de diffusion sur tout le pourtour $\delta\Omega$ de la zone Ω à obstruer. La méthode FMM (*Fast Marching Method*) [Tel04] permet de hiérarchiser l'ordre de traitement des pixels p sur la bande $\delta\Omega$ de manière adéquate (voir figure C.3). La diffusion est alors plus fine.

Lorsque le pixel p, est traité de nouveaux pixels inconnus entrent dans la bande $\delta\Omega$. Un coût T leur est assigné en fonction de leur distance à la bande $\delta\Omega$ originale. Les valeurs des pixels de la bande $\delta\Omega$ sont triées et le pixel p avec la quantité T la plus faible est à son tour traité. La méthode procède ainsi de suite jusqu'à ce que la bande $\delta\Omega$ ne contienne plus de pixels.

La méthode utilisée pour traiter un pixel p est décrite par la formule C.7. Il s'agit d'une somme pondérée sur les pixels voisins valides appartenant à $\beta(p)$.

$$D(p) = \frac{\sum_{q \in \beta(p)} \omega(p,q)[D(q) + \nabla D(q)(p-q)]}{\sum_{q \in \beta(p)} \omega(p,q)} \tag{C.7}$$

Le terme de pondération $\omega(p,q)$ est la multiplication des 3 termes suivants :

$$\omega_{dst}(p,q) = \frac{1}{\|p-q\|_2^2} \tag{C.8}$$

$$\omega_{dir}(p,q) = \frac{p-q}{\|p-q\|_2^2} \nabla T(p) \tag{C.9}$$

$$\omega_{lev}(p,q) = \frac{1}{1 + \|T(p) - T(q)\|_2} \tag{C.10}$$

Cette méthode a l'avantage d'être plus simple qu'une méthode d'optimisation globale puisqu'elle ne nécessite pas de calcul d'inversion de matrice. Elle est toutefois plus fine qu'une méthode basée sur un simple filtre puisqu'elle conditionne l'ordre de traitement des pixels. Le résultat de cette méthode appliqué à une image de profondeur est visible sur la figure C.5.

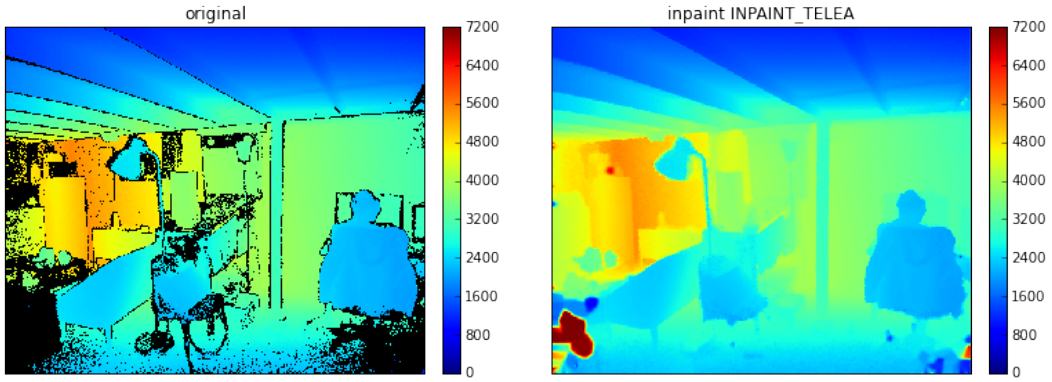


FIGURE C.5 – **a.** Image de profondeur originale. **b.** Image corrigée avec la méthode d'*inpainting* décrite dans [Tel04].

C.2 Utilisation de l'image d'intensité lumineuse comme guide pour l'amélioration de la carte de profondeur

C.2.1 Approche basée sur un filtre de convolution avec l'utilisation d'un guide.

C.2.1.1 Filtre bilatéral

Les approches dévelopées ci-dessous partagent l'utilisation d'un filtre pour améliorer la carte de profondeur. Le filtre défini par son noyau (*kernel*) est appliqué (convolution) à chaque pixel de l'image. Ces filtres permettent de lisser les zones homogènes pour obtenir une cohérence spatiale, tout en conservant les ruptures nettes qui apparaissent aux contours des objets de la scène. Il s'agit de préserver les contours (famille des filtre : *edge-preserving*).

Le filtre fondateur de cette catégorie est le filtre bilatéral [TM98]. Il s'agit en fait d'un noyau composé d'un filtre gaussien sur le voisinage et sur l'intensité des pixels. Ainsi, si l'écart d'intensité entre le pixel courant et le pixel voisin est trop important, l'impact du pixel voisin sur le filtrage du pixel courant est limité. De cette façon, les contours seront préservés tandis que les zones homogènes sont dé-bruitées. Ce filtre est mathématiquement décrit dans l'équation suivante C.11. Les paramètres σ_s et σ_r ajustent respectivement la similarité spatiale et la similarité d'intensité. Le filtrage bilatéral joint [KCLU07] prend en compte une image guide I différente de l'image à filtrer. La figure C.6 illustre l'effet d'un filtre bilatéral joint pour l'*upsampling* d'une image de profondeur avec comme guide l'image couleur haute résolution.

$$q_i = \sum_{j \in \omega_i} W_{ij}(I) p_j \tag{C.11}$$

avec :

$$W_{ij}(I) = \frac{1}{K_i} \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma_s^2}\right) \exp\left(-\frac{\|I_i - I_j\|_2^2}{\sigma_r^2}\right) \tag{C.12}$$

L'application qui nous intéresse consiste à filtrer l'image de profondeur P en prenant pour guide l'image couleur I associée. En effet, la technologie actuelle des capteurs de profondeur délivre des images de faible résolution où les contours sont particulièrement incertains. Les images en couleur ne contiennent pas d'informations directement sur la profondeur mais ont l'avantage de présenter des contours précis. Une vigilance subsiste : s'il n'y a pas de discontinuité dans la couleur à la frontière de deux objets alors il n'est pas possible d'avoir un contour précis.

Il est fait référence à l'utilisation du filtre bilatéral joint pour l'amélioration des cartes de profondeur dans ces travaux [CS12, KCKA10]. Une autre méthode est présentée dans [YYDN07]. L'image de profondeur est suréchantillonnée à la résolution de l'image couleur.

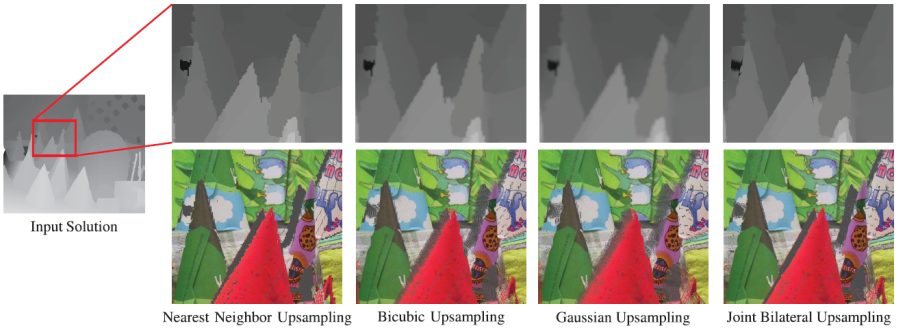


FIGURE C.6 – *Upsampling* d'une carte de profondeur basse résolution avec comme guide l'image couleur haute résolution associée. On peut voir que la méthode basée sur le filtre bilatéral est visuellement plus performante qu'une autre méthode d'interpolation.

Une méthode itérative basée sur l'utilisation d'un filtre bilatéral va affiner les contours de l'image de profondeur.

Le filtre bilatéral n'est pas linéaire. Par ailleurs il est tout de même possible de le décomposer en deux passes tout en obtenant une approximation très satisfaisante du filtre original (voir *Practiral Rendering and Computation with Direct3D 11*). On peut également se référer à [PD09] qui explicite une rapide approximation. Une amélioration de l'implémentation du filtre bilatéral joint est décrite dans [DBPT10].

C.2.1.2 Extension du filtre bilatéral

Les articles [GMO+10, GAM+11, GAA+12, GAS+13] font échos à la thèse de *Frédéric Garcia* concernant l'amélioration des cartes de profondeur par la fusion en utilisant des images d'intensité de haute résolution. L'auteur met en évidence certaines limites du filtre bilatéral joint, notamment le transfert de la texture de l'image couleur dans l'image de profondeur. Le filtre qu'il décrit s'appelle PWAS pour *Pixel Weighted Average Strategy*. Une carte de crédibilité C est établie à partir de la carte de profondeur P. Cette carte de crédibilité est ensuite injectée dans la création du noyau de filtrage tel que décrit dans l'équation C.13. Il s'agit d'une amélioration du filtre bilatéral joint.

$$q_i = \sum_{j \in \omega_i} W_{ij}(I) C_i j p_j \tag{C.13}$$

Diverses variations sont également éprouvées pour améliorer l'efficacité de ce filtre, dont l'expérimentation d'un filtre unifié qui correspond à la somme d'un filtre bilatéral sur la carte de profondeur P et du filtre PWAS décrit précédemment. Les termes sont pondérés par la carte de crédibilité C tel que décrit dans l'équation C.14.

$$q_i = (1 - C_i) \sum_{j \in \omega_i} W_{ij}(I) C_i j p_j + C_i \sum_{j \in \omega_i} W_{ij}(P) C_i j p_j \tag{C.14}$$

La figure C.7 illustre l'utilisation de ces filtres sur une image de profondeur.

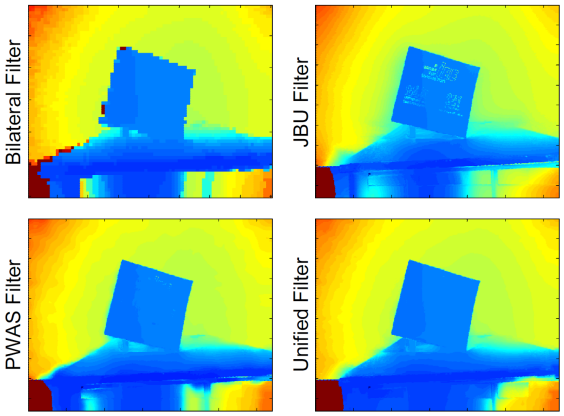


FIGURE C.7 – Filtrage d'une carte de profondeur avec les méthodes de filtrage suivantes : bilatéral, bilatéral joint, PWAS et filtre unifié. (Illustration tirée de [GAM+11])

Enfin, une autre extension du filtre bilatéral intitulée *Noise Aware Filter Depth Upsampling* (NAFDU) est présentée dans [CBTT08]. Un terme de *blending* permet de basculer l'image guide entre l'image de profondeur ou l'image couleur haute résolution. Cette méthode se rapproche du filtre unifié développé précédemment.

C.2.2 Méthodes d'optimisation globale

Dans l'article [VLK12], l'auteur commence par obstruer les trous des images de profondeur à l'aide d'une méthode de convolution qui définit les poids du noyau par une gaussienne sur l'image d'intensité associée. Une étape de diffusion anisotropique [PM90] est ensuite effectuée pour améliorer la carte de profondeur. Cette diffusion est appliquée au sein d'un framework pyramidal. Le résulat de l'utilisation de cette méthode est représentée par la figure C.8.

Une méthode basée sur une approche variationnelle du second ordre est décrite dans [FRR+13]. Un terme prenant en compte les discontinuités dans la texture couleur est introduit dans le système à optimiser. Un résultat est visible sur la figure C.9.

Une autre méthode temps réel, basée également sur une approche variationnelle, est décrite dans [ZNI+14]. L'implémentation est effectuée au GPU à l'aide d'un *successive over-relaxation* SOR solver. Enfin, l'article [HKH+13] décrit une méthode d'*inpainting* spécifiques aux cartes de profondeur, basée sur un critère de continuité de la dérivée seconde (*laplacian smoothness prior*). Un terme qui prend en compte l'image couleur est ajouté.

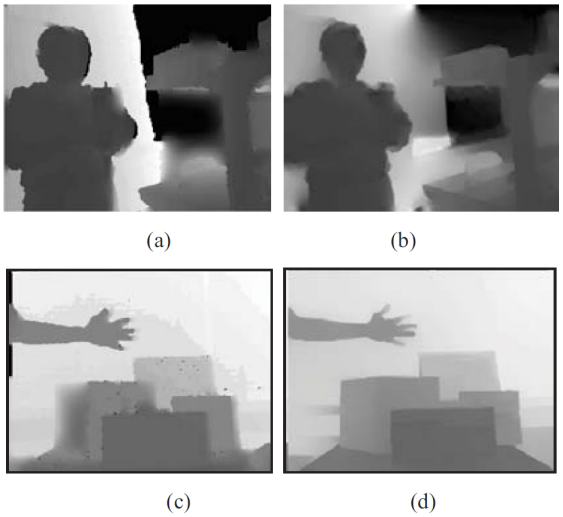


FIGURE C.8 – **(a)** et **(c)** correspondent à des images de profondeur obtenues après une étape d'*inpainting*. **(b)** et **(d)** correspondent à des images de profondeur obtenues après l'étape de diffusion anisotropique. *(Illustration tirée de [VLK12])*

FIGURE C.9 – Résultat du filtrage de la carte de profondeur avec la méthode d'optimisation globale décrite dans [FRR$^+$13]. *(Illustration tirée de [FRR$^+$13])*

C.2.3 *Guided Fast Marching Method* (GFMM)

La méthode d'*inpainting* GFMM utilisée dans [LGL12] correspond à une extension de la méthode [Tel04] FMM puisqu'elle inclut un terme de similarité basé sur l'image couleur associée (voir figure C.10).

Un coût E est assigné à chaque pixel p de la bande $\delta\Omega$ en fonction de sa distance T à la bande $\delta\Omega$ originale et en fonction du terme de similarité de la couleur C (voir equation C.7). Le terme λ sert à pondérer entre les termes T et C.

$$E = (1 - \lambda)T + \lambda(1 - C) \tag{C.15}$$

Le pixel p associé à la quantité E la plus faible est traité en premier avec l'équation C.7. Le terme de pondération $\omega(p, q)$ correspond à la multiplication des termes décrits dans les

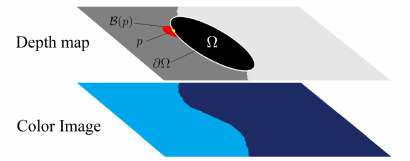


FIGURE C.10 – *Guided Fast Marching Method*. La zone inconnue est notée Ω, le pourtour de cette zone est la bande de pixel notée $\delta\Omega$. Le pixel à traiter, situé sur la bande $\delta\Omega$ est p, son voisinage est noté $\beta(p)$. L'image d'intensité lumineuse est associée à l'image de profondeur afin de guider l'*inpainting*.(Illustration tirée de [LGL12])

équations C.8,C.9 et C.10ainsi que le terme suivant C.16 :

$$\omega_{col}(p,q) = \exp\left(\frac{\|I(p) - I(q)\|_2^2}{2\sigma_c^2}\right) \tag{C.16}$$

L'utilisation de cette méthode est démontrée dans la figure suivante C.11. Un filtre guidé est associé pour améliorer le résultat.

FIGURE C.11 – Les trous d'une image de profondeur issue d'une *Kinect* sont obstrués avec une méthode d'*inpainting* FMM puis GFMM. Un filtrage guidé est effectué pour atténuer le bruit et affiner les contours. (Illustration tirée de [LGL12])

Annexe D

Clips de test

FIGURE D.1 – échantillon d'images issues des clips de test

Bibliographie

[APP+12] Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Wörgötter, and Babette Dellen. Depth-supported real-time video segmentation with the kinect. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 457–464. IEEE, 2012.

[BBZ+14] G Briand, F Bidgolirad, JF Zlapka, JM Lavalou, M Lanouiller, M Christie, J Lvoff, P Bertolino, and E Guillou. On-set previsualization for vfx film production. In *International Broadcasting Convention (IBC). Netherland, Amsterdam*, 2014.

[Ber14] Pascal Bertolino. Sensarea, a general public video editing application. In *21st IEEE International Conference on Image Processing*, Paris, France, october 2014.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV : Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[Bli94] James F Blinn. Compositing. 1. theory. *Computer Graphics and Applications, IEEE*, 14(5) :83–87, 1994.

[Bou14] Jean-Yves Bouguet. Matlab calibration tool, 2014.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[Bri08] Ron Brinkmann. *The art and science of digital compositing : techniques for visual effects, animation and motion graphics.* Morgan Kaufmann, 2008.

[Bro71] D.C. Brown. Close-range camera calibration. *Photogramm. Eng*, 37(8) :855–866, 1971.

[BSCB00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[BV06] Yuri Boykov and Olga Veksler. Graph cuts in vision and graphics : Theories and applications. In *Handbook of mathematical models in computer vision*, pages 79–96. Springer, 2006.

[BW09] Amit Bleiweiss and Michael Werman. Fusing time-of-flight depth and color for real-time segmentation and tracking. *Dynamic 3D Imaging*, pages 58–69, 2009.

[Can86] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, 1986.

[CBTT08] Derek Chan, Hylke Buisman, Christian Theobalt, and Sebastian Thrun. A noise-aware filter for real-time depth upsampling. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*, 2008.

[CP11] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1) :120–145, 2011.

[CPD07] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics (TOG)*, 26(3) :103, 2007.

[CS00] Brian Curless and Steven Seitz. 3d photography. *Course Notes for SIGGRAPH 2000*, 2000.

[CS12] Massimo Camplani and Luis Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. In *IS&T/SPIE Electronic Imaging*, pages 82900E–82900E. International Society for Optics and Photonics, 2012.

[DBPT10] Jennifer Dolson, Jongmin Baek, Christian Plagemann, and Sebastian Thrun. Upsampling range data in dynamic environments. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1141–1148. IEEE, 2010.

[DCSCO12] Meir Johnathan Dahan, Nir Chen, Ariel Shamir, and Daniel Cohen-Or. Combining color and depth for enhanced image segmentation and retargeting. *The Visual Computer*, 28(12) :1181–1193, 2012.

[DDH+15] Julia Diebold, Nikolaus Demmel, Caner Hazırbaş, Michael Moeller, and Daniel Cremers. Interactive multi-label segmentation of rgb-d images. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 294–306. Springer, 2015.

[dGB15] Timothee de Goussencourt and Pascal Bertolino. Using the unity® game engine as a platform for advanced real time cinema image processing. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4146–4149. IEEE, 2015.

[DM08] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, page 31. ACM, 2008.

[DMZC10] Carlo Dal Mutto, Pietro Zanuttigh, and Guido M Cortelazzo. Scene segmentation by color and depth information and its applications. *STreaming Day*, 2010.

[EL93] John Ens and Peter Lawrence. An investigation of methods for determining depth from focus. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(2) :97–108, 1993.

[FB86] John G Fryer and Duane C Brown. Lens distortion for close-range photogrammetry. *Photogrammetric engineering and remote sensing*, 52(1) :51–58, 1986.

[FHM+93] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Gérard Berry, Jean Vuillemin, et al. Real time correlation-based stereo : algorithm, implementations and applications. Technical report, Inria, 1993.

[FRR+13] David Ferstl, Christian Reinbacher, Rene Ranftl, Matthias Rüther, and Horst Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 993–1000, 2013.

[GAA+12] Frederic Garcia, Djamila Aouada, Hashim Kemal Abdella, Thomas Solignac, Bruno Mirbach, and Björn Ottersten. Depth enhancement by fusion for passive

and active sensing. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 506–515. Springer, 2012.

[GAM+11] Francisco Garcia, Djamila Aouada, Bruno Mirbach, Thomas Solignac, and Bjorn Ottersten. A new multi-lateral filter for real-time depth enhancement. In *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, pages 42–47. IEEE, 2011.

[GAS+13] Francisco Garcia, Djamila Aouada, Thomas Solignac, Bruno Mirbach, and Bjorn Ottersten. Real-time depth enhancement by fusion for rgb-d cameras. *Computer Vision, IET*, 7(5) :1–11, 2013.

[GHTC03] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8) :930–943, 2003.

[GJRW15] Ling Ge, Ran Ju, Tongwei Ren, and Gangshan Wu. Interactive rgb-d image segmentation using hierarchical graph cut and geodesic distance. In *Pacific Rim Conference on Multimedia*, pages 114–124. Springer, 2015.

[GKHE10] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141–2148. IEEE, 2010.

[GMO+10] Frederic Garcia, Bruno Mirbach, Bjorn Ottersten, Frédéric Grandidier, and Angel Cuesta. Pixel weighted average strategy for depth sensor data fusion. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2805–2808. IEEE, 2010.

[GMP12] J George, J Minard, and A Porter. Rgbd toolkit. *STUDIO for Creative Inquiry at Carnegie Mellon University*, 2012.

[GO11] Eduardo SL Gastal and Manuel M Oliveira. Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics (TOG)*, 30(4) :69, 2011.

[GO12] Eduardo SL Gastal and Manuel M Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics (TOG)*, 31(4) :33, 2012.

[Gre50] Allen R Greenleaf. *Photographic optics*. Macmillan, 1950.

[HBEC14] Steven Hickson, Stan Birchfield, Irfan Essa, and Henrik Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 344–351, 2014.

[HKH+13] Daniel Herrera, Juho Kannala, Janne Heikkilä, et al. Depth map inpainting under a second-order smoothness prior. In *Image Analysis*, pages 555–566. Springer, 2013.

[Hor86] Berthold Horn. *Robot vision*. MIT press, 1986.

[HST13] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(6) :1397–1409, 2013.

[IKH+11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion : real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[JKJ+13] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset : Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision*, pages 141–165. Springer, 2013.

[JY12] Bing-Zhong Jing and Daniel S Yeung. Recovering depth from images using adaptive depth from focus. In *Machine Learning and Cybernetics (ICMLC), 2012 International Conference on*, volume 3, pages 1205–1211. IEEE, 2012.

[KCKA10] Sung-Yeol Kim, Ji-Ho Cho, Andreas Koschan, and Mongi A Abidi. Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2358–2361. IEEE, 2010.

[KCLU07] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (TOG)*, 26(3) :96, 2007.

[KM07] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[KTPW11] Martin Knecht, Christoph Traxler, Werner Purgathofer, and Michael Wimmer. Adaptive camera-based color mapping for mixed-reality applications. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 165–168. IEEE, 2011.

[KW93] Hans Knutsson and Carl-Fredrik Westin. Normalized and differential convolution. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pages 515–523. IEEE, 1993.

[LGL12] Junyi Liu, Xiaojin Gong, and Jilin Liu. Guided inpainting and filtering for kinect depth maps. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2055–2058. IEEE, 2012.

[LLW04] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 689–694. ACM, 2004.

[LMM+15] E Lachat, H Macher, MA Mittet, T Landes, and P Grussenmeyer. First experiences with kinect v2 sensor for close range 3d modelling. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5) :93, 2015.

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166, 2009.

[LSP06] Hao Li, Raphael Straub, and Hartmut Prautzsch. Structured light based reconstruction under local spatial coherence assumption. In *3D Data Processing,*

Visualization, and Transmission, Third International Symposium on, pages 575–582. IEEE, 2006.

[MBC11] Cyrille Migniot, Pascal Bertolino, and Jean-Marc Chassery. Automatic people segmentation with a template-driven graph cut. In *2011 18th IEEE International Conference on Image Processing*, pages 3149–3152. IEEE, 2011.

[MFL+12] Dan Miao, Jingjing Fu, Yan Lu, Shipeng Li, and Chang Wen Chen. Texture-assisted kinect depth inpainting. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 604–607. IEEE, 2012.

[NLB+05] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11) :1–11, 2005.

[OBMC01] Manuel M Oliveira, Brian Bowen, Richard McKenna, and Yu-Sung Chang. Fast digital image inpainting. In *Appeared in the Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain*, pages 106–107, 2001.

[Ope10] OpenNI organization. *OpenNI User Guide*, November 2010. Last viewed 19-01-2011 11 :32.

[PD09] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. *International journal of computer vision*, 81(1) :24–52, 2009.

[PGKT10] Christian Plagemann, Varun Ganapathi, Daphne Koller, and Sebastian Thrun. Real-time identification and localization of body parts from depth images. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3108–3113. IEEE, 2010.

[PM90] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7) :629–639, 1990.

[Pri10] PrimeSense Inc. *Prime Sensor^{TM} NITE 1.3 Algorithms notes*, 2010. Last viewed 19-01-2011 15 :34.

[RHHL02] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 438–446. ACM, 2002.

[SBK+08] Ingo Schiller, Bogumil Bartczak, Falko Kellner, Jan Kollmann, and Reinhard Koch. Increasing realism and supporting content planning for dynamic scenes in a mixed reality system incorporating a time-of-flight camera. In *Visual Media Production (CVMP 2008), 5th European Conference on*, pages 1–10. IET, 2008.

[SBX14] Martin Stommel, Michael Beetz, and Weiliang Xu. Inpainting of missing values in the kinect sensor's depth maps based on background estimates. *Sensors Journal, IEEE*, 14(4) :1107–1116, 2014.

[SHKZ14] Ling Shao, Jungong Han, Pushmeet Kohli, and Zhengyou Zhang. *Computer vision and machine learning with RGB-D sensors*. Springer, 2014.

[SK11] Ingo Schiller and Reinhard Koch. Improved video segmentation by adaptive combination of depth keying and mixture-of-gaussians. In *Scandinavian conference on Image analysis*, pages 59–68. Springer, 2011.

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1) :116–124, 2013.

[SZ13] Alexander Shpunt and Zeev Zalevsky. Three-dimensional sensing using speckle patterns, March 5 2013. US Patent 8,390,821.

[TCLBI13] P Trouve, F Champagnat, G Le Besnerais, and J Idier. Estimation de profondeur à partir du flou de défocalisation d'un imageur chromatique. In *XXIVème Colloque Gretsi 2013*, 2013.

[Tel04] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1) :23–34, 2004.

[THMR13] Michael Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 673–680, 2013.

[Tho06] Graham Thomas. Mixed reality techniques for tv and their application for on-set and pre-visualization in film production. In *International Workshop on Mixed Reality Technology for Filmmaking*, pages 31–36, 2006.

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[VL01] Etienne Vincent and Robert Laganiere. Matching feature points in stereo pairs : A comparative study of some matching strategies. *Machine Graphics and Vision*, 10(3) :237–260, 2001.

[VLK12] Krishna Rao Vijayanagar, Maziar Loghman, and Joohee Kim. Refinement of depth maps generated by low-cost depth sensors. In *SoC Design Conference (ISOCC), 2012 International*, pages 355–358. IEEE, 2012.

[WBH11] Jakob Wasza, Sebastian Bauer, and Joachim Hornegger. Real-time preprocessing for dense 3-d range imaging on the gpu : defect interpolation, bilateral temporal averaging and guided filtering. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1221–1227. IEEE, 2011.

[WFFD11] Marcus Wallenberg, Michael Felsberg, Per-Erik Forssén, and Babette Dellen. Channel coding for joint colour and depth segmentation. In *Joint Pattern Recognition Symposium*, pages 306–315. Springer, 2011.

[WTRF09] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12) :2115–2128, 2009.

[XS93] Yalin Xiong and Steven A Shafer. Depth from focusing and defocusing. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pages 68–73. IEEE, 1993.

[YMB77] William A Yasnoff, Jack K Mui, and James W Bacus. Error measures for scene segmentation. *Pattern recognition*, 9(4) :217–231, 1977.

[YTA09] Qingxiong Yang, Kar-Han Tan, and Narendra Ahuja. Real-time o (1) bilateral filtering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 557–564. IEEE, 2009.

[YYDN07] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-depth super resolution for range images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673. IEEE, 1999.

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11) :1330–1334, 2000.

[ZNI+14] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33(4) :156, 2014.

[ZZ14] Cha Zhang and Zhengyou Zhang. Calibration between depth and color sensors for commodity depth cameras. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 47–64. Springer, 2014.

Résumé — La *previz on-set* est une étape de prévisualisation qui a lieu directement pendant la phase de tournage d'un film à effets spéciaux. Cette proposition de prévisualisation consiste à montrer au réalisateur une vue assemblée du plan final en temps réel. Le travail présenté dans cette thèse s'intéresse à une étape spécifique de la prévisualisation : le *compositing*. Cette étape consiste à mélanger plusieurs sources d'images pour composer un plan unique et cohérent. Dans notre cas, il s'agit de mélanger une image de synthèse avec une image issue de la caméra présente sur le plateau de tournage. Les effets spéciaux numériques sont ainsi ajoutés à la prise de vue réelle. L'objectif de cette thèse consiste donc à proposer un système permettant l'ajustement automatique du mélange entre les deux images. La méthode proposée nécessite la mesure de la géométrie de la scène filmée. Pour cette raison, un capteur de profondeur est ajouté à la caméra de tournage. Les données sont relayées à l'ordinateur qui exécute un algorithme permettant de fusionner les données du capteur de profondeur et de la caméra de tournage. Par le biais d'un démonstrateur matériel, nous avons formalisé une solution intégrée dans un moteur de jeux vidéo. Les expérimentations menées montrent dans un premier temps des résultats encourageants pour le *compositing* en temps réel. Nous avons observé une amélioration des résultats suite à l'introduction de la méthode de segmentation conjointe. La principale force de ce travail réside dans la mise en place du démonstrateur qui nous a permis d'obtenir des algorithmes efficaces dans le domaine de la *previz on-set*.

Mots clés : prévisualisation, temps réel, moteur de jeux vidéo, capteur de profondeur, segmentation RGBD, *Unity*, *compositing*, réalité augmentée, *Kinect*.

Abstract — On-set previs is a preview step that takes place directly during the shooting phase of a film with special effects. The aim of on-set previs is to show to the film director an assembled view of the final plan in realtime. The work presented in this thesis focuses on a specific step of the previs : compositing. This step consists in mixing multiple images to compose a single and coherent one. In our case, computer graphics images and film camera images must be mixed. The objective of this thesis is to propose a system for automatic adjustment of the compositing. The method requires the measurement of the geometry of the filmed scene. For this reason, a depth sensor is added to the main camera. The data is sent to the computer that executes an algorithm to merge data from depth sensor and the main camera. Through a hardware demonstrator, an integrated solution in a video game engine was formalized. The experiments gives encouraging results for realtime compositing in real time. Improved results were observed with the introduction of a joint segmentation method using both depth and color information. The main strength of this work lies in the development of a demonstrator that allowed us to obtain effective algorithms in the field of on-set previs.

Keywords : on-set previs, realtime, game engine, depth sensor, RGBD segmentation, *Unity*, compositing, augmented reality, *Kinect*.