Titre: Title:	Apprentissage de modèles probabilistes pour la vision stéréoscopique en temps réel
Auteur: Author:	Lucas Berthou
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Berthou, L. (2012). Apprentissage de modèles probabilistes pour la vision stéréoscopique en temps réel [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/991/

UNIVERSITÉ DE MONTRÉAL

APPRENTISSAGE DE MODÈLES PROBABILISTES POUR LA VISION STÉRÉOSCOPIQUE EN TEMPS RÉEL

LUCAS BERTHOU
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

APPRENTISSAGE DE MODÈLES PROBABILISTES POUR LA VISION STÉRÉOSCOPIQUE EN TEMPS RÉEL

présenté par : BERTHOU Lucas
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme CHERIET Farida, Ph.D., présidente.
M PAL Christopher J., Ph.D. membre et directeur de recherche.
M SEKKATI Hicham, Ph.D., membre.

"L'important c'est pas
de comprendre ce qu'on
fait, c'est de pouvoir
mettre des photos
sur Facebook."
Étienne Maxou...
Et plus sérieusement à
tout ceux qui ont rendu
cette maîtrise possible.

REMERCIEMENTS

Dans un premier lieu je tiens tout particulièrement à remercier Christopher J. Pal, mon advisor au cours de cette maîtrise, pour l'ensemble de ses conseils, ses explications, sa disponibilité et sa receptivité. Je tient également à remercier l'ensemble des membres du laboratoire pour leur aide et leur soutient, plus particulièrement Fannie, Louis-Philippe, David, Simon et Ling.
Je souhaite également remercier l'équipe pédagogique pour les connaissances qu'elle m'a apporté au cours de ces deux années passées à l'école Polytechnique Montréal. Ainsi que l'équipe administrative qui, malgré quelque faute de parcours a fait de son mieux et m'a grandement facilité la vie au cours de mes études.
Et je remercie également toutes les personnes qui m'ont prodigué des conseils que ce soit sur mes recherches ou sur la rédaction de ce mémoire.

Je souhaite également remercier l'ensemble de ma famille et de mes amis pour le soutient et le réconfort qu'ils m'ont apporté tout au long de cette entreprise. Les nommer tous serait long et fastidieux mais je pense notamment à Yves, Gautier, Margaux, Jean, Marion, Jérôme, Adrien, Amaury, Louis, Patricia, Jean-Baptiste, Virgile, César et Daniel. Je n'oublie pas non plus tout ceux chez qui j'ai pu être amené à travailler, échanger, réfléchir ou simplement passer la nuit au cours de mes tribullations.

Je remercie également tout ceux que j'oublie ou qui estiment qu'ils ne sont pas assez cités dans les recouvrements des cathégories précédentes, et les prie de bien vouloir m'en excuser.
Enfin je remercie Raoul pour son affection et sa patience.

RÉSUMÉ

Il existe de nombreuses approches pour capturer des scènes en trois dimensions, la plus couramment utilisée est d'avoir recours à la stéréoscopie, qu'elle soit active ou passive. Le principe sous-jacent à de telles techniques est toujours le même et consiste à retrouver des correspondances aux travers différentes prises de vues. La recherche de ces correspondances aboutit à la création de cartes de disparités. Nous présentons ici une étude de différentes approches, aussi bien passive qu'active, pour construire de telles cartes. Nous nous intéressons également aux modèles probabilistes qui permettent leur débruitage et l'amélioration des résultats obtenus. Enfin, nous proposons également une approche basée sur l'utilisation de modèles de base et la combinaison de différentes techniques de calcul de disparités pour construire notre propre modèle.

Il existe deux critères d'évaluation pour les approches de numérisation tridimensionnelle : le temps de traitement et la qualité des captures. Contrairement à d'autres systèmes cherchant à optimiser seulement l'un de ces deux critères, les résultats des travaux présentés ici sont obtenus en concentrant nos efforts sur la qualité mais aussi sur le temps nécessaire aux calculs. Pour cette dernière raison, nous avons choisi d'utiliser des techniques de parallélisation massive sur processeur graphique (General Purpose on Graphical Processor Unit (GPGPU)). Nous étudierons donc des implémentations parallélisées et optimisées pour traiter nos images stéréoscopiques ou nos cartes de disparités, tels que l'algorithme du Census ou de Viterbi, entièrement sur un processeur graphique.

Enfin nous verrons comment combiner divers sources d'informations telles qu'une Kinect et une caméra stéréo pour obtenir la meilleure qualité de carte de disparités possible. Nous verrons également un montage optimisé pour la numérisation de visages et l'évaluation et la comparaison de nouveaux modèles.

ABSTRACT

Among the various existing strategies to capture 3D information out of a scene, the most commonly used is stereoscopy, either active or passive. Underneath theses strategies there is always the same principle of finding corresponding points across captured views. Dense corresponding points are used to generate a disparity map that can then be transformed in a depth map. This shows the importance of finding good approaches to build theses disparity maps. We will explore several approaches both active and passive to do so. We also present various probabilistic models that allow one to denoise data and improve the quality of our disparity maps. We will also introduce a new way of combining these models and various strategies of getting disparity to build our own new model.

We present a probabilistic framework to compute disparity maps focusing on both quality, efficiency and speed. To reduce the time needed for computing our disparity maps we chose to use general purpose computing on graphics processing units techniques to massively parallelize our algorithms. Hence we will present some optimized parallelized implementations of algorithms to treat both stereoscopic images and disparity maps such as the computation of Census descriptors and the Viterbi algorithm, such that all the processing is done on a GPU.

Finally we show how to combine various sources by adding a Kinect to our model based on stereo camera to improve either the quality of our outputted disparity map or the time required by our algorithms. We present a camera rig configuration optimized for face scanning, evaluate and compare different models and techniques for combining projected light techniques with passive stereo.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

CAM	Champ Aléatoire de Markov (CRF)
CPU	Central Processing Unit (processeur)
GPGPU	General Purpose on Graphical Processor Unit
MMC	Modèle de Markov Caché (HMM)

CHAPITRE 1

INTRODUCTION

Un très grand nombre de tâches que nous sommes amenés à réaliser dans la vie de tous les jours nous sont accessibles uniquement grâce à notre perception tridimensionnelle de l'espace qui nous entoure. Qu'il s'agisse de se saisir d'un objet, d'éviter un obstacle ou encore de se servir un verre d'eau, la perception de profondeur est primordiale. Il est donc facile d'imaginer l'importance accordée à la troisième dimension en vision par ordinateur, que ce soit pour la navigation robotique (Curiosity), pour le jeu vidéo (Kinect), pour le domaine médical (détection de scoliose), ou la réalisation de film (Avatar). Si nous sommes capables en tant qu'être humain de percevoir la 3D c'est grâce à notre vision stéréoscopique, par analogie le même principe est donc utilisé dans la plupart des systèmes de capture de profondeur en informatique. Le point clé derrière la vision stéréoscopique est l'établissement de correspondances à travers diverses vues. Ces correspondances permettent d'établir ce qu'on appelle des cartes de disparités qui, une fois transformées, permettent à leur tour d'obtenir des cartes de profondeurs, de la 3D.
Il existe de nombreuses méthodes pour construire ces cartes de disparités, chacune ayant ses avantages et ses inconvenients. Lorsque l'on connait la véritable carte de dispariés, il est toutefois possible de comparer la précision des cartes de disparités obtenues ainsi que le temps de calcul nécessaire pour chaque méthode.
Enfin il est intéressant de considérer que, même si l'on obtient une carte de profondeurs de qualité moyenne il est possible d'avoir recours à des techniques d'apprentissage et des méthodes probabilistes afin de la débruiter.

1.1 Définitions et concepts de base

1.1.1 Carte de disparités

La disparité, en vision stéréo, est communément définie comme le déplacement qu'il est nécessaire de faire subir à un point, d'une vue donnée, pour le superposer à un point correspondant dans une autre vue (Brown et al., 2003, p. 994). La Figure 1.1 illustre la provenance de cette disparité, son lien avec la profondeur dans le cadre de caméras trou d'épingle (pinhole). En effet, avec O_1 et O_2 nos cameras, T la distance les séparant, z la profondeur de notre point d'intéret, f la distance focale de nos caméras et $d = (X_2 - O_2) - (X_1 - O_1)$ la

disparité entre les projections de P sur chacun des capteurs, il est facile de démontrer que $z = f\frac{T}{d}$.

L'ensemble des disparités pour chacun des points d'une scène vue en stéréo constitue ce qu'on

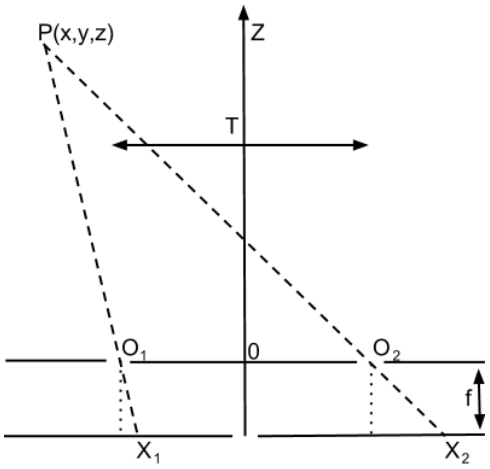

Figure 1.1 Stéréo dans le cadre de caméra trou d'épingle

appelle une carte de disparités. Il est facile, une fois les caméras calibrées ensemble (calcul de T et f dans notre cas simpliste) de transformer cette dernière en carte de profondeurs.

Cependant, dans la pratique, on utilise très peu de caméra pinhole, qui nécessite de longues expositions, des scènes statiques et un bon éclairrage. En rajoutant un systeme optique, on ne modifie pas foncièrement le principe de fonctionnement de la vision stéréoscopique, mais introduit des distorsions et des artefacts qu'il est important de prendre en considération. Une fois ces artéfacts déterminés, il est toutefois possible de rectifier et d'aligner les images en utilisant des lignes épipolaires afin d'annuler les distorsions et de permettre et de faciliter la création de carte de disparités.

Toute la difficulté lors du calcul de carte de disparités réside donc dans l'établissement de mise en correspondances précise entre les images capturées. Malheureusement, la correspondance pixel par pixel n'est pas unique, on peut avoir des textures répétitives ou une absence de texture (couleur unie). De plus nos captures peuvent être déformées, en effet on peut penser aux problèmes de l'éclairage qui peut faire varier différemment la couleur capturée sur chacun des capteurs ou encore le fait que chaque capteur possède un bruit propre par exemple. L'ensemble de ces variations est appelé ensemble des variations photométriques. Elles viennent compliquer la détection de correspondances. Enfin, il faut également considérer le problème d'occlusion qui survient lorsqu'un objet masque une partie de la scène pour seulement l'un des capteurs, il devient alors impossible de trouver une correspondance (Figure 1.2).

Il est donc nécessaire d'élaborer des stratégies prenant en compte ces différents paramètres

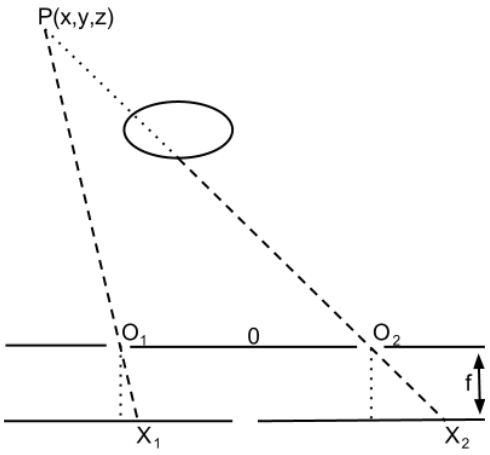

Figure 1.2 Occlusion du point P par un objet dans la caméra 2

pour établir des correspondances. La plus intuitive consiste à considérer des correspondances de voisinage, c'est la technique dite de correspondance par bloc, elle sera développée plus en détail plus loin. Afin de réduire le bruit et les variations de lumière, il peut être intéressant de considérer une version filtrée de nos images telles que dans la méthode dite du census, également expliquée plus en détail ultérieurement dans le chapitre sur les stratégies passives. Mais l'idée la plus sûre en pratique reste d'avoir un code unique pour chaque point de la scène, ceci peut se réaliser à l'aide de projections. Cette approche servira d'introduction aux autres dans la partie sur l'utilisation de la lumière structurée et autres stratégies actives.

1.1.2 Modèles probabilistes

Nous considérons ici des scènes que l'on numérise. Il arrive donc souvent que l'on ait un a priori sur la géométrie perçue (sur ce que l'on est supposé obtenir) soit par connaissance de scènes similaires soit grâce à des numérisations antérieures. L'utilisation de modèles probabilistes, tels que des Modèles de Markov Cachés (Modèle de Markov Caché (MMC)), permet d'utiliser ces connaissances a priori afin d'améliorer une carte de disparités. Il faut cependant faire attention à bien paramétrer notre modèle probabiliste afin d'éviter le surapprentissage et de ne pas dégrader notre carte en voulant trop coller à un a priori qui n'est pas réellement la scène d'intérêt.
De tels modèles sont souvent utilisés en posttraitement pour débruiter et combler les trous dans des cartes de disparités. Ils utilisent alors l'a priori sur la scène et une carte de disparités précalculée. Mais il est également possible d'utiliser de tels modèles en amont de la construction de la carte de disparités. Dans ce cas, l'apriori permet d'améliorer les points

correspondants entrainant une amélioration des disparités.
Enfin il peut être intéressant d'envisager différents modèles plus ou moins complexes quant à leur degré de liaison, au lieu d'un MMC on pourrait, par exemple, envisager d'utiliser un Champ Aléatoire de Markov (Champ Aléatoire de Markov (CAM)) ou plusieurs MMC croisés.

1.1.3 Parallélisation

Faire des calculs sur l'ensemble de l'image nécessite un nombre calculs lié au nombre de pixels et donc à la résolution de notre système de capture. Or, pour capturer un grand nombre de détails et pouvoir faire de bonnes mises en correspondances, il est intéressant d'avoir une haute résolution d'image. Cependant les calculs sont indépendants pour des voisinages et des valeurs de disparités différents. Établir des correspondances est donc hautement parallélisable et les algorithmes de calcul de disparités sont donc généralement bien adaptés au calcul sur processeur graphique (GPGPU). Cette parallélisation pour l'établissement de correspondances peut être poussée encore plus loin en parallélisant également les calculs nécessaires pour l'utilisation de modèles probabilistes.
L'utilisation de GPGPU permet de diminuer drastiquement le temps nécessaire à l'obtention de nos cartes de disparités sans en diminuer la qualité.

1.2 Éléments de la problématique

Il existe de nombreuses applications, touchant un grand nombre de domaines, reposant sur la reconstruction 3D par vision stéréoscopique. Ces applications nécessitent d'avoir une carte de profondeurs de bonne qualité en un temps raisonnable. Il existe certains périphériques, tel que la Kinect de Microsoft permettant d'obtenir de telles cartes en un temps raisonnable. Mais la qualité peut ne pas suffire pour certaines applications qui nécessite un rendu 3D précis par exemple ou des valeurs stables et précises et cohérentes au fil du temps, par ailleurs la technique utilisée repose sur de la stéréoscopie active, projetant un schéma infrarouge qui n'est pas toujours correctement réfléchi ou perçu. De plus, cela peut ne pas être souhaitable, car notre scène est alors altérée. Une alternative offrant une meilleure qualité, consiste à utiliser un scanner laser (Leica par exemple) qui permet d'obtenir une carte d'excellente qualité, mais qui nécessite beaucoup de temps en raison de la numérisation point par point ou line par ligne et de l'utilisation d'un laser. Enfin des méthodes basées sur l'utilisation de lumières structurées permettent d'obtenir une bonne carte de disparités, mais reposent une fois de plus sur la projection de schémas lumineux qui peuvent être indésirables et nécessitent un temps de capture rallongé.

Un premier champ de recherche restant grandement à explorer réside dans le choix des textures projetées dans le cadre de stéréoscopie active. En effet, entre la Kinect qui se contente d'un schéma infrarouge et un scanner à lumière structurée qui nécessite la projection d'une série de patrons en niveaux de gris, il y a de la marge laissée à l'exploration. Peut-être est-il possible de trouver une stratégie donnant de très bons résultats en un temps raisonnable.

Par ailleurs, l'ensemble de ces techniques repose sur de la stéréoscopie active, il serait intéressant de voir dans quelle mesure on peut s'en rapprocher en utilisant uniquement la stéréoscopie passive. Essayer de ne pas avoir recours à la projection de schémas lumineux et déterminer dans quelle mesure cela dégrade la carte de disparités produite. Les évolutions technologiques de ces dernières années font qu'il est de plus en plus facile d'obtenir des images de hautes définitions, mais également qu'il est maintenant possible de faire d'importants calculs beaucoup plus rapidement. L'utilisation d'images de meilleures résolutions devrait permettre d'obtenir une bonne carte de disparités lorsque la scène le permet, tandis que la parallélisation massive devrait permettre de réduire les temps de calcul nécessaires. Cette combinaison de facteurs devrait alors permettre d'obtenir des résultats que l'ont peut espérer similaire à l'état de l'art en termes de numérisation 3D dans des temps raisonnables et sans avoir recours à des techniques projectives.

De plus certaines approches donnant de très bons résultats, tant d'un point de vue précision que temps de calcul, recourent à la programmation dynamique et impose donc des choix

arbitraires là ou un modèle probabiliste serait plus permissif et mathématiquement plus approprié. Il peut donc être intéressant et bénéfique de modifier légèrement ces techniques afin d'améliorer les résultats obtenus. Il faut cependant trouver une bonne paramétrisation de nos modèles ainsi qu'un moyen efficace d'acquérir les connaissances a priori sur lesquelles ils sont basés.
De même, il est intéressant de voir que ces modèles peuvent être utilisé a posteriori (en aval), une fois notre carte de disparités initiale construite, pour faire un travail de débruitage ou, au contraire, a priori (en amont de la détermination des disparités initiale) pour déterminer une carte de disparités plus cohérente en débruitant cette fois les correspondances trouvées.
Enfin il est intéressant d'étudier le gain de qualité que peut nous apporter l'utilisation de ce que nous appellerons la stéréoscopie multimodale. Cette stratégie vise à utiliser plusieurs approches stéréoscopiques et à les combiner pour obtenir le meilleur de chacune et essayer de combler chacune de leurs lacunes. Il existe alors différentes approches envisageables quant à la combinaison que l'on souhaite faire, on peut par exemple utiliser un système de caméras stéréo pour raffiner des résultats obtenus par une Kinect ou au contraire utiliser les résultats d'un autre système pour accélérer les calculs ou nous donner un a priori sur notre scène afin de débruiter les cartes de disparités.

1.3 Objectifs de recherche

Les objectifs de recherche sont les suivants :

- Définir une méthodologie et un schéma expérimental permettant la comparaison de différentes techniques de conception de cartes de disparités.
- Explorer différentes méthodes pour le calcul de disparités et les implémenter efficacement en GPGPU.
- Utiliser des méthodes probabilistes pour remplacer la programmation dynamique de certaines approches.

1.4 Plan du mémoire

La suite de ce mémoire se décompose en six chapitres. Le deuxième chapitre présente une revue critique de la littérature pertinente dans le cadre de ce projet. Cette revue couvre diverses techniques employées en stéréo passive et active ainsi que le fonctionnement de scanner à lumière structurée et différents modèles probabilistes appropriés à notre étude.
Le troisième chapitre se concentre sur le fonctionnement, l'implémentation et l'amélioration d'un scanner à lumière structuré de sorte à pouvoir en comparer les résultats avec d'autres

approches. On y explore également quelques pistes pour la stéréoscopie active.
Le chapitre quatre présente sommairement le fonctionnement, l'implémentation et l'optimisation GPGPU de différents algorithmes passifs destinés à la création de cartes de disparités.
Le cinquième chapitre introduit un nouveau modèle probabiliste combinant les coûts de la correspondance par bloc et du census en amont de la création d'une carte de disparité.
Dans le chapitre numéro six nous introduisons un schéma expérimental à double utilité, permettant à la fois la combinaison de diverses sources pour l'obtention de cartes de disparités, mais également la comparaison de stratégies.
Le dernier chapitre propose une discussion générale sur les résultats obtenus par rapport aux objectifs mentionnés ici ainsi que plusieurs suggestions et propositions pour des améliorations et des travaux futurs. Ce chapitre vient conclure le mémoire

CHAPITRE 2

REVUE DE LITTÉRATURE

Dans ce chapitre nous présentons une revue critique de la littérature existante portant sur les sujets abordés dans la suite de ce mémoire. Cette revue de littérature se décompose en trois parties, la première est consacrée à la définition de cartes de disparités (2.1.1) et aux divers techniques visant à leur obtention utilisant (voir 2.1.2) ou non (voir 2.1.3) des projections lumineuses. La deuxième aborde les modèles probabilistes qui sont généralement utilisés pour améliorer les performances et la qualité des cartes de disparités. Enfin, la troisième traite d'autres méthodes de capture de profondeur.

2.1 Carte de disparités

2.1.1 Carte de disparités

Avant d'aborder réellement la notion de carte de disparités et pour bien comprendre d'où vient le déplacement d'un point dans différentes prises de vues, il est important de bien comprendre le fonctionnement d'un appareil photo. Richard Szeliski en donne une bonne explication (voir Szeliski, 2010, ch. 6). La prise d'une photo n'est en fait rien de plus que la projection d'une scène 3D sur un capteur 2D. Cette projection peut être décomposée en deux parties : a) un changement de repère pour transformer le monde et le placer dans le système de coordonnées de la camera, définissant ainsi les paramètres extrinsèques ; b) suivi de la projection des points sur le plan image de la camera, et qui définit les paramètres intrinsèques de la camera.

Pour la calibration d'un système de cameras stéréoscopiques, il faut déterminer les paramètres intrinsèques et extrinsèques pour chacune des caméras. Une fois ces paramètres obtenus (souvent sous forme d'une matrice extrinsèque et son inverse ainsi que deux matrices intrinsèques, une par capteur), et après avoir établit une correspondance entre les images il est aisé d'extraire une information de profondeur comme illustré en Figure 1.1. L'établissement de correspondances se fait généralement sous la forme de disparité entre les positions d'un même point dans plusieurs vues (voir Forsyth et Ponce, 2002).

La notion de disparité de la vision stéréoscopique a été introduite pour la première fois par David Marr en 1982 (voir Marr, 1982b,a). Il introduit également trois principes de base que doit respecter un système de disparité stéréo, la compatibilité qui se traduit par le fait que

les caractéristiques doivent correspondre à travers les différentes vues (les pixels correspondants doivent avoir, sensiblement, la même couleur). L'unicité visant à établir qu'un élément correspond au maximum à un et un seul autre élément dans une autre vue. Et la continuité qui assure que la disparité varie de manière continue sur une même surface.
Il existe deux familles d'approches pour établir une carte de disparité, les méthodes denses, faisant correspondre une disparité à chaque point, et les méthodes clairsemées, créant un ensemble non complet de couples point disparité(voir ISGRO *et al.*, 2004). Nous avons restreint notre étude aux cartes de disparités denses. Daniel Scharstein et Richard Szeliski ont publié en 2002 une excellente revue des différentes méthodes principalement utilisées dans cette perspective (voir Scharstein et Szeliski, 2002). Les cartes de disparités denses sont obtenues par corrélation d'un voisinage aux travers différentes vues, une fenêtre autour d'un pixel est comparée à l'ensemble des fenêtres de même taille dans l'autre vue, la fenêtre maximisant la corrélation donne alors une valeur de disparité pour notre point.
La recherche dans l'intégralité de l'image pouvant s'avérer couteuse les images sont généralement corrigées et alignées pour pouvoir restreindre la recherche que le long d'une courbe, idéalement une droite, correspondant à une ligne horizontale, ou verticale, de pixels dans nos images (voir Szeliski, 2006; Ayache et Hansen, 1988). Nous avons donc ramené notre étude à ce cas particulier généralement adopté (voir Scharstein et Szeliski, 2002).
Un autre point important lors de la construction de carte de disparités est le choix de la métrique utilisée pour mesurer la corrélation. Même lorsque l'on considère l'algorithme de correspondance par bloc, il est possible d'utiliser une distance fondée sur l'erreur au carré ou sur la valeur absolue de l'erreur. Il est également possible d'utiliser des images en niveaux de gris ou en couleur, enfin la taille d'un bloc est également à choisir judicieusement afin d'obtenir les meilleurs résultats possibles (voir Koschan, 1993). Il existe d'autres modifications possibles pour la métrique, elles seront détaillées plus loin dans la section 2.1.3.

2.1.2 Méthodes Actives

Afin d'évaluer la qualité d'une carte de disparités, où ne serait-ce que pour avoir un a priori nécessaire à certains modèles probabilistes (voir 2.2), il est nécessaire d'avoir une base de vérité de la scène que l'on souhaite numériser. Il existe plusieurs moyens de mesurer précisément une scène, on peut par exemple utiliser un scanner laser (voir Biber *et al.*, 2004) ou utiliser des rendus 3D (OpenGL) pour tester un algorithme (voir Marichal-Hernandez *et al.*, 2006). Toutefois, la méthode principalement utilisée dans le cadre de numérisation stéréoscopique consiste à utiliser de la lumière structurée (voir Lanman et Taubin, 2009).
L'utilisation de lumière structurée (comme du laser) requiert cependant de projeter de la lu-

mière sur une scène. Toutefois, elle présente le double avantage d'être précise et de pouvoir se superposer à une capture stéréoscopique sans nécessiter la moindre calibration additionnelle (voir Scharstein et Szeliski, 2003). Cette technique repose intégralement sur le principe d'unicité de Marr, chaque point de la première vue doit être rendu unique et ainsi ne correspondre qu'à un et un seul point de la seconde. On souhaite de plus éviter autant que possible les erreurs de lecture et conserver ainsi le principe de compatibilité. Cette double tâche s'effectue généralement en projetant une succession de patrons lumineux, en niveau de gris, ainsi que leur inverse. Le but de cette manoeuvre est d'assurer que les variations d'éclairage le long d'une ligne sont uniques en tout point. Joan Batlle et al. propose une étude du choix du schéma lumineux (voir Salvi *et al.*, 2004), qui est généralement basé sur les codes de Gray ou une onde sinusoïdale.

L'utilisation d'une onde sinusoïdale requiert des capteurs extrêmement précis et un éclairage parfaitement contrôlé. Il faut effectivement être capable de décoder précisément la valeur d'intensité projetée. Les codes de Gray eux nécessitent plus de projections lumineuses, mais sont plus faciles d'utilisation. De plus, ils ont la propriété de peu varier (un seul bit) entre deux valeurs consécutives ce qui permet de minimiser les erreurs.

En utilisant cette technique avec les codes de Gray et un montage stéréo Scharstein et Szeliski ont produit un dataset de couple d'images stéréoscopiques et de bases de vérités qui sert de référence pour la plus part des algorithmes de reconstruction 3D (voir Scharstein et Szeliski, 2003).

Mais il existe d'autres approches utilisant la projection d'une seule texture sur une scène dans un cadre de numérisation stéréoscopique. Le but de ces projections est alors d'ajouter des détails sur la scène photographiée pour essayer de désambiguifié des régions possédant de faibles texturages rendant difficile l'application du principe d'unicité (voir Battle *et al.*, 1998).

Il existe alors de nombreuses stratégies visant à maximiser l'entropie du patron lumineux utilisé dans le but d'assurer une bonne unicité de chaque point de la scène. Zhang et al. proposent par exemple de se contenter de lignes aléatoires (voir Zhang *et al.*, 2004). Kang *et al.* (1995) évaluent quant à eux le gain apporté par l'utilisation de schémas sinusoïdaux. Lim Jongwoo suivit de Pagès et al. propose d'améliorer encore les performances en utilisant des codes de De Bruijn qui ont la particularité de ne pas se répéter, Kurt Konolige pousse l'expérience plus loin en s'assurant d'un maximum de dissimilarité en utilisant un codage de Hamming (voir Lim, 2009; Konolige, 2010; Pagès *et al.*, 2005).

Koschan et al. différent légèrement dans leur approche et proposent de ne pas se contenter de schémas noir et blanc, mais d'utiliser des patrons lumineux colorés. Ils justifient cette approche en démontrant une augmentation de la qualité de près de 20% de la carte de dis-

parités obtenue en utilisant la couleur dans l'algorithme de correspondance par bloc sur des plans rapprochés d'un seul objet (voir Koschan *et al.*, 1996). Cependant, Scharstein et Szeliski réfutent cette amélioration des performances dans un cas plus général (voir Scharstein et Szeliski, 2002).
Zhang *et al.* (2004) introduit également une autre idée importante. La projection de textures sur une scène fait qu'on ne peut pas en capturer le couleur au même instant. Pour pallier à ce problème, ils proposent d'enregistrer à soixante images par secondes, d'utiliser deux sur trois images consécutives pour projeter un schéma lumineux et son inverse et de ne rien projeter sur la troisième afin de capturer la vraie texture de la scène. En utilisant la cohérence temporelle il leur est possible de traquer les correspondances et d'obtenir une carte de disparités et la texture correspondante à 20 images par seconde.
C'est également une stratégie projective qui est utilisée dans le cas de la Kinect de Microsoft. Une lumière infrarouge est ainsi projetée sur un filtre pseudo aléatoire ce qui a pour but de créer un schéma lumineux composé d'un grand nombre de points répartis dans l'espace. Une caméra infrarouge est alors utilisée pour capturer la position de ces points sur la scène. Les déformations subies par le patron sont alors calculées localement et l'on obtient ainsi une disparité entre le projecteur et la caméra infrarouge. En pratique la Kinect donne accès uniquement à la carte de profondeur (voir Kramer *et al.*, 2012). Cette carte peut alors être complété en interpolant entre les points détectés.
La Kinect est à l'origine un périphérique de jeu et, dans cette optique, la qualité n'est pas primordiale. Elle a toutefois très vite été adoptée par le monde de la recherche et il est possible de posttraiter les données, afin d'en améliorer la qualité (voir Izadi *et al.*, 2011a,b; Newcombe *et al.*, 2011).

2.1.3 Méthodes Passives

L'idée la plus intuitive pour construire une carte de disparité, en utilisant un couple de vue stéréo et sans avoir recours à des projections, consiste à comparer les valeurs d'intensité de chacun des pixels des images capturées et chercher pour chaque pixel son correspondant le plus proche dans l'autre image. Toutefois, cette méthode est très instable, en effet elle repose sur la comparaison de valeurs qui peuvent être légèrement bruitées ou qui sont éclairées différemment selon la prise de vue, entrainant ainsi des erreurs. Pour pallier à ce problème, une première stratégie consiste à considérer un bloc autour du pixel et à comparer une région et donc un ensemble de valeurs (voir Koschan, 1993). Les valeurs d'intensité de deux blocs sont alors généralement comparées en utilisant la somme des différences au carré. Précédemment, la somme des valeurs absolues de ces différences était utilisée pour des raisons de rapidités,

mais il a été établi que les résultats obtenus sont moins précis (voir Scharstein et Szeliski, 2002).

Cependant, l'utilisation de cette métrique simple ne résout le problème qu'en utilisant plus de données et en comptant sur un moyennage des erreurs pour stabiliser les résultats. Ansar *et al.* (2004) ont donc eu l'idée de ne pas se contenter de cette comparaison, mais tout d'abord de filtrer l'image afin d'en réduire les bruits et d'augmenter la stabilité de la comparaison. Pour se faire, ils utilisent un filtre paramétrique appelé filtre bilatéral qui garantit de conserver les détails importants de l'image tout en floutant les zones homogènes ce qui a pour effet de supprimer les bruits propres aux capteurs (voir Ansar *et al.*, 2004).

Zabih et Woodfill (1994) poussent l'idée d'utiliser une version filtrée de l'image encore plus loin en utilisant un filtre non-paramétrique et plus proche d'un gradient local. Ils proposent deux approches, une version simple appelée transformation du rang et une version améliorée appelée census. L'idée consiste à construire un descripteur local pour chaque pixel qui sera robuste aux bruits et aux changements d'illuminations liés aux déplacements spatiaux de chaque caméra. Pour obtenir un tel descripteur, leur première idée consiste à utiliser une fenêtre autour du pixel d'intérêt et à comparer l'intensité de chacun des pixels de cette fenêtre avec la valeur centrale (cette valeur centrale est généralement exclue de la comparaison, mais n'impacte pas sur les performances). Le descripteur du rang est alors construit en comptant le nombre de pixels dont la valeur est inférieure à la valeur centrale. Chaque pixel étant remplacé par son descripteur et l'on peut alors appliquer l'algorithme de correspondances par blocs.

Cette transformation améliore déjà significativement la qualité de la carte de disparités crée, mais Zabih et Woodfill (1994) proposent également une version optimisée du filtre en rajoutant une information de position au descripteur local. Pour se faire, au lieu de se contenter du nombre de pixels d'intensité inférieure, ils encodent également leur position dans un tableau de bits. Pour chaque position dans la fenêtre autour d'un pixel d'intérêt, soit l'intensité est inférieure à celle considérée et on enregistre un un à la postion correspondante dans notre descripteur, soit elle est suppérieure ou égale et on enregistre dans ce cas un zéro. Le descripteur ainsi créé contient à la fois l'information de rang, mais également celle de position (voir Zabih et Woodfill, 1994). Cette approche donne d'excellents résultats et est à l'heure actuelle à la base de beaucoup d'autres approches, dont l'état de l'art en terme de création de carte de disparités (voir Scharstein et Szeliski, 2002).

Le principal défaut de cette technique est qu'elle est plus couteuse en temps. En effet, il faut dans un premier temps appliquer une transformation non-paramétrique à l'ensemble de l'image puis utiliser une distance de Hamming (plus couteuse que la somme des carrés) sur chacun des descripteurs. Il existe toutefois des propositions d'implémentations optimisées

pour GPGPU atteignant des temps de calcul tout à fait raisonnables (voir Weber *et al.*, 2009).
Un autre axe de recherche, pour l'amélioration des cartes de disparités, se rapproche de cette technique et consiste à trouver une solution pour modéliser la cohérence spatiale. En effet, le second principe de Marr assure que la disparité doit varier de façon continue sur une même surface. Or rien n'impose cette continuité dans les méthodes présentées précédemment. Hirschmuller (2008) propose d'utiliser la programmation dynamique pour faire une agrégation des coûts de disparités. L'idée est alors non plus de minimiser la métrique usuelle, mais de minimiser une fonction d'énergie définie sur un voisinage de pixels et un ensemble de disparité (voir Hirschmuller, 2008).
Zhang et al. proposent une approche combinant la correspondance par blocs avec une somme de valeur absolue, le census, et l'agrégation de coûts. L'idée derrière cette approche est que le census apporte une robustesse intéressante, mais perd toute information d'intensité. Cette combinaison donne à l'heure actuelle les meilleurs résultats sur le dataset Middlebury (voir Mei *et al.*, 2011).
Enfin, il existe des méthodes hybrides qui utilisent à la fois des approches clairsemées pour obtenir une idée générale et complète la carte de disparités ainsi produite à l'aide d'une technique dense. C'est le cas par exemple de Humenberger *et al.* (2010) qui établi dans un premier temps des correspondances de superpixel avant de raffiner en calculant la disparité propre à chaque pixel. Yin *et al.* (2010) proposent dans la même optique d'utiliser de multiple résolutions et d'utiliser des captures à basse résolution pour restreindre les zones de recherches et les calculs sur les images haute résolution.

2.2 Méthodes probabilistes

Parmi les techniques précédemment énumérées, un grand nombre ont recours à la programmation dynamique soit pour améliorer la qualité en traitement a priori des coûts de dissimilitude soit dans le but d'homogénéiser les disparités a posteriori. Le problème de telles approches réside dans les contraintes qu'elles imposent et qui doivent être préétablies et formulées de façon fixe dans leurs implémentations. Or les contraintes nécessaires à une formulation en programmation dynamique d'un problème de disparité pour de la macrophotographie ne sont surement pas les mêmes que pour faire de la stéréo en extérieur et reconstruire des villes ou pour permettre la navigation autonome d'un robot sur mars.
Cependant, le principe de base derrière ces stratégies consiste à transmettre des messages le long de chemins au travers notre carte de disparités. Ce chemin peut être plus ou moins

complèxe, allant d'une simple ligne horizontale à un modèle en étoile qui passerait par chacun des points d'intérêts(voir Hirschmuller, 2008). Or c'est précisément ce que cherchent à modéliser, de façon probabiliste, les réseaux bayésiens(voir Jensen, 1993). Le fait d'introduire un modèle probabiliste plutôt que d'utiliser des contraintes fixes, apporte une plus grande robustesse aux erreurs ainsi qu'une plus grande adaptabilité pour changer le modèle d'échelle sans changer tout l'algorithme utilisé.

Les réseaux bayésiens permettent en effet aussi bien de représenter un modèle de connaissances que de calculer des probabilités conditionnelles(voir Russell et Norvig, 2003). Or dans notre cas, nous souhaitons modéliser la contrainte de continuité de Marr, cette contrainte impose que si le voisinage d'un point est à une certaine disparité alors ce point doit probablement être également à cette disparité, ou en est proche. Un réseau bayésien correctement paramétré nous permettra alors de relier nos connaissances sur un voisinage à la probabilité qu'un point soit à une certaine disparité.

Les réseaux de croyance, autrement appelé réseaux de propagation de la croyance (Belief Network et Belief Propagation Network) permettent de pousser ce raisonnement encore plus loin (voir Ripley, 1996, ch8, p246-285). Au lieu de considérer un petit réseau sur le voisinage d'un point, on considère alors tout un ensemble de points s'avoisinant les uns les autres. On va ensuite chercher à propager des messages, croyances, dans tous le réseau afin d'aboutir a un état qui maximisera la probabilité marginale de l'ensemble de nos points.

Parmi ces réseaux on distingue notamment les modèles de Markov cachés (MMC) qui sont surement la forme la plus courante et la plus utilisée lorsque l'on souhaite faire du traitement du signal. Ces modèles considèrent qu'un noeud de notre réseau ne dépend que du noeud qui le précède. Mais surtout les MMC introduisent une notion d'observations et de variables cachées (voir Bishop, 2006). Comme illustré en Figure 4.8 un MMC se décompose en effet en deux parties. D'une part les variables représentant les observations, ce que l'on a pu décoder, une disparité a priori, une version bruitée de l'information qui nous intéresse vraiment. D'autre part les variables cachées qui représentent les vraies valeurs prises par nos variables, l'information non bruitée qui nous intéresse. Les variables cachées engendrent donc les variables observées, mais elles sont également dépendantes de celles qui les précèdent. On peut donc établir un sens pour notre réseau : de la première variable à la dernière. Raisonner sur un tel modèle consiste donc à trouver l'état de chacune des variables cachées qui maximise à la fois la probabilité d'émission : la probabilité que les observations découlent effectivement des variables cachées, et la probabilité de transition : la probabilité que chaque variable survienne après celle qui la précède et avant celle qui la suit en terme de valeur.

Il est aisé de voir la puissance et l'utilité d'un tel modèle pour le débruitage de données dans divers domaines. Il existe donc de nombreux algorithmes pour traiter les MMC. L'algorithme

le plus utilisé et le plus connu pour traiter les MMC est sûrement l'algorithme de Viterbi (1967), originellement créé pour les télécommunications et la robustesse aux erreurs. Cet algorithme est à mi-chemin entre le raisonnement probabiliste et la programmation dynamique. En effet, le principe consiste à propager des valeurs dans notre réseau, à calculer leurs probabilités et, une fois le réseau entièrement parcouru, à revenir sur nos pas en prenant le chemin menant au maximum de probabilité (back-track).

Cet algorithme est, en fait, une version plus rapide et légèrement moins précise de l'algorithme Forward-Backward (voir Bishop, 2006). L'algorithme Forward-Backward consiste lui en deux passages au lieu d'un : des valeurs sont propagées à la fois dans le sens du réseau et dans le sens opposé au réseau. On calcule alors des probabilités partielles pour chacune de ces propagations, et, une fois toutes les propagations passant par un noeud sont effectuées, on peut alors combiner ces valeurs partielles pour obtenir la probabilité de chacune des valeurs que peut prendre le noeud. On peut alors maximiser la probabilité de chaque état et non plus la probabilité d'une suite de valeurs dans le réseau. Un autre avantage de cette approche vient du fait qu'on a, à la fin, la densité de probabilité de chaque état possible pour chacun des noeuds, ce qui peut servir dans un traitement a posteriori des données.

Les MMC sont donc de très bons outils pour certaines modélisations, mais ils nécessitent une dépendance restreinte des noeuds entre eux. On peut facilement en imaginer une utilisation pour le calcul de disparités et la vision stéréoscopique si l'on se contente de lignes dans notre carte de disparités (voir Schaeffer et Parr, 2006). Mais beaucoup de stratégies de programmation dynamique améliorent les résultats, car elles considèrent la cohérence de la scène dans plus d'une direction, Hirschmuller (2008) par exemple considère une étoile à seize branches autour de chaque point.

Une première idée pour augmenter la dimension de notre continuité peut être d'utiliser consécutivement plusieurs MMC. Ma (2012) par exemple utilise un premier MMC sur les lignes horizontales suivi d'un MMC vertical. Cette approche nécessite cependant de grandes précautions afin d'éviter le surapprentissage.

Il existe également d'autres types de modèles tels que les champs aléatoires conditionnels (Conditional Random Field, CAM) qui permettent de modéliser des dépendances plus importantes en quadrillant notre carte de disparités. De tels modèles sont généralement utilisés pour de la segmentation d'image (voir Bhole *et al.*, 2011), mais aussi pour du débruitage et de l'amélioration de carte de profondeur (voir Weinman *et al.*, 2008). Les CAM, de par leur grand nombre de connexions et leur complexité, donnent d'excellents résultats, mais requièrent des temps de calcul extrêmement importants. Leur utilisation dans un système en temps réel n'est, par exemple, pas envisageable.

Mais si l'on revient à la définition, un réseau bayésien est simplement un graphe dans lequel

chaque noeud représente une variable aléatoire et chaque arrête une influence conditionnelle entre deux variables (voir Jensen, 1993). On peut donc modéliser n'importe quelle liaison entre différentes positions dans notre carte de disparité à l'aide d'un réseau bayésien. Le modèle en étoile à seize branches qu'utilise Hirschmuller (2008) pour faire de la programmation dynamique peut donc être vu comme un réseau bayésien sur lequel on peut appliquer des inférences probabilistes plutôt que des règles logiques arbitraires.
Il existe de nombreuses règles et formules mathématiques permettant de transformer un réseau bayésien quelconque en graphe des intersections afin de faciliter les calculs d'inférences (Jensen, 1993, offre une très bonne introduction en la matière). Nous développerons plus loin (chapitre 5) les calculs nécessaires à notre cas particulier.

2.3 Autres méthodes d'acquisition tridimensionnelle

Au cours de la phase d'exploration de cette maîtrise, diverses autres approches ont été rencontrées, implémentées et testées. Nous avons choisi de ne finalement pas les utiliser dans notre approche finale parce qu'elle ne s'applique pas suffisamment bien au problème considéré, que ce soit pour des raisons de temps de calcul ou de qualité des résultats. Nous jugeons tout de même qu'il est intéressant de prendre connaissance de ces techniques que ce soit pour comprendre le cheminement ayant abouti à ce mémoire, par simple culture ou pour envisager des développements futurs aux travaux présentés ici.
Une des premières approches considérées fut la reconstruction de forme à partir des ombres (Shape From Shading). Cette stratégie requiert souvent une connaissance et un parfait contrôle de la position des sources lumineuses ainsi que des paramètres et du type de la caméra (voir Prados et Faugeras, 2005). Une alternative consiste à utiliser un plus grand nombre de sources lumineuses et une décomposition en éléments simples (voir Schindler, 2010; Basri et Jacobs, 2001). Le principe de base derrière cette technique reste le même et consiste à déterminer une carte de normales, liées aux dérivées partielles et à les utiliser pour résoudre un problème de Gauss-Seidel. Toutefois, il a été démontré (voir Prados et Faugeras, 2005) que ce problème est mal posé, il existe donc plusieurs solutions pour une même acquisition. Il faut par ailleurs apporter quelques modifications à cette technique si l'on s'intéresse à des surfaces non régulières ou discontinues (voir Camilli et Prados, 2006).
Ce problème étant mal posé il est nécessaire d'avoir un très bon a priori pour converger vers la véritable solution. Du *et al.* (2011) propose de résoudre ce problème en utilisant d'une part la programmation linéaire et une approximation de convexité pour simplifier le problème, mais surtout de combiner la reconstruction à partir des ombres à de la stéréo classique par

correspondance par bloc. Ils obtiennent ainsi une bonne carte de disparités qui est ensuite raffinée par la résolution d'un système différentiel.

L'utilisation d'un numériseur laser a également été envisagée pour construire des cartes de vérité servant soit à la comparaison et à l'évaluation de nos calculs soit à l'entrainement de modèles probabilistes. Biber *et al.* (2004) constitue une bonne introduction à la technique basée sur le temps de vol de la lumière laser. Pour des détails plus avancés ainsi qu'un exemple de solution commerciale nous recommandons la documentation du scanner Leica LensStation C10 (voir Leica, 2012).

Enfin il est important de mentionner des approches clairsemées telles que l'ajustement de faisceaux (voir Triggs *et al.*, 2000; Agarwal *et al.*, 2009). Ces approches, qui utilisent la détection et la mise en correspondance de points clés (voir Lowe, 2004) pour établir le positionnement de prise de vue, ne construisent, à proprement parlé, pas de carte de disparités. Mais il est envisageable de combiner différentes captures stéréoscopiques d'une même scène en les repositionnant dans le même espace grâce à cette stratégie.

CHAPITRE 3

MÉTHODES ACTIVES

3.1 Introduction

Nous appelons carte de vérité une carte de disparités pour laquelle chacune des valeurs de disparité est mesurée avec exactitude. Comme mentionné précédemment, il est important d'être en mesure de produire de telles cartes pour les scènes que nous souhaitons numériser. Que cela soit à des fins d'apprentissage pour utiliser des modèles probabilistes sur des scènes similaires, ou simplement pour évaluer la qualité de nos algorithmes. Pour des raisons de calibrations et de précisions, une approche de stéréoscopie active a été envisagée. Ce chapitre couvre diverses stratégies qui ont été étudiées pour aboutir à l'obtention d'une carte de vérité ainsi que l'étude de divers schémas exploitables pour faire de la stéréoscopie active.

3.2 Lumière structurée

La lumière structurée est un des moyens les plus précis à l'heure actuelle d'obtenir une carte de disparités d'excellente qualité (voir Scharstein et Szeliski, 2003; Battle *et al.*, 1998; Lanman et Taubin, 2009). Le principe de base derrière cette technique consiste à attribuer un unique identifiant à chaque point de la scène que l'on souhaite numériser puis à retrouver les correspondances non pas sur les intensités lumineuses capturées, mais directement sur les identifiants.

Il existe de nombreuses variantes à cette approche, les schémas lumineux à projeter par exemple peuvent être choisis selon plusieurs stratégies pour aboutir à un résultat unique ou suffisamment unique dans un voisinage pour être identifiables. Il est également possible d'utiliser cette approche en utilisant une unique caméra et un projecteur ou un couple de caméras en stéréo en plus du projecteur.

Il existe des solutions commerciales qui utilisent cette approche afin de construire un numériseur 3D, nous avons notamment expérimenté la solution proposée par 3D3 Solution (FlexScan3D). Cependant, nous désirons avoir un très grand contrôle sur ces numérisations, notamment nous désirons avoir une carte de disparités et non pas une carte de profondeurs ou un nuage de point. Il nous faut de plus un repère connu pour pouvoir comparer à d'autres solutions ou encore la possibilité de numériser et, dans la foulé, de faire d'autres captures afin d'éviter de perdre la calibration ou d'avoir du mouvement dans la scène.

Pour toutes ces raisons, nous avons choisi d'implémenter notre propre version de numériseur par lumière structurée. Dans cette optique nous avons donc dû faire un certain nombre de choix quant à la stratégie finale que nous souhaitons utiliser.

3.2.1 Choix d'implémentation

Comme mentionné précédemment, cette technique reposant sur la projection de schémas lumineux, il est tout à fait possible d'envisager de calibrer une caméra avec un projecteur et de n'utiliser que ces deux appareils pour numériser. La caméra est alors utilisée pour mesurer les déformations des schémas projetés qui sont directement liées à la géométrie de la scène. On peut donc calculer une carte de disparités non pas entre deux vues de la scène ou entre la position de points géométriques, mais directement entre le patron projetté (non-déformé) et la vue capturée par la caméra (l'image résultante). Cette technique présente l'avantage de n'utiliser qu'une caméra ce qui réduit la présence de bruits liés à l'acquisition. Mais elle requiert une calibration extrêmement précise, une très bonne résolution aussi bien pour la capture que pour la projection et une synchronisation précise du projecteur et de la caméra. L'autre possibilité concernant l'acquisition consiste à utiliser un couple de caméras en stéréo. La succession de schémas lumineux a alors pour but de rendre chaque pixel des captures unique. On peut ensuite utiliser une approche classique de correspondance par bloc afin de construire une carte de disparités. Un intérêt non négligeable de l'utilisation de deux caméras est qu'après calibration, il est alors possible d'aligner et de rectifier les prises de vue afin que chaque ligne corresponde à la même ligne de l'autre vue. Les schémas lumineux projetés n'ont alors plus qu'à rendre unique chaque position dans une ligne et l'on peut construire notre carte de disparités en balayant simplement chacune des lignes dans nos multiples images. Cette solution apporte de plus l'avantage d'utiliser un couple de caméras en stéréo qu'il est également possible d'utiliser pour le reste de nos expériences. Enfin, la carte de disparités produite est directement dans le repère de nos caméras elle est donc très facilement exploitable dans le cadre de nos expériences. C'est donc assez naturellement que nous avons choisi d'avoir recours à cette stratégie.

L'autre choix d'implémentation à faire dans le cadre d'une numérisation par lumière structurée concerne les schémas lumineux que l'on souhaite projeter. La section suivante (3.3) propose une exploration sur l'utilisation de la couleur, toutefois les schémas lumineux utilisés sont généralement binaire, transcrit en noir et blanc. Ceci se justifie, entre autres, par le fait que la stéréoscopie ne gagne pas toujours dans l'utilisation de la couleur et bon nombre d'algorithmes (le census notamment) utilisent des niveaux d'intensités et non pas des couleurs en entrées. Mais surtout par l'importante différence qui existe entre l'absence de projection

(noir) et la projection de toutes les couleurs superposées (blanc). Non seulement de tels schémas jouent exclusivement sur l'intensité, mais en plus ils sont robustes à la couleur de la scène : éclairer un objet bleu avec une lumière rouge ou ne pas l'éclairer donne des résultats très proches.
Les deux stratégies principales consistent à utiliser une série de signaux sinusoïdaux ou une série de codes de Gray. Nous avons choisi d'utiliser les codes de Gray qui nécessitent plus de projections, mais qui sont plus stables et robustes à la présence d'un éclairage ambiant. Les codes de Gray, aussi appelés binaire réfléchi, sont des codes binaires qui ne changent que d'un bit entre deux codes consécutifs, le Tableau 3.1 illustre les huit premiers codes binaires ainsi créés. Cette propriété assure qu'une erreur lors du décodage aura un impact restreint. Mais surtout, elle évite d'avoir à éclairer ou ombrager de trop petites zones (un unique pixel projeté ne serait pas forcément détecté), en effet on est assuré qu'un bit n'est jamais fixé à un ou zéro pour être remis à zéro, respectivement un, sur le code suivant.

Tableau 3.1 Les huit premiers codes de Gray

	Binaire	Code de Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

3.2.2 Algorithme

Une fois ces choix d'implémentation faits, l'algorithme en lui-même ne présente pas de grande difficulté. Une série de schémas lumineux (voir Figure 3.1) est projetée. Chaque schéma code un bit du code de Gray. Afin de pouvoir déterminer si l'on projette ou non de la lumière, chaque schéma est immédiatement suivi de son inverse. Le décodage consiste donc à déterminer si l'intensité lumineuse est plus importante quand on projette le patron lumineux ou lorsqu'on projette son inverse. La concaténation de chacun de ces bits nous permet d'identifier de façon unique chacun des pixels le long d'une ligne de numérisation.
Chaque série de captures peut alors être transformée en une matrice d'identifiants. Comme nous avons un montage en stéréo calibré (voir Figure 3.2) nous obtenons deux matrices cor-

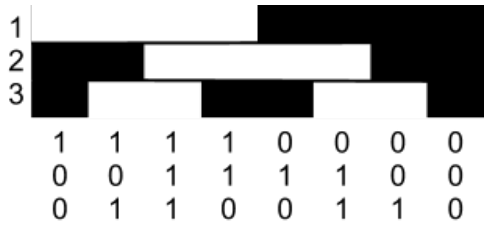

Figure 3.1 Schémas projetés au fil du temps et décodage.

respondantes et l'on peut alors chercher les disparités sur chacune des lignes.

Figure 3.2 Montage comprenant un projecteur, une caméra stéréoscopique pour la numérisation par lumière structurée et une kinect pour de futures expériences.

3.2.3 Résultats

Cette section présente plusieurs résultats obtenus par cette méthode, en ayant recours à dix-huit schémas lumineux, à titre informatif. On présente également un rendu coloré des codes de Gray décodés sur chacun des deux exemples, un visage mannequin Figure 3.3 et un vrais visage Figure 3.4.

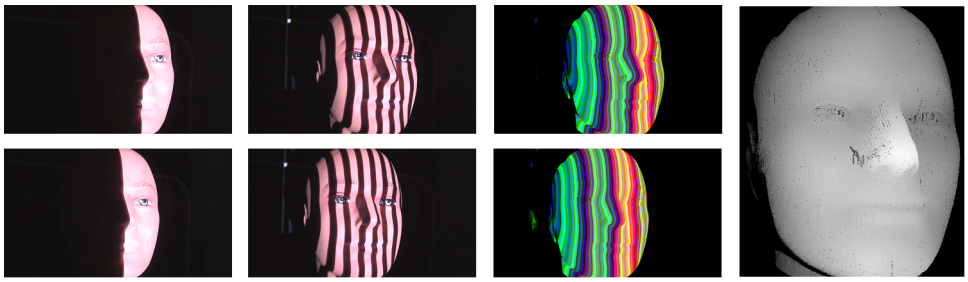

Figure 3.3 Résultats obtenus sur une tête de mannequin. Vue de gauche en haut et de droite en bas. Avec, de gauche à droite, la capture de deux patrons lumineux, l'illustration des code Gray décodés, et la carte de disparités produite.

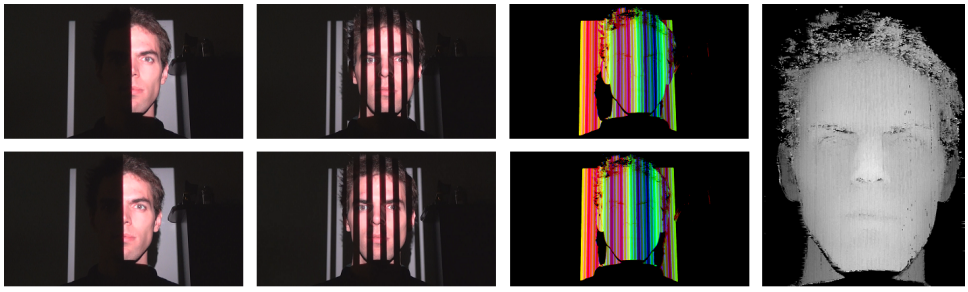

Figure 3.4 Résultats obtenus sur un vrai visage. Vue de gauche en haut et de droite en bas. Avec, de gauche à droite, la capture de deux patrons lumineux, l'illustration des code Gray décodés, et la carte de disparités produite.

3.3 Série colorée

3.3.1 Utilisation de six schémas

La méthode précédente nécessite la projection de dix-huit schémas lumineux noir et blanc. Pour essayer de réduire le temps nécessaire à la capture et la construction de la carte de vérité, nous avons envisagé la possibilité d'utiliser des couleurs pour combiner ces schémas. Notre première stratégie a été de superposer les schémas dans un espace de couleur différent, le premier schéma noir et blanc devient noir et rouge, le second utilise le vert et le troisième le bleu. On peut alors projeter tous ces trois schémas en une seule fois et réduire le nombre de projections par trois. La figure 3.5 illustre la superposition de nos trois premiers schémas.

Cette stratégie réduit le temps nécessaire pour l'acquisition, elle présente toutefois le risque de ne pas bien répondre pour certaines couleurs. Comme mentionné précédemment, la couleur d'un objet est définie par la longueur d'onde qu'il peut réfléchir. Si on ne projette pas cette longueur d'onde sur un objet, il sera alors parfaitement noir. Les objets d'une couleur parfaite peuvent alors être problématiques et dans la pratique nous avons observé une perte de qualité

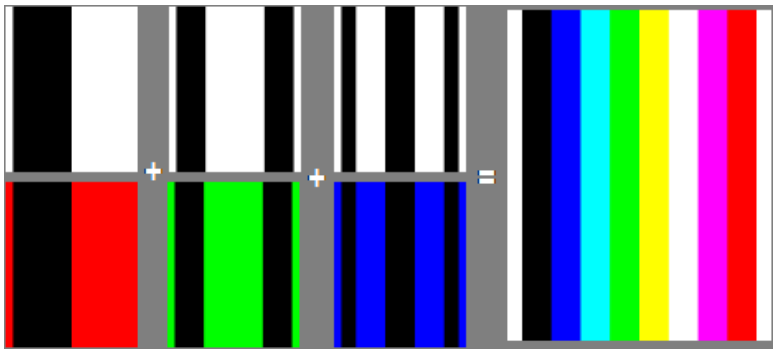

Figure 3.5 Schémas noir et blanc convertis en couleur et superposés.

de l'ordre de 1.023% en moyenne, en utilisant cette stratégie plutôt que la première, sur une dizaine de mesures de poses diverses. Nous mesurons l'erreur moyenne en comptant le nombre de disparités incorrectes dans notre carte. La figure 3.6 illustre les défauts apparaissant et les artefacts que l'approche engendre comparée à la figure 3.4.

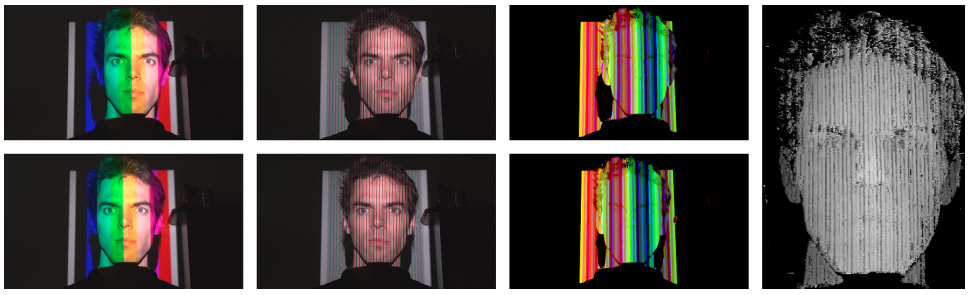

Figure 3.6 Résultats obtenus en utilisant une version colorée de nos schémas. Vue de gauche en haut et de droite en bas. Avec, de gauche à droite, la capture de deux patrons lumineux, l'illustration des code Gray décodés, et la carte de disparités produite.

3.3.2 Utilisation de deux schémas

Toujours dans le but de réduire le temps nécessaire à la construction d'une carte de vérité nous avons ensuite envisagé diverses stratégies pour ne projeter qu'un seul schéma et son opposé pour essayer de le décoder. Le principe de fonctionnement de l'albédo et de la rémission des couleurs, ainsi que la précision des capteurs photographiques font que cette approche est assez peu réaliste. En effet, cette stratégie est plus proche de la simple stéréoscopie active que d'une approche par lumière structurée.

Somme des codes de Gray

En poursuivant notre expérimentation sur les codes de Gray nous avons eu l'idée de transformer nos six schémas précédents en les sommant, en séparant chacun des canaux et en normalisant à la fin. Cette représentation n'est pas des plus judicieuses, en effet les codes 110, 101 et 011 par exemple donnent tous une somme de deux. Cependant dans un voisinage restreint, on obtient une assez bonne unicité et les résultats obtenus sont assez encourageants (voir Tableau 3.2). La Figure 3.8 présente les schémas ainsi construits (la somme et la somme des complémentaires).

Barycentre des codes de Gray

Pour résoudre le problème d'unicité le long d'une scanline, nous avons alors envisagé de remplacer notre simple somme par un barycentre assurant l'unicité. Chaque couleur de nos schémas est codée sur 256 valeurs possibles. Nous voulions nous assurer d'utiliser le plus possible de ces valeurs afin de maximiser le contraste et d'être en mesure de le percevoir une fois projeté. Cette stratégie peut être vue comme la conversion de nos bits en base deux à un espace étiré par un facteur α. Cette idée se traduit mathématiquement par l'équation $\alpha + 2\alpha + 4\alpha = 255$ on obtient alors $\alpha = \frac{255}{7}$. Afin de bien différentier cette approche de la précédente, il est possible de considérer les schémas produits comme le résultat d'une somme pondérée des schémas colorés de la stratégie précédente (voir figure 3.7). Dans la pratique, l'unicité apportée par cette stratégie n'est pas primordiale et l'amélioration en terme de qualité de carte de disparités n'est pas si importante (voir Tableau 3.2). La Figure 3.8 présente les schémas ainsi construits (le barycentre et son complémentaire).

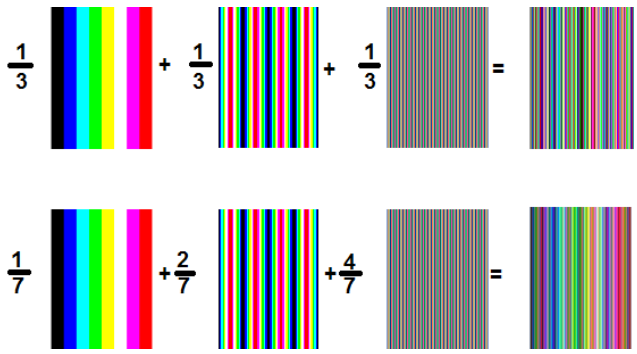

Figure 3.7 Construction des schémas résultants des codes de Gray soit par simple somme (en haut) soit en utilisant des barycentres pour garantir l'unicité (en bas).

Schéma aléatoire

Une dernière idée, plus utilisée dans une perspective de technique de base, consiste à utiliser une texture aléatoire et son opposée afin d'essayer de retrouver les déformations subies par cette texture plutôt que de faire de la simple correspondance par bloc. Les résultats obtenus sont également reportés dans le Tableau 3.2 et les textures utilisées sont données à titre d'exemple dans la Figure 3.8.

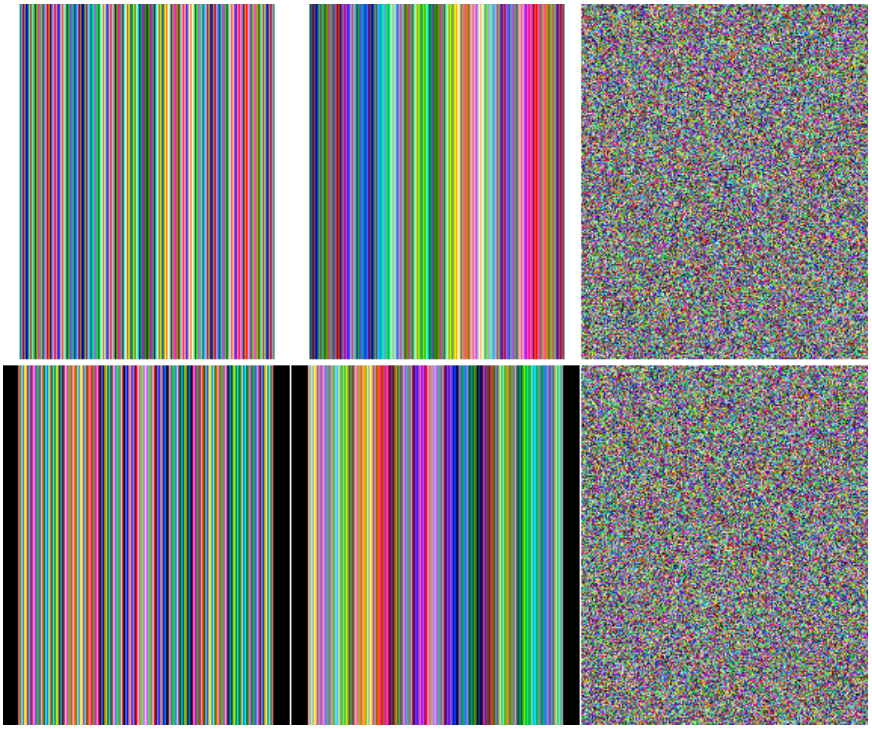

Figure 3.8 Schéma utilisé dans l'approche utilisant seulement deux patrons (schéma en haut et inverse en bas). À gauche la somme des codes de Gray, au centre la version barycentrique, à droite la version aléatoire.

3.3.3 Algorithme

L'algorithme utilisé, que ce soit avec six schémas ou avec deux, varie très peu de celui utilisé dans le cadre des schémas noir et blanc. Chaque schéma est projeté suivi de son inverse. On cherche ensuite à déterminer le code de gray correspondant à chaque pixel en comparant le niveau de luminosité de chacun des canaux rouge vert et bleu des deux images ainsi capturées. Une fois la version bruitée des schémas obtenue, on procède à la même recherche de correspondances que dans le cadre des patrons binaires.

3.3.4 Résultats

Il existe plusieurs métriques pour évaluer la qualité d'une carte de profondeurs. Il est notamment possible de comparer l'erreur moyenne sur l'ensemble des points considérés ou le nombre de ces derniers mal repositionnés à un certain seuil près. La mesure la plus communément utilisée est l'erreur de reconstruction à une unité près : on comptabilise alors le nombre de fois ou la disparité obtenue diffère de la vraie disparité de plus d'un, on converti ensuite ce compte en un pourcentage d'erreur.
Nous avons mesuré les erreurs en utilisant l'erreur moyenne (moyenne de l'erreur sur toute la carte de disparité) ainsi que l'erreur de reconstruction à un près sur une dizaine de captures réalisées dans les mêmes conditions. Nous présentons ici (voir Tableau 3.2) l'erreur moyenne sur l'ensemble de ces captures. L'ensemble de ces résultats met en évidence un léger gain de qualité obtenu par l'utilisation de la somme pondérée des codes de Gray. Par ailleurs, la faible dégradation des mesures ainsi produites est encourageante dans la mesure où l'on est capable de construire un très bon a priori de notre scène, pouvant par exemple nous servir pour des méthodes d'apprentissage, en ne projetant que deux schémas consécutivement.
Ces résultats confirment également l'importance de l'unicité, la somme brute des codes de Gray donne un moins bon résultat, car elle ne garantit pas l'unicité de chacun des points au sein d'une même ligne. De même, on constate que, conformément à ce qui est mentionné dans la littérature, l'entropie et la non-prédictibilité du schéma projeté donnent de bonnes performances en moyenne. En effet, le schéma aléatoire donne des résultats très proches de ceux obtenus en utilisant une pondération de code de Gray.

Tableau 3.2 Comparaison des méthodes utilisant une approche colorée de la lumière structurée

Méthode	Erreur moyenne (disp.)	Erreur à un pixel (%)
six schémas	0.13	1.156
Somme Gray	0.76	3.909
Barycentre Gray	0.61	3.711
Aléatoire	0.60	4.132

3.4 Stéréoscopie active

La stéréoscopie active consiste à utiliser la projection de texture sur une scène d'intérêt afin de pallier au problème dit de "défaut de texture". En effet, l'établissement de correspondance dans une zone fortement texturée, sans répétition, ne poserait pas de problème. Mais lorsque l'on est confronté à des régions où l'on ne distingue qu'une couleur unie ou une tex-

ture répétitive, il devient plus compliqué de calculer notre carte de disparités. Les approches présentées précédemment peuvent être considérées comme excessives, elles nécessitent l'emploi de plusieurs schémas lumineux projetés sur une scène qui doit être statique pendant les projections. La stéréoscopie active se contente d'une seule projection à la fois pour venir enrichir les textures déjà présentes dans la scène numérisée. Au lieu de chercher à décoder un schéma lumineux déformé pour établir des correspondances, on calcule directement ces dernières sur les captures de la scène éclairée.
Dans la pratique, cette stratégie offre un très bon point de référence : la carte de disparité obtenue est de bonne qualité. Mais le rajout de texture n'est cependant pas suffisant pour obtenir une carte équivalente à celle de vérité.
Cette approche repose sur les mêmes algorithmes de mise en correspondance que pour la stéréoscopie passive. Ces techniques seront plus amplement développées dans le chapitre suivant. Une fois encore l'utilisation de différents schémas est envisageable. En nous basant sur la littérature et en partant du principe que cette stratégie n'est ici qu'un point de repère, nous avons choisi de nous contenter d'un schéma aléatoire et d'une méthode de correspondance par bloc.

3.4.1 Résultats

Les résultats obtenus avec cette approche sont de loin moins bons que ceux obtenus en utilisant des projections multiples. Toutefois, il est intéressant de les comparer aux résultats obtenus sur la même scène sans aucune projection (voir Tableau 3.3). Le problème de "défaut de texture" est effectivement grandement réduit.

Tableau 3.3 Comparaison entre stéréoscopies active et passive

Méthode	Erreur moyenne (disp.)	Erreur à un pixel (%)
Aléatoire	1.4	17.03
Barycentre Gray	1.4	15.19
Passive	3.0	36.00

3.5 Contraintes temporelles et visuelles

Comme indiqué dans l'introduction de ce mémoire, la motivation principale est d'avoir la meilleure numérisation possible en un temps le plus court possible. Les techniques et résultats présentés ici peuvent sembler être une bonne piste. Cependant, certaines contraintes ont été ignorées. Dans un premier lieu, la lumière structurée ne peut pas être réalisée en

temps réel sans un équipement hautement spécialisé. De plus, projeter des schémas lumineux impacte la numérisation et les vraies couleurs de la scène. Il pourrait être envisagé d'utiliser des infrarouges pour résoudre ce problème, mais la qualité de ce genre d'approches n'est pas optimale. La figure 3.9 résume l'ensemble des approches présentées dans ce chapitre et illustre la qualité de chacune ainsi que le temps nécessaire à chaque cas.
L'idée est donc dans un premier temps d'explorer différentes approches passives et d'éventuellement utiliser des approches actives avec parcimonie pour évaluer nos résultats ainsi que pour les améliorer en utilisant des stratégies d'apprentissage.

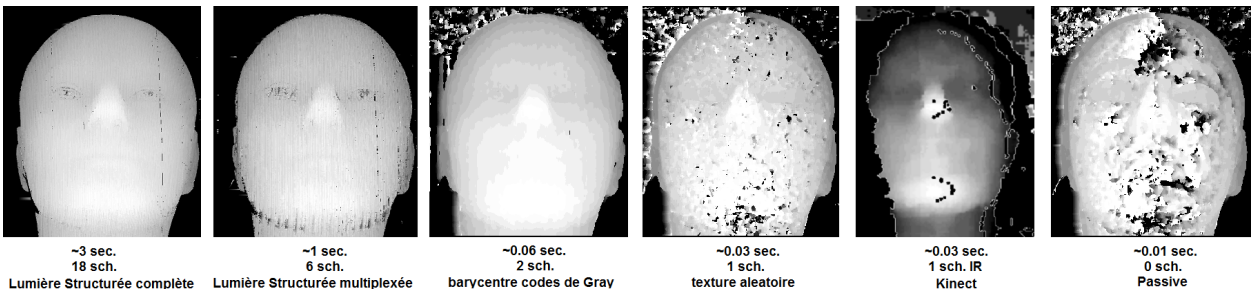

Figure 3.9 Comparaison empirique des temps d'acquisition et de la qualité des cartes de disparités produites par les différentes méthodes présentées.

CHAPITRE 4

MÉTHODES PASSIVES

4.1 Introduction

Cette section présente les algorithmes de base nécessaire dans notre modèle final. Elle se place directement dans le prolongement de travaux précédents entrepris par Ma (2012). Nous présenterons donc sommairement les algorithmes et nous reviendrons plus en détail sur leurs implémentations en GPGPU et sur les optimisations qui ont été apportées afin de réduire les temps de calcul. Toutefois, nous invitons le lecteur à se référer aux publications sur les travaux précédents pour plus de détails (voir Ma, 2012).

4.2 Correspondance par bloc

Il s'agit surement de l'algorithme le plus intuitif. Il consiste à découper une fenêtre de taille arbitraire autour de chaque pixel et d'essayer de retrouver cette fenêtre dans la capture dans seconde vue (la Figure 4.1 illustre ce principe). Cette stratégie nécessite cependant de spécifier plusieurs paramètres. Tout d'abord, la taille de la fenêtre considérée impacte fortement le résultat, une fenêtre trop petite entrainera un grand nombre de fausses correspondances avec le cas extrême ou on ne cherche qu'un unique pixel, mais une fenêtre trop grande entrainera également un lissage important qui est indésirable et répondra encore moins bien aux zones d'occlusions. Ce problème est solvable soit par la recherche de la meilleure taille de fenêtre soit par l'utilisation de multiple résolution, que ce soit pour les tailles ou directement les prises de vues (Yin *et al.*, 2010). L'autre paramètre important est le choix de la métrique utilisée pour calculer la similarité. Les deux plus utilisées son généralement les distances associées aux normes L_1 et L_2.

$$d_1(R, C) = \sum_{i \in F} |R(i) - C(i)|$$
$$d_2(R, C) = \sum_{i \in F} (R(i) - C(i))^2$$

Avec R la fenêtre de référence (autour du pixel dont on cherche la disparité), C celle à laquelle on la compare, et F l'ensemble des indices selon la taille des blocs. Ces distances sont aussi appelées distance du chauffeur de taxi (city block) et distance euclidienne. Mais il est possible, bien que plus complexe d'utiliser une métrique dépendant de la distance au

centre de la fenêtre pour pondérer l'importance de chaque pixel (type earth mover distance).

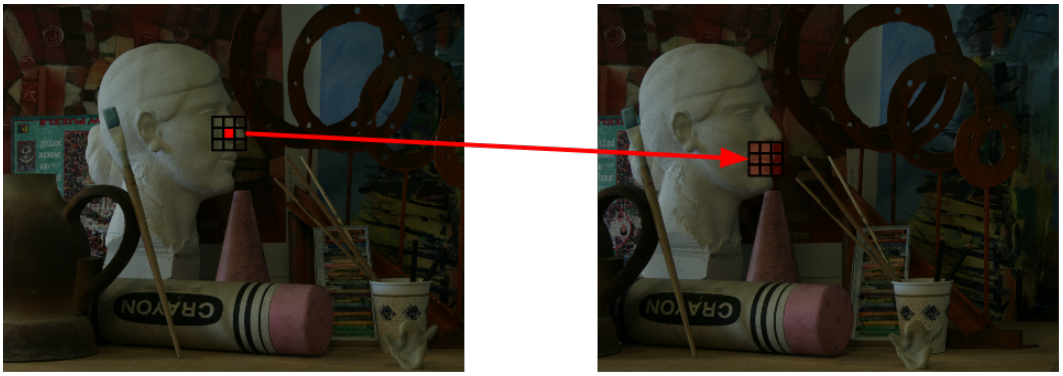

Figure 4.1 Une fenêtre est construite autour de notre pixel d'intérêt (rouge opaque) dans notre première vue (gauche) et on cherche la meilleure correspondance (fenêtre rosée) dans l'autre vue (droite)

De même qu'au chapitre précédent, il est souhaitable de rectifier et d'aligner les images capturées afin de réduire le nombre de comparaisons à faire. Nous nous placerons toujours dans ce cas. La recherche de correspondances se limitera donc à une ligne dans la seconde vue. Le calcul de disparité se résume donc à la formule suivante :

$$d(i,j) = j - argmin_k(\sum_{l \in F} (R(i,j,l) - C(i,k,l))^2)$$

Ou $d(i,j)$ est la disparité pour le pixel situé sur la i^{me} ligne et la j^{me} colonne, $R(i,j,.)$ notre fenêtre de référence autour du pixel d'intéret et $C(i,k,.)$ la fenêtre de comparaison à la position k. Notre disparité est donc bien $j - k_{min}$.
Une étude empirique nous a permis de déterminer qu'une fenêtre carrée de 9 pixels de côté utilisant la distance euclidienne pour comparer les fenêtres donnait les meilleurs résultats (voir Tableau 4.1).

4.2.1 Implémentation

Comme le montre la description précédente, nous avons ici à faire à un algorithme relativement simple à implémenter de façon usuelle en utilisant le processeur classique (Central Processing Unit, CPU). Il est cependant judicieux de bien considérer les calculs effectués, en effet on constate assez rapidement qu'il est possible d'accélérer la procédure en réutilisant une partie des calculs (voir Figure 4.2). En effet, les fenêtres de comparaison se supperposant elles partagent un certain nombre de pixels (six dans notre illustration ici) les calculs effectués sont

donc identiques. On constate par ailleurs une très forte indépendance, dans le cas de fenêtres disjointes, il est alors fort intéressant d'envisager une implémentation utilisant un très grand nombre de threads et l'utilisation de GPGPU semble parfaitement s'appliquer ici. Ma (2012) propose de plus amples détails ainsi qu'une implémentation hautement parallélisée de cet algorithme (nous encourageons le lecteur à s'y référer). Toutefois, l'algorithme proposé com-

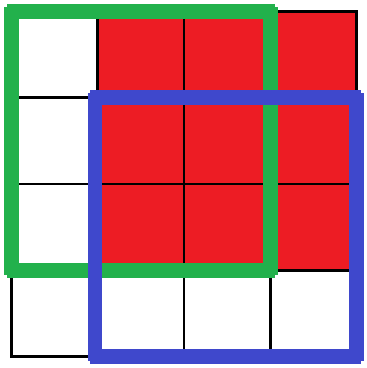

Figure 4.2 Les calculs fait pour la fenêtre rouge peuvent être réutiliser dans le cas de la fenêtre verte et la fenêtre bleu

porte quelques lacunes. Tout d'abord, d'un point de vue pratique, il se limite aux images du jeu de données Middlebury (niveau de disparité maximum et résolution fixés notamment). Par ailleurs, l'architecture GPGPU n'est pas forcément judicieusement exploitée. Nous ne reviendrons pas en détail sur la généralisation de l'algorithme, qui constitue changements mineurs, mais il est intéressant de bien comprendre comment une meilleure utilisation de la mémoire permet de diviser le temps de calcul nécessaire par deux.

L'une des erreurs les plus fréquentes quand on en vient à parler de GPGPU consiste à négliger les temps de transfert en mémoire. L'utilisation d'une carte graphique, pour effectuer des calculs, requiert souvent d'envoyer les données sur lesquelles on souhaite travailler en mémoire. Une fois les calculs effectués, il faut alors aller récupérer les résultats dans un autre tampon mémoire de la carte graphique. Ces étapes sont couteuses en temps et il arrive parfois que la parallélisation n'apporte pas un gain en temps suffisant pour combler la perte engendrée par la transmission des données.

Une autre erreur liée à l'utilisation de la mémoire, consiste à ignorer les différents niveaux de mémoires disponibles sur une carte graphique (voir Figure 4.3). La carte graphique ne donne accès en lecture et écriture qu'à ce qu'on appelle la mémoire globale. Mais les temps de transfert entre cette mémoire et les processeurs d'une carte graphique sont considérables

et doivent être limités. Il existe un autre type de mémoires à accès beaucoup plus rapide : la mémoire partagée. Il est donc très souvent judicieux de copier une partie de nos données depuis la mémoire globale dans la mémoire partagée pour ensuite travailler sur cette copie.

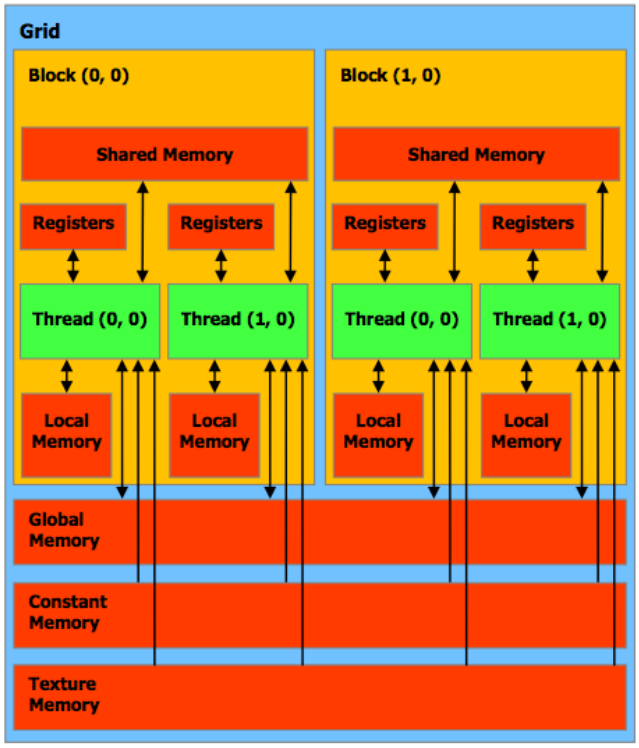

Figure 4.3 Organisation de la mémoire dans un processeur graphique (extrait de http ://www.caam.rice.edu/ timwar/RMMC/CUDA.html)

4.2.2 Choix d'implémentation

Même si nous sommes restés assez proches de l'implémentation proposée, nous avons quand même exploré différentes tailles de blocs pour aboutir à la conclusion qu'une fenêtre carrée de 9 pixels de côté donnait les meilleurs résultats. Il est cependant nécessaire de souligner un autre point qui aura son importance dans une implémentation future (Chapitre suivant).

L'importance accordée aux temps de calcul minimal ainsi que la généralité de l'implémentation vient du fait que l'on souhaite combiner cette méthode avec une autre dans un modèle probabiliste. Pour ce faire, nous avons besoin d'accéder non seulement à la meilleure valeur de disparité trouvée par l'algorithme, mais aussi aux coûts de correspondances. Nous avons

donc choisi de diviser en deux noyaux openCL cette implémentation. Dans un premier temps, nous construisons un tenseur tridimensionnel comprenant tous les coûts associés à chacune des disparités possibles. Un second noyau traite alors chaque position dans ce tenseur pour trouver la disparité de coût minimal correspondante. Nous reviendrons plus en détail sur cette procédure dans le prochain chapitre, mais il est important de faire le lien et de voir l'importance de cet algorithme dans notre modèle final.

4.2.3 Résultats

Il existe des jeux de données permettant d'évaluer la qualité de carte de disparités construites en utilisant des méthodes passives. La référence généralement utilisée et celui de Middlebury, qui fournit six différentes scènes capturées en stéréo ainsi que la carte de vérité pour chacune de ces scènes. Nous utilisons ce jeu de données non seulement pour la qualité des images fournies, mais aussi afin de pouvoir comparer nos résultats à l'état de l'art dans le domaine. Nous utiliserons donc ce jeu de données pour l'établissement de la plus part de nos résultats préliminaires.

Les modifications apportées à l'implémentation préalable ne changent pas foncièrement l'algorithme et les résultats en terme de qualité restent les mêmes. Nous obtenons cependant un gain de temps de 50% en moyenne, de 0.02 secondes nécessaires dans l'implémentation précédente nous obtenons désormais 0.01 secondes pour les calculs (pour calculer des cartes de disparités de taille 463x370), en terme de temps nécessaire pour construire nos cartes de disparités ainsi qu'un algorithme bien plus général et réutilisable pour d'autres jeux de données.

Par ailleurs, le Tableau 4.1 dresse un bref comparatif des différences en terme de précision obtenue en faisant varier la taille des fenêtres que nous comparons. Nous indiquons ici les résultats en terme de taille de rayon, les blocs, centrés sur nos pixels d'intérêt, sont donc des carrés de côté de deux fois plus un pixel. Ces résultats illustrent la grande complexité existante dans le choix de la taille optimale, pour certaines scènes un bloc de petite taille semble être préférable, mais pour d'autres un bloc plus grand donnera de meilleurs résultats.

L'annexe A contient diverses cartes de disparités obtenues sur le jeu de données Middlebury. Nous avons cependant jugé opportun de reproduire ici deux exemples extrêmes pour la taille de fenêtre optimale.

Tableau 4.1 Comparaisons de taux d'erreur des pixels visibles en pourcentage (%) pour différentes tailles de blocs possibles

Méthode	μ	*Arts*	*Books*	*Dolls*	*Laundry*	*Moebius*	*Reindeer*	CPU time (s)	GPU time (ms)
rayon = 2	24.65	26.25	28.98	14.25	34.86	22.77	20.80	0.85	9.1
rayon = 3	23.08	25.92	22.23	14.26	36.21	20.88	18.99	0.85	9.5
rayon = 4	22.50	27.00	19.39	15.14	34.12	20.61	18.72	0.86	9.9
rayon = 5	22.66	28.33	17.73	16.40	32.86	21.08	19.54	0.86	10.4
rayon = 6	23.43	30.05	17.23	18.08	32.01	21.88	21.32	0.87	11.0

4.3 Census

Le constat derrière l'algorithme du Census est que le bruit propre au système d'acquisition, les variations d'éclairage et de positionnement de notre paire de caméras stéréoscopiques et le capteur déforment localement l'image et nuisent à la correspondance par bloc. L'utilisation de filtre semble alors tout indiquée, mais les filtres usuels, tels que les filtres gaussiens ou bilatéraux et plus généralement l'ensemble des filtres paramétriques, vont, pour leur part, avoir tendance à supprimer de l'information essentielle à une bonne mise en correspondance. Zabih et Woodfill (1994) ont donc eu l'idée de considérer des filtres non paramétriques, des filtres dépendant entièrement des valeurs auxquelles on les applique. La première proposition était d'utiliser le rang de chaque pixel. Ce filtre consiste à considérer une fenêtre autour d'un point d'intérêt et à y compter le nombre de pixels d'intensités qui lui sont inférieures. On peut donc voir le rang comme la position du pixel d'intérêt dans une version triée par valeurs d'intensités de la fenêtre. On peut définir le rang par la formule :

$$r(i,j) = \sum_{l \in F} \xi_{ij}(I_F(i,j,l))$$
$$\xi_{ij}(k) = \begin{array}{r} 0 \text{ si } I(i,j) \leq k \\ 1 \text{ si } I(i,j) > k \end{array}$$

Où $r(i,j)$ est le rang du point (i,j) dans notre image I coupée en fenêtres I_F, $\xi_{ij}(k)$ la fonction qui détermine si l'intensité k est supérieure à celle du point considéré, F représentant toujours notre fenêtre de pixels et $I_F(i,j,.)$ l'ensemble des intensités dans cette fenêtre. Il est alors possible de faire une correspondance par bloc sur les versions filtrées de chacune des deux images en cherchant des blocs ou les rangs, et non plus les intensités, correspondent.
L'utilisation d'une telle approche améliore légèrement les résultats, mais on constate assez facilement qu'on perd beaucoup d'information lors du filtrage. Zabih et Woodfill (1994) ont alors eu l'idée de conserver une partie de cette information en plus du calcul du rang en mémorisant la position des pixels. Cette stratégie aboutie à la méthode dite du Census, au

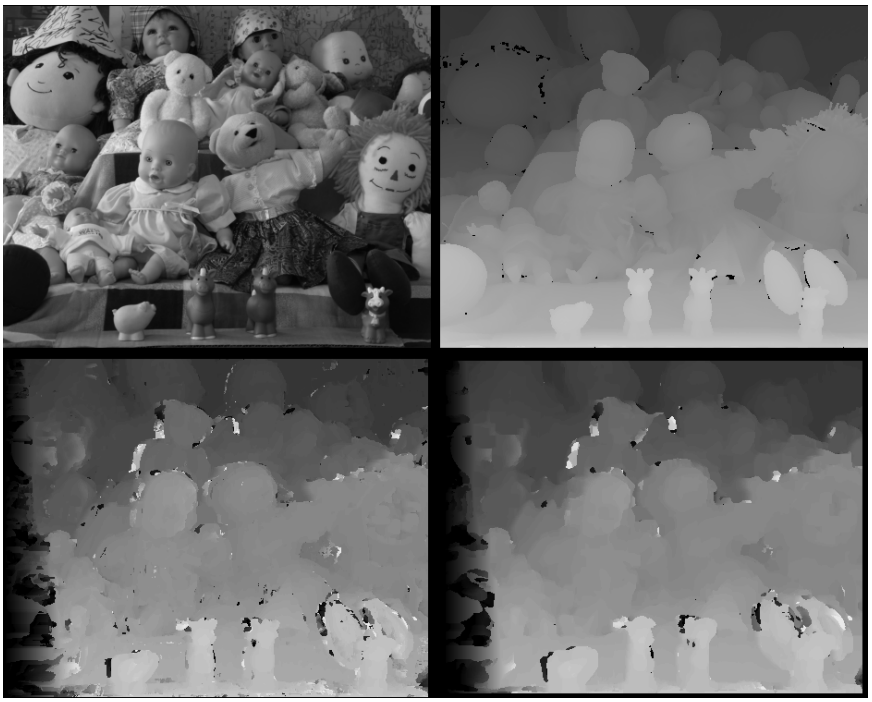

Figure 4.4 Scène vue de gauche (haut gauche) et base de vérité (haut droit) de la scène Dolls et résultats obtenus pour un rayon de 2 (bas gauche, meilleur) et de 6 (bas droit) en utilisant la correspondance par blocs

lieu de se contenter de compter le nombre de pixels dont l'intensité est inférieure à celle du centre de la fenêtre le Census vise à retenir leur position ainsi que la comparaison avec le pixel d'intérêt. On veut donc construire un descripteur pour chacun des pixels. Par exemple si l'on considère une fenêtre carrée de trois pixels de côté avec les intensités suivantes :

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 2 \\ 2 & 2 & 0 \end{matrix}$$

On peut garder la forme de la fenêtre tout en appliquant notre fonction ξ. On obtient alors :

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Notons que la valeur au centre de notre descripteur sera toujours zéro, l'intensité étant comparée à elle-même. Cette valeur est donc généralement ignorée. De même, on peut réordonner notre descripteur en une chaine de bits ce qui facilite la manipulation, mais ne change pas le fonctionnement de l'algorithme.
À des fins de visualisation, et uniquement dans ce but, nous pouvons transformer la version

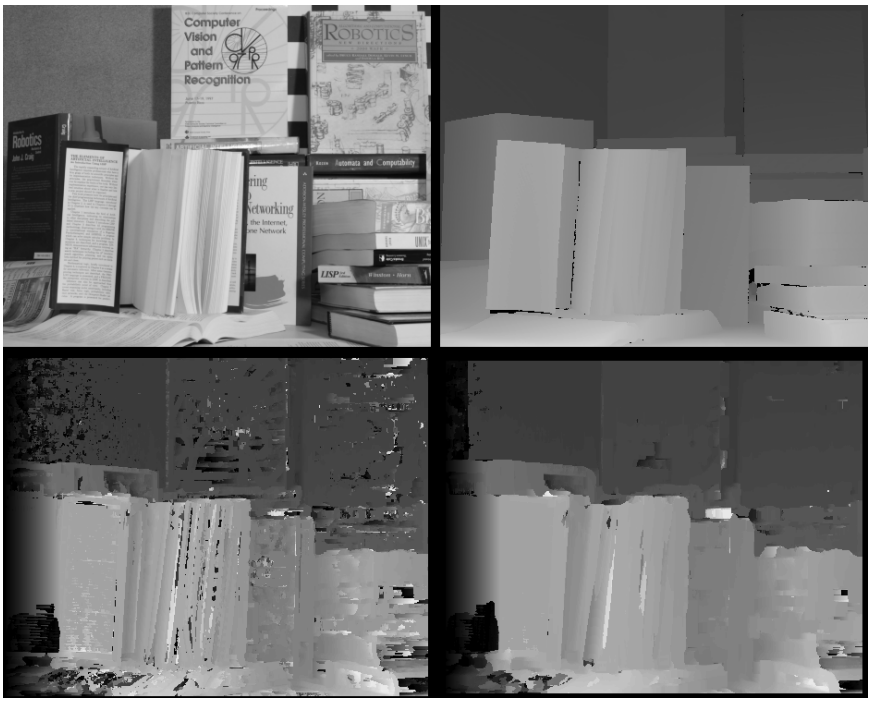

Figure 4.5 Scène vue de gauche (haut gauche) et base de vérité (haut droit) de la scène Books et résultats obtenus pour un rayon de 2 (bas gauche) et de 6 (bas droit, meilleur) en utilisant la correspondance par blocs

filtrée de notre image en une nouvelle image : chacun des bits de nos descripteurs servant alors à l'encodage d'une couleur (voir figure 4.6).

Une fois encore, la dernière étape consiste à utiliser une approche de correspondance par blocs, en considérant des blocs de descripteurs et une distance valide sur cette représentation. Aux vues du caractère binaire de nos descripteur, une distance euclidienne entre deux descripteurs n'aurait pas grand sens, on utilise donc la distance de Hamming qui mesure la dissimilitude de chacun des bits de nos fenêtres d'intérêt. Cette distance peut s'écrire :

$$d(R,C) = \sum_{i \in F} \sum_{j \in D} 1 - \delta(R(i,j), C(i,j))$$

Où $d(R,C)$ représente la distance entre notre fenêtre de référence R et celle de comparaison C, toutes les deux contenant F descripteurs de dimension D et δ l'indicatrice d'Euler :

$$\delta(i,j) = \begin{array}{l} 1 \text{ si i} = \text{j} \\ 0 \text{ si i} \neq \text{j} \end{array}$$

Si l'on considère par exemple les deux descripteurs :

$$\text{R} = \begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad \text{C} = \begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Figure 4.6 Image Art extrait du jeu de données Middlebury (à gauche) et sa version filtrée parla technique du Census (à droite).

On obtient une distance de $d(R, C) = 3$ en effet nous avons trois bits correspondants ayant des valeurs différentes.

4.3.1 Choix d'implémentation

Cet algorithme est également hautement parallélisable et parfaitement adapté pour l'utilisation de GPGPU. Nous avons donc fait une implémentation en OpenCL compatible avec notre implémentation de la correspondance par bloc.
Notre implémentation se divise en trois noyaux, le premier s'occupe du filtrage de nos images et convertit les valeurs d'intensités en descripteurs locaux pour chaque pixel. Notre deuxième noyau calcule l'ensemble des coûts de correspondances en utilisant la distance de Hamming sur la version filtrée de l'image et enregistre ces coûts dans un tenseur tridimensionnel. Enfin, notre dernier noyau cherche pour chaque position la disparité minimisant le coût dans notre tenseur.
Une fois encore, nous avons divisé le calcul des coûts et la recherche du minimum afin de pouvoir ultérieurement effectuer différentes manipulations sur les coûts afin d'améliorer notre carte de disparités.
Pour la transformation de notre image en descripteur, chaque bloc est responsable d'une portion continue de l'image. Ceci nous permet dans un premier temps d'utiliser tous nos threads pour charger l'ensemble des pixels en mémoire partagée en les transformant en valeurs d'intensités. Cela réduit ensuite le nombre de calculs ainsi que celui d'accès couteux à la mémoire globale. De même, la mise en correspondance de plusieurs blocs consécutifs sur plusieurs lignes voisines se fait dans le même groupe de threads afin de réutiliser autant que possible les calculs et de faire le meilleur usage de la mémoire partagée. Cette mémoire étant limitée, et vu les nombreuses façons d'agencer les calculs, trouver la combinaison optimale

est une tâche compliquée et il est probablement possible d'améliorer nos résultats. Toutefois, aux vues du coût en temps nécessaire pour calculer la distance de Hamming la réutilisation des calculs semble être non négligeable.
De même que pour la correspondance par bloc, la taille des fenêtres joue énormément sur les résultats une étude empirique nous a permis de trouver une taille optimale de neuf par neuf pour le calcul des descripteurs et de onze par onze pour la mise en correspondance.

4.3.2 Résultats

Cette implémentation montre une fois encore la puissance de la parallélisation massive. En effet, nous sommes partis d'une implémentation Matlab prenant de l'ordre de 5 minutes par carte de disparités (de taille 463x370), une fois ce prototypage effectué, une version C++ sur CPU a alors été implémentée, mais elle nécessitait toujours une vingtaine de secondes pour effectuer les calculs. Notre version finale sur GPGPU (utilisant OpenCL) tourne en 0.03 seconde. La forte différence de temps nécessaire par rapport à la correspondance par bloc, s'explique par le fait qu'il est nécessaire de construire une version filtrée de chacune de nos vues d'une part, mais elle est également due au fait que la distance est plus couteuse à calculer. La même réutilisation des calculs est cependant possible et a un impact encore plus considérable et divise les temps de calcul par 3 (de 0.09 à 0.03 seconde).
Nous proposons par ailleurs la même étude sur l'impact de la taille du rayon sur la qualité de la carte de disparité produite. Nous avons cependant deux rayons qu'il est possible de faire varier : le rayon lors de notre filtrage ainsi que celui lors de la mise en correspondance. Une étude préliminaire nous a permis de confirmer les résultats présents dans la littérature établissant un rayon de 4 pour le filtrage. Le tableau 4.2 présente les résultats sur les variations pour la mise en correspondance.
Une fois encore on constate qu'il n'y a pas de taille idéale pour tous les cas. On constate

Tableau 4.2 Comparaisons de taux d'erreur des pixels visibles en pourcentage (%) des différentes tailles de blocs possibles

Méthode	μ	*Arts*	*Books*	*Dolls*	*Laundry*	*Moebius*	*Reindeer*	CPU time (s)	GPU time (ms)
rayon = 2	14.03	14.56	14.54	7.351	26.22	12.79	8.721	18	24
rayon = 3	13.95	15.44	13.16	8.033	24.94	12.78	9.341	21	30
rayon = 4	14.37	16.53	12.54	9.069	24.76	13.14	10.21	24	37
rayon = 5	15.10	17.46	12.75	10.33	25.09	13.79	11.20	28	45
rayon = 6	16.04	18.30	13.53	11.69	25.47	14.70	12.52	33	55

cependant que la taille des fenêtres adaptées a tendance à être plus petite, ce qui se justifie par le fait que notre image filtrée prend en compte un ensemble de pixels pour chaque descripteur. Une fenêtre de descripteur recouvre donc plus de pixels dans l'image d'origine.
Une fois encore pour l'intégralité des cartes de disparités obtenues grâce à cette technique nous invitons le lecteur à se référer à L'annexe A. À des fins de visualisations et de comparaisons rapide avec les résultats de la méthode précédente, nous présentons ici la Figure 4.7 illustrant la carte de disparité obtenue sur la scène Books avec chacune des deux tailles extrêmes de blocs à comparer.

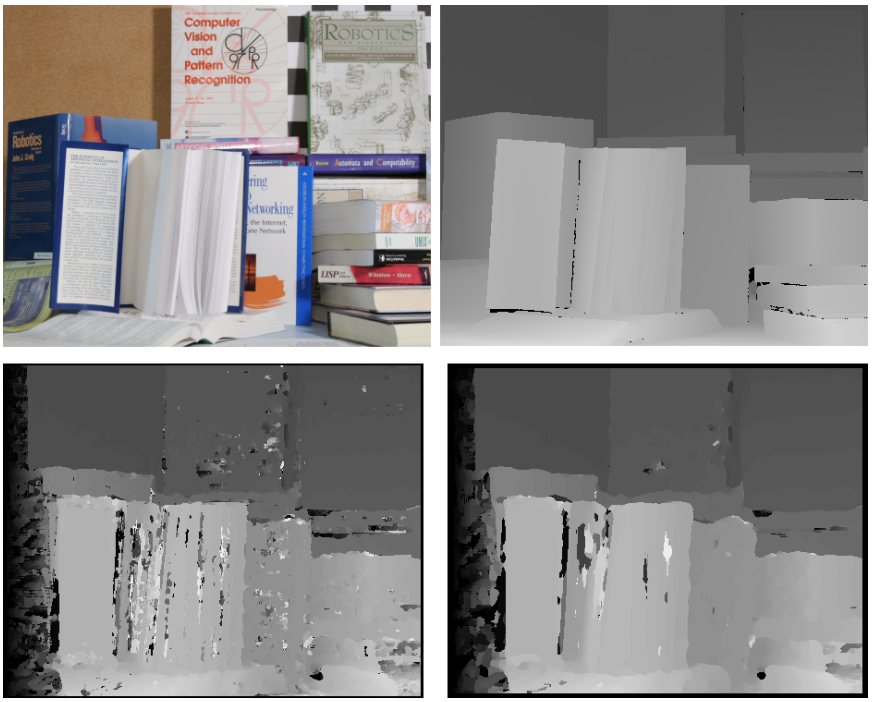

Figure 4.7 Scène vue de gauche (haut gauche) et base de vérité (haut droit) de la scène Books et résultats obtenus pour un rayon de 2 (bas gauche) et de 6 (bas droit) par la méthode du Census

4.4 Modèle de Markov Caché

Pour conclure ce chapitre sur les étapes préliminaires à l'établissement d'un modèle nous allons une fois de plus développer un point déjà présent dans les travaux précédents du laboratoire : l'utilisation de Modèles de Markov Caché (MMC ou HMM). Toutefois, il est primordial de bien appréhender la théorie de base que nous développons rapidement ici et pour laquelle Ma (2012) constitue une bonne lecture. Nous couvrirons cependant les bases nécessaires pour bien comprendre le raisonnement et les modèles présentés par la suite. Nous présentons

également une nouvelle implémentation propre au traitement stéréoscopique massivement parallélisé. Russell et Norvig (2003) (chapitre 15) présente une très bonne introduction à ce genre de modèles probabilistes.

4.4.1 Théorie

Un MMC sert à modéliser un processus stochastique temporel décrit en tout temps par une unique variable aléatoire discrète. Chaque variable dépendant de celle au temps précédent et n'étant perçue qu'au travers une observation stochastiquement bruitée. La Figure4.8 illustre les liens entre les variables ainsi que le principe d'observations et de variables cachées. Les MMC sont des modèles très puissants et couramment utilisés pour l'inférence, un raisonnement probabiliste sur l'ensemble des variables observées permet en effet de décoder avec une certaine précision pour retrouver les variables cachées.

Dans notre cas nous considérons des cartes de disparités et nous ne nous intéressons pas à

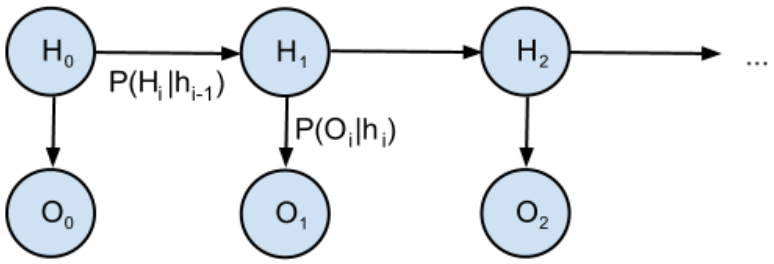

Figure 4.8 Modèle de Markov Caché où H représente les variables cachées et O les observations.

la cohérence temporelle, mais il est possible d'adapter notre MMC pour modéliser une suite de points continue dans notre image. Concrètement une ligne de pixel dans notre carte de disparités. En effet, le principe de continuité nous impose une relation liant probabilistiquement chaque point à son voisin. On peut donc considérer qu'une carte de disparités calculée représente une version bruitée de la carte de vérité. Une série de MMC, bien paramétrés, nous permet alors d'inférer une version moins bruitée : on cherche à retrouver les vraies valeurs de disparités (cachées) qui nous ont conduits à observer celles calculées précédemment.

4.4.2 Viterbi

Il existe de nombreux algorithmes visant à décoder des MMC ou plus généralement des modèles probabilistes temporels, ou modèles stochastiques. Le plus connu et le plus couramment utilisé est l'algorithme de Viterbi, cet algorithme se décompose en deux étapes. Dans un

premier temps, on propage les probabilités sur les nœuds du réseau MMC, puis une fois tout le réseau parcouru, on remonte le chemin (backtrack) aboutissant au maximum de probabilité.

Propagation de probabilité

Un MMC comprend deux probabilités conditionnelles : la probabilité de transition et la probabilité d'émission. La probabilité de transition modélise la probabilité de passer d'une valeur de variable cachée au temps t à une autre valeur au temps $t+1$. La probabilité d'émission, quant à elle, sert à modéliser les variations, le bruit, existant entre les observations et les variables cachées. La Figure4.8 situe ces deux probabilités sur notre modèle.
La propagation de probabilité consiste à calculer récursivement la probabilité qu'a chacune de nos valeurs cachées d'être dans un état donné. L'utilisation de la récursion vient du fait que la densité probabilité marginale pour une variable dépend de celle qui précède ainsi que de la valeur observée. Cette récurrence se traduit par la formule :

$$P(H_{t+1} = X) = max_i(P(H_{t+1} = X|H_t = i)P(H_t = i))P(o_{t+1}|H_{t+1} = X)$$

Où $P(H_{t+1} = X|H_t = i)$ est la probabilité de transition de l'état i à t à l'état X à $t+1$ et $P(o_{t+1}|H_{t+1} = X)$ est la probabilité d'émission dans l'état X pour l'observation o_{t+1}. L'initialisation est généralement faite à l'aide de connaissance a priori ou en utilisant une probabilité uniforme revenant à se baser uniquement sur l'émission et l'observation. On voit donc toute l'importance d'une bonne modélisation de nos probabilités conditionnelles.
En pratique cet algorithme n'est pas directement exploitable dû au grand nombre de multiplications de petites valeurs nécessaires (manipulation de probabilités). Cela engendre en effet des instabilités numériques (arrondis à 0), pour résoudre ce problème on utilise le log de nos probabilités. La fonction logarithme présente l'avantage d'être strictement croissante et la recherche de maximum n'est donc pas impactée et les faibles différences des probabilités sont augmentées.

Décodage

Une fois la propagation terminée, on a alors une densité de probabilité pour la dernière variable de notre chaine. On peut donc utiliser le maximum a posteriori (MAP) pour déterminer quel doit être l'état de cette variable. On a également calculé les densités de chacune des variables précédentes et l'on pourrait être tenté de les utiliser pour déterminer leur état ainsi que leur valeur. Mais cette stratégie ne serait pas correcte, en effet elle perdrait la cohérence future de la chaine : chaque variable serait vue comme la fin du MMC et les résultats

produits seraient erronés. La stratégie adoptée pour résoudre ce problème dans le cadre de l'algorithme de Viterbi consiste à utiliser une méthode de "back-tracking".
Lors de la propagation, en plus de déterminer chaque valeur de probabilité, on enregistre la transition qui l'engendre : l'état j à l'instant $t-1$. On a alors :

$$j = argmax_i(P(H_{t+1} = X|H_t = i)P(H_t = i))$$

Le maximum de probabilité de la variable finale représente alors un chemin à travers notre MMC : une suite $(j_t)_0^N$ d'états par lesquels on doit passer pour maximiser la probabilité finale. La densité de probabilité finale ne représente en effet pas la densité de probabilité de notre dernière variable, mais bien la densité de toute notre suite d'états. Cette approche nous permet donc de décoder le MMC, déterminer la valeur la plus probable pour chacun des états, en ne faisant qu'un parcours au travers chacune des variables.

Implémentation

Il existe de nombreuses implémentations optimisées pour différentes tâches pouvant aller de la finance aux télécommunications. Parmi ces implémentations, Nvidia en fournit une en OpenCL et l'on pourrait être tenté de l'utiliser (voir travaux précédents). Cependant, aucune de ces implémentations disponibles n'est optimisée pour effectuer l'intégralité des calculs nécessaires directement en OpenCL. L'implémentation Nvidia par exemple se restreint à faire les calculs pour une unique variable au profit d'un très grand nombre d'états possibles. Utiliser cette implémentation nécessite donc de multiplier le nombre d'appels aux noyaux nécessaires ainsi que le nombre de transferts mémoires. Les performances sur un grand nombre de variables et un faible nombre d'états s'en trouvent donc détériorées. Nous avons donc implémenté notre propre version afin de réduire autant que possible les temps de calcul.
À première vu, l'algorithme de Viterbi n'est pas approprié pour la parallélisation massive et l'emploie de technologies telles qu'OpenCL. En effet, chaque variable dépendant de la précédente et l'intégralité du chemin parcouru n'apparait qu'une fois les calculs pour la dernière variable effectués. On peut se demander comment paralléliser cet algorithme. Cependant en y regardant bien on constate que pour chacune des variables la probabilité $P(H_{t+1} = X|h_t)$ peut être calculée séparément pour chaque couple (X, h_t). Une approche de diviser pour régner permettra ensuite de retrouver la valeur maximale pour chacune des valeurs de h_t.
Il est également intéressant de considérer que dans notre cas nous souhaitons de plus traiter une image en la considérant comme une série de MMC indépendants et donc traitables en parallèle.
Notre implémentation découle donc de ces constats. Chacun de nos blocs (au sens OpenCL)

traite ainsi une valeur de X possible et chacun des threads calcul dans un premier temps la probabilité pour une transition donnée. Une fois tous les $P(H_{t+1} = X|h_t)$ calculés, lorsque tous les threads ont fini un bloc va alors chercher le maximum de probabilité. Pour cette étape chaque thread compare deux valeurs et enregistre le maximum avant de recommencer sur deux nouvelles valeurs (ou de s'arrêter s'il n'y a plus assez de valeurs à comparer). Lorsque le dernier thread termine, nous avons calculé $max_i(P(H_{t+1} = X|H_t = i)P(H_t = i))$ tout en enregistrant l'argument maximal. Le bloc peut alors soit traiter une nouvelle valeur de X si nécessaire soit, une fois tous les X traités, passer à la variable H_{t+2} et recommencer de façon exactement similaire jusqu'à ce que l'on soit en mesure de décoder notre HMM.
Deux possibilités s'offrent alors pour la phase de bactracking : décoder en GPU ou en CPU. Le décodage est en effet très linéaire et consiste uniquement à corriger notre carte de disparité. L'utilisation du CPU sera donc plus rapide pour cette tâche, mais cela nécessite de transférer l'intégralité des indices de transition de la mémoire vidéo à la mémoire vive avant d'effectuer ce calcul. Effectuer le calcul sur le GPU, plus lent sur un seul thread, permet par contre de réduire la taille du transfert final. Dans la pratique nous n'avons pas mesuré de différence significative et avons opté pour effectuer l'intégralité des calculs en GPU.

4.4.3 Paramétrisation

Comme mentionné précédemment, une bonne modélisation des probabilités est nécessaire au bon fonctionnement de notre modèle. Plus nos probabilités seront proches de la réalité plus notre carte de disparités débruitée sera de bonne qualité. Le contre-coût de probabilités trop proche d'une scène est qu'alors notre modèle risque de ne pas généraliser, l'obtention de ces probabilités par ailleurs repose sur la connaissance d'une base de vérité qui est couteuse à obtenir.
Les probabilités sont généralement paramétrées à partir de connaissances sur des scènes similaires également numérisées. Afin d'améliorer la portabilité de notre modèle et de pouvoir l'utiliser sur plusieurs scènes, nous utilisons uniquement trois paramètres pour modéliser notre probabilité. Nous décomposons donc en probabilité de ne pas changer d'état (α_T)ou de ne pas avoir de bruit (α_E), la probabilité de passer dans un état avec une faible variation, un degré de disparité par exemple (β_T, β_E) et la probabilité d'avoir de fortes variations (γ_T, γ_E).

Transition :

$$P(H_{t+1} = h_t|h_t) = \alpha_T$$
$$P(|H_{t+1} - h_t| < 2|h_t) = \beta_T$$
$$P(|H_{t+1} - h_t| > 1|h_t) = \gamma_T$$

Emission :

$$P(o_t = h_t|h_t) = \alpha_E$$
$$P(|o_t - h_t| < 2|h_t) = \beta_E$$
$$P(|o_t - h_t| > 1|h_t) = \gamma_E$$

Nous calculons chacun de ces paramètres en utilisant une base de vérité ainsi qu'une version bruitée de notre carte de disparités pour plusieurs scènes similaires. En parcourant pixel par pixel ces deux cartes, il est aisé de calculer la moyenne de chacun des cas qui nous permettent alors d'approximer les probabilités. Chaque transition dans la base de vérité nous donne les probabilités de transition tandis que la comparaison entre la version bruitée et la table de vérité nous donne les probabilités d'émission.

4.4.4 Résultats

Nous comparons les résultats obtenus en utilisant la correspondance par bloc, le census, un MMC horizontal sur chacune des lignes des résultats obtenus par census.
Ces résultats rappellent les lacunes de la correspondance par bloc comparativement au Census. L'expérience illustre également la puissance et le grand intérêt qu'apporte l'utilisation de modèles probabilistes avec un gain moyen de 9.18%. Les paramétrisations et tests présentés ici se basent sur une approche dite du leave one out : toutes les données de Middlebury sont utilisées pour la paramétrisation à l'exception de celle sur laquelle on souhaite mesurer les performances.
L'utilisation de MMC se caractérise par un phénomène de trainées lié au moyenage qui se

Tableau 4.3 Comparaisons de taux d'erreur des pixels visibles en pourcentage (%) des quelques combinaisons de méthodes

Méthode	μ	*Arts*	*Books*	*Dolls*	*Laundry*	*Moebius*	*Reindeer*	CPU time (s)	GPU time (ms)
BM	22.50	27.00	19.39	15.14	34.12	20.61	18.72	0.86	9.9
Census	13.95	15.44	13.16	8.033	24.94	12.78	9.341	21	30
MMC	13.09	14.36	12.74	7.504	23.19	12.20	8.590	120	176

produit lors de la phase de décodage. Pour illustrer ce phénomène, nous présentons ici les résultats obtenus sur la scène Art du jeu de données Middleburry (voir Figure 4.9). Nous utiliserons également cette scène dans la suite de ce mémoire à des fins de comparaison visuelle de nos résultats. L'ensemble des cartes de disparités obtenues grâce à cet algorithme peut être consulté en Annexe A.

4.5 Discussion

Nos meilleurs résultats sont obtenus en utilisant successivement deux MMC, cela nous permet en effet de prendre en considération la cohérence verticale et horizontale. Mais la superposition de deux modèles probabilistes unidimensionnels ne constitue pas un modèle bidimensionnel. De plus, l'algorithme de Viterbi ne maximise pas la probabilité pour chaque variable, mais la probabilité d'un ensemble de variables. Enfin, nous cherchons à débruiter les disparités a posteriori, mais on a alors perdu l'information sur les coûts et de bons matchs peuvent être ignorés. On constate donc que cette approche, même si elle donne des résultats corrects, peut être amélioré.

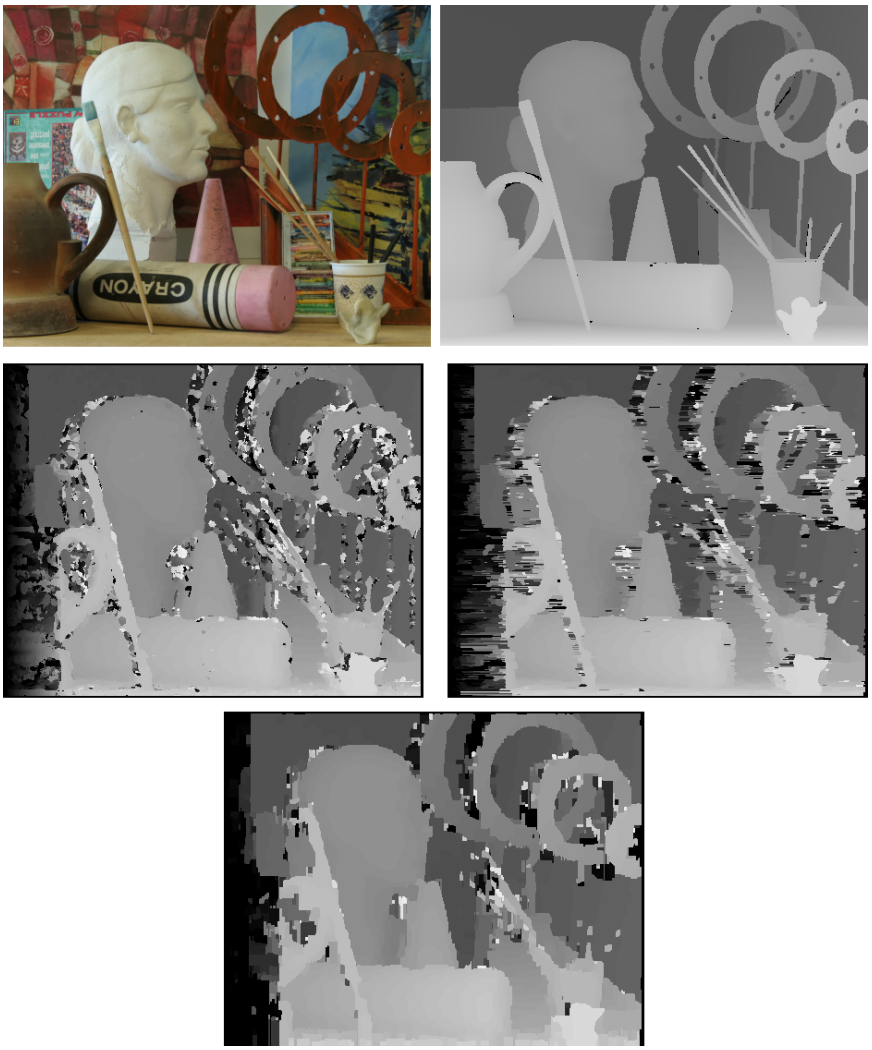

Figure 4.9 Scène vue de gauche (haut gauche) et base de vérité (haut droit) de la scène Art et résultats obtenus pour un rayon de 3 en utilisant l'approche du Census (milieu gauche) puis en appliquant l'algorithme de Viterbi horizontalement sur ces premiers résultats (milieu droit) et enfin verticalement (bas)

CHAPITRE 5

Modèle bidimensionnel

Dans ce chapitre nous présentons un nouveau modèle probabiliste bidimensionnel permettant de traiter l'ensemble des coûts afin de construire une carte de disparités de bonne qualité. Dans un premier temps, nous revenons sur l'intérêt de l'utilisation de modèles probabilistes directement sur les coûts et non sur les disparités. Nous verrons ensuite comment combiner efficacement des modèles unidirectionnels afin d'en construire un bi-directionnel.

5.1 Forward/Backward

5.1.1 Théorie

Dans le chapitre précédent, nous avons introduit l'algorithme de Viterbi et son implémentation pour le calcul de cartes de disparités. Cependant, cet algorithme ne calcule pas les densités de probabilité de chaque variable, mais plutôt un chemin de plus grande probabilité. Il existe d'autres algorithmes permettant de décoder des MMC tel que l'algorithme du forward/backward. Cet algorithme se décompose également en deux passes, mais au lieu de faire du back-tracking, cette fois-ci on propage également des messages depuis la fin du MMC en combinant les probabilités partielles propagées dans les deux directions, ce qui amène à la construction d'une probabilité marginale.
La première étape de l'algorithme consiste à transformer le MMC, purement génératif, en un modèle hybride, conjoint pour les variables cachées, et génératif pour les observations (voir Figure 5.1). Ce dernier modèle est plus intéressant que le premier vu que la notion de temps et de lien génératif existant entre les variables cachées ne se justifie pas vraiment dans notre cadre. Cependant contrairement à l'algorithme de Viterbi nous n'avons plus accès aux probabilités aussi simplement, nous avons besoin de définir une fonction jointe Ψ ainsi qu'une fonction de normalisation Φ et nous avons alors :

$$\begin{aligned} P(\{o\},\{d\}) &\neq \prod_{i=1} P(o_i|d_i)P(d_i|d_{i-1})P(d_0) \\ &= \frac{\prod_{i=1} \Psi(d_{i-1},d_i)P(o_i|d_i)}{\prod_{i=1} \Phi(d_i)} \end{aligned}$$

Afin de faciliter le calcul de ces fonctions, il est commun d'utiliser une version modifiée de notre graphe mettant en avant les connexions entre nos variables. Ce nouveau graphe est

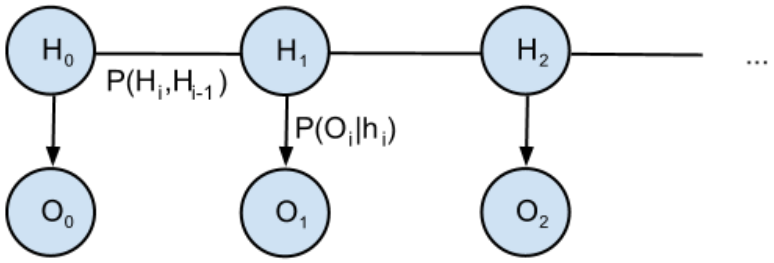

Figure 5.1 Modèle hybride mieux adapté aux observations.

appelé graphe des facteurs (factor graph, voir figure 5.2). Le calcul des valeurs de nos fonctions Φ et Ψ se fait alors par une approche de propagation de messages en utilisant un algorithme de max produit sur ce graphe. La figure 5.2 illustre le passage itératif des messages, dans le cas d'un graphe des facteurs d'un modèle conjoint (ne prenant pour le moment pas compte des observations), en suivant les règles suivantes :

$$\mu_{f_j \to X_i}(x_i) = max_{x_{i\pm1}}(\mu_{X_{i\pm1} \to f_j}(x_{i\pm1}) f_j(x_{i\pm1}, x_i)$$
$$\mu_{X_i \to f_j}(x_i) = \mu_{f_{j\pm1} \to X_i}(x_i)$$

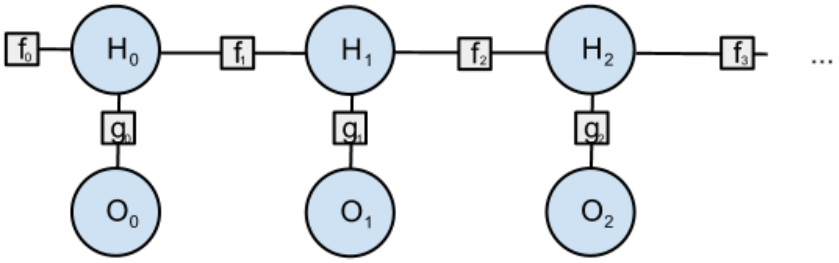

Figure 5.2 Graphe des facteurs partiel pour notre modèle.

Avec $j = i$ ou $j = i - 1$ selon le sens du message que l'on souhaite passer, f marquant le lien unissant deux variables aléatoires X pouvant prendre les valeurs x. Pour l'initialisation $f_0(x_1)$ est un a priori à fixer tandis que $\mu_{X_N \to f_{N-1}}(x_N) = 1$. On obtient alors la densité de probabilités pour chaque variable :

$$\frac{\Psi(x_{i-1}, x_i)}{\Phi(x_i)} = \mu_{f_{i-1} \to X_i}(x_i) \mu_{f_i \to X_i}(x_i)$$

Le schéma 5.2 illustre le passage des différents messages dans notre MMC à travers un exemple. On a alors :

$$f_i(H_{i-1}, H_i) = P(H_i | H_{i-1})$$
$$g_i(O_i, H_i) = P(O_i | H_i)$$
$$\mu_{X_i \to f_{i-1}}(x_i) = \mu_{f_i \to X_i}(x_i) \mu_{g_i \to X_i}(x_i)$$

On obtient alors en réinjectant et en regroupant ces expressions et en utilisant des probabilités partielles pour les propagations forward P_f et backwar P_b :

$$P_f(H_i) = max_k(P(H_i|H_{i-1} = k)P_f(H_{i-1} = k)P(o_{i-1}|H_{i-1} = k))$$
$$P_b(H_i) = max_k(P(H_{i+1} = k|H_i)P_b(H_{i+1} = i)P(o_{i+1}|H_{i+1} = k))$$
$$P(H_i) = P_f(H_i)P_b(H_i)P(o_i|H_i)$$

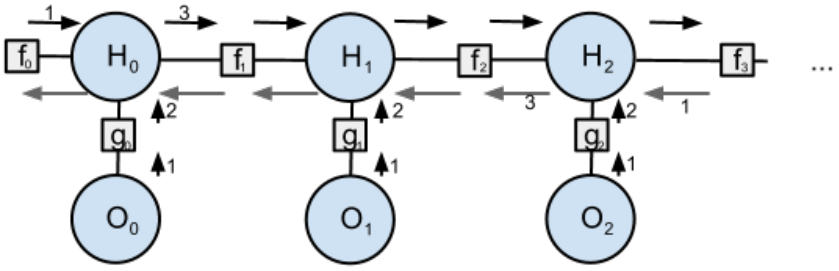

Figure 5.3 Passage des messages dans l'algorithme du forward (noir) backward (gris).

Cette stratégie est très proche de l'algorithme de Viterbi, en effet la phase "forward" correspond à la propagation des probabilités dans notre MMC. Les résultats sont donc très proches, mais l'utilisation de cette stratégie apporte une amélioration dans la qualité de notre carte. En contrepartie, elle nécessite d'effectuer plus de calculs. Une fois ces calculs faits nous avons toutefois accès aux probabilités de chaque variable indépendamment on peut donc envisager d'utiliser ces probabilités pour les combiner avec d'autres par exemple dans le cas d'un MMC vertical et d'un MMC horizontal.

5.1.2 Algorithme

L'algorithme repose encore une fois sur les modélisations des probabilités de transition et d'émission. Encore une fois par récurrence, on peut calculer des probabilités partielles et les propager. Pour chaque variable on calcule ainsi les mêmes messages que dans le cas de Viterbi. En effet, on peut modifier légèrement la notation pour inclure la partie générative dans notre message P_f :

$$P_f(H_t = X) = max_i(P(H_t = X|H_{t-1} = i)P_f(H_{t-1} = i))P(o_t|H_t = X)$$

Il s'agit alors bien de la même probabilité que dans l'algorithme de Viterbi. On leur ajoute alors les probabilités retours (back-probabilities). Il s'agit d'une combinaison similaire de nos probabilités qui vise à déterminer quelles chances a-t-on, pour une variable cachée dans un certain état, d'aller dans la suivante pour laquelle on connait des probabilités partielles retour et d'émettre alors notre observation. La formulation mathématique est très symétrique :

$$P_b(H_t = X) = max_i(P(H_{t+1} = i|H_t = X)P_b(H_{t+1} = i)P(o_{t+1}|H_{t+1} = i))$$

On obtient alors, avec cette nouvelle notation, les probabilité finales pour chaque noeud :

$$P(H_t) = P_f(H_t)P_b(H_t)$$

Dans cette version finale la partie générative est incluse dans P_f est n'apparait donc pas explicitement, mais est bien présente.
Une fois encore, on utilise les log-probabilités plutôt que les vraies probabilités pour éviter les instabilités numériques. Une fois les probabilités finales calculées, on peut décoder notre MMC en maximisant chacune des variables indépendamment.

5.1.3 Implémentation

L'implémentation change très peu par rapport à celle de l'algorithme de Viterbi décrite au chapitre précédent : dans un premier noyau, nous enregistrons cette fois l'ensemble des probabilités partielles dans un tenseur tridimensionnel (une matrice par MMC, et un MMC par ligne de nos images) et non plus le meilleur chemin pour chaque état de chaque variable (forward). Un second noyau complète les calculs après propagation des messages dans le sens inverse (backward). En sortie de l'algorithme, nous avons un tampon de mémoire contenant l'intégralité des probabilités pour chacune de nos variables cachées. Un troisième noyau fait alors une recherche de maximum pour produire une version débruitée de notre carte de disparités.

5.1.4 résultats

Nous avons comparé les performances de cet algorithme avec celui de Viterbi, autant d'un point de vue vitesse d'exécution que de qualité de la carte de disparités produite sur le dataset de Middlebury. Les résultats sont retranscrits dans le tableau 5.1. On constate qu'assez logiquement le plus grand nombre de calculs nécessaires, lié au "backward", impacte le temps nécessaire pour la création de la carte et que les gains apportés en terme de qualité sont assez minimes. Toutefois, comme nous l'avons mentionné précédemment l'algorithme de forward/backward présente l'avantage de calculer la densité de probabilité de chacune de nos variables, or ces probabilités seront primordiales par la suite.
Viterbi et 2-Viterbi sont un rappel des résultats obtenus au chapitre précédent afin de faciliter la comparaison avec les résultats obtenus en utilisant l'algorithme du forward/backward

appliqué horizontalement sur la carte produite par l'approche du census (F/B) ou verticalement sur les résultats obtenus horizontalement (2-F/B).

Une fois encore nous proposons de comparer visuellement les résultats obtenus à l'aide de

Tableau 5.1 Comparaison des résultats obtenus après débruitage en utilisant l'algorithme de Viterbi (rappel) et l'algorithme du Forward/Backward.

Méthode	μ	*Arts*	*Books*	*Dolls*	*Laundry*	*Moebius*	*Reindeer*	CPU time (s)	GPU time (ms)
Viterbi	13.09	14.36	12.74	7.504	23.19	12.20	8.590	120	176
2-Viterbi	12.67	13.87	12.00	7.125	22.63	11.89	8.481	240	295
F/B	12.97	14.23	12.38	7.487	23.01	12.19	8.521	-	251
2-F/B	12.43	13.50	11.80	7.099	22.45	11.75	7.985	-	443

nos différents algorithmes de débruitage sur l'exemple Art du jeu de données Middlebury (voir figure 5.4).

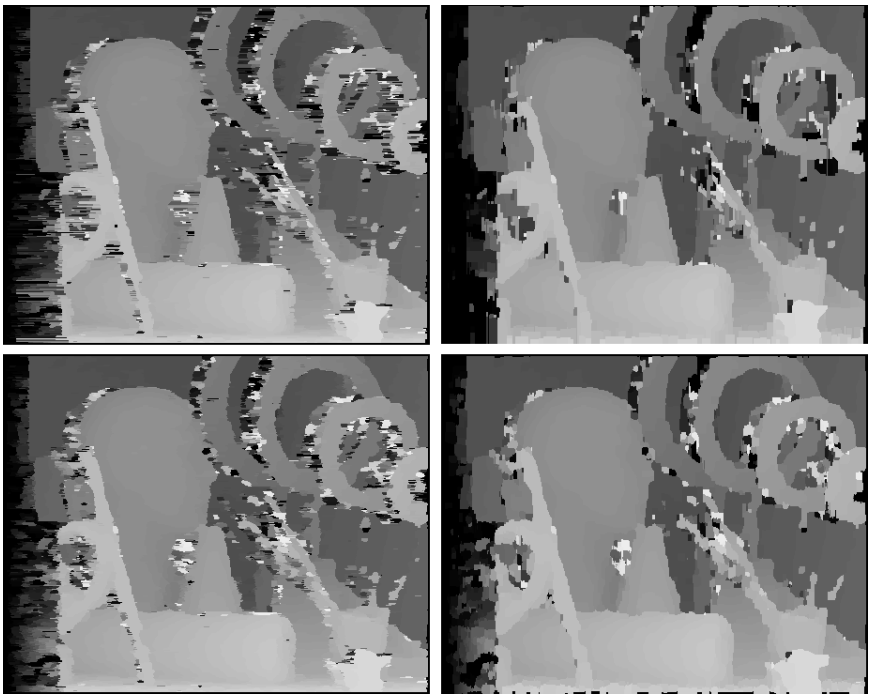

Figure 5.4 Comparaison visuelle des résultats obtenus par chaque algorithme sur l'exemple d'Art, en haut Viterbi (horizontal à gauche et vertical sur horizontale à droite) en bas forward/backward (même disposition).

5.2 Deux dimensions

5.2.1 Modèle

Nous avons présenté a la section précédente une solution prenant en compte à la fois la cohérence verticale et horizontale de notre carte de disparité utilisant deux MMC. Cette façon de procéder n'est pas rigoureuse, elle considère ces cohérences unidimensionnelles à tour de rôle et non pas comme une seule à deux dimensions. Pour s'en convaincre il suffit par exemple d'inverser l'ordre d'utilisation de nos MMC, les résultats sont alors différents. L'utilisation de l'algorithme du forward/backward par contre, offre la possibilité de travailler directement sur les probabilités. Toutefois, il y a quelques précautions à prendre et la combinaison correcte de ces probabilités n'est pas si évidente.

La figure 5.5 montre le modèle probabilistique que l'on souhaite utiliser sur chacune de nos

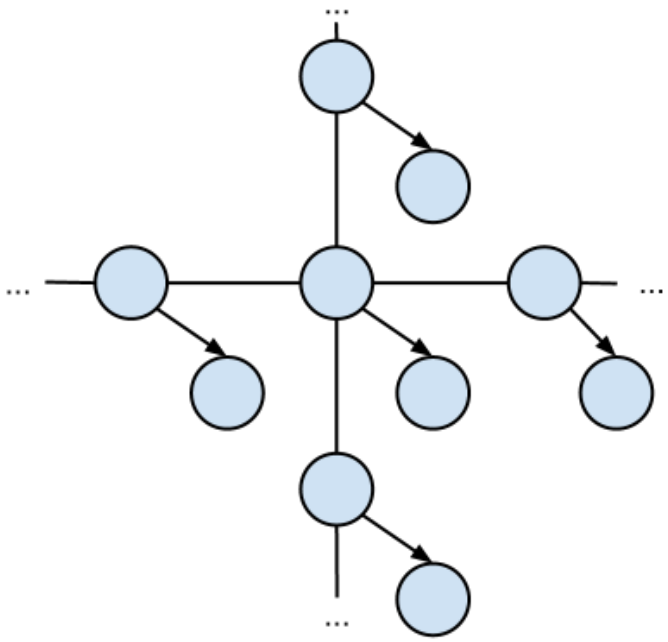

Figure 5.5 Schéma bidirectionel centré sur notre variable d'intérêt.

variables. Il est important de bien considérer que l'on souhaite utiliser un schéma de ce type centré sur chacune de nos variables et non pas faire de l'inférence pour l'ensemble de nos variables sur ce modèle en croix ou sur un modèle beaucoup plus complet (maillage). On veut donc avoir un modèle différent pour chaque variable, de telles approches sont plus souvent réservées à la programmation dynamique comme nous l'avons mentionné dans la revue de littérature. Le problème d'une telle approche dans notre cas vient du grand nombre de calculs que cela nécessiterait pour décoder chacun de nos modèles.

5.2.2 Réutilisation de calculs

En partant de ce constat, nous avons cherché à trouver un moyen de réutiliser les calculs afin de pouvoir utiliser de tels modèles tout en ne pénalisant pas trop les temps de calcul. Nous avons vu que l'algorithme de forward/backward permet de calculer les probabilités. Nous avons donc souhaité décomposer notre modèle en deux chaines unidirectionnelles de façon similaire aux approches précédentes, mais cette fois-ci en considérant les résultats obtenus comme des résultats intermédiaires, et non plus comme nos probabilités finales.
Lorsque l'on raisonne sur des graphes complexes il peut être judicieux d'utiliser un graphe des cliques ; il s'agit d'une version ou les variables dépendantes sont regroupées afin de bien dégager les indépendances et de permettre une factorisation des calculs. Dans notre cas le modèle est relativement simple et le graphe des cliques sert principalement à confirmer les indépendances et à bien mettre en évidence une décomposition triviale (voir Figure 5.6). En effet, on voit assez facilement comment séparer notre modèle en deux chaines, ce qui facilite les calculs et surtout permet de les réutiliser facilement.
Il faut bien sûr considérer que nous avons un graphe de ce type par variable cachée. Avec

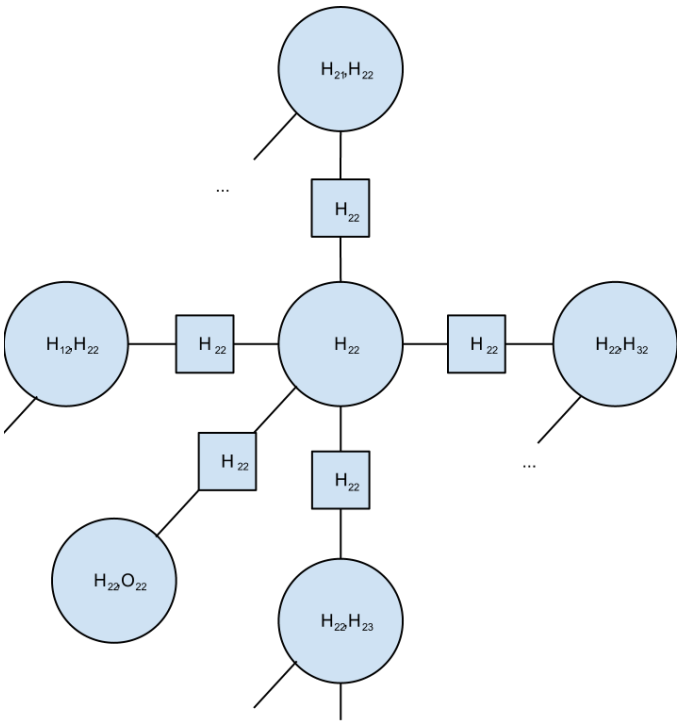

Figure 5.6 Graphe des cliques centré sur une de nos variables cachées.

H_{ij} nos variables cachées sur la colonne i et la ligne j de notre image. O représentant les

observations correspondantes.

Par ailleurs, ce graphe met aussi en évidence un point qu'il est important de bien considérer : le comptage double (double counting). Une version naïve de la séparation de nos calculs viserait en effet à faire les calculs pour un MMC horizontal et un vertical puis de multiplier nos probabilités pour obtenir nos probabilités finales. Il faut cependant considérer que dans notre modèle en croix nous avons :

$$P(\{o^h\},\{d^h\},\{o^v\},\{d^v\}) = \frac{\prod_{i\neq k}\Psi_h(d^h_{i-1},d^h_i)P(o^h_i|d^h_i)}{\prod_{i=1}\Phi_h(d^h_i)}.\frac{\prod_{i\neq k}\Psi_v(d^v_{i-1},d^v_i)P(o^v_i|d^v_i)}{\prod_{i=1}\Phi_v(d^v_i)} .\Psi_h(d^h_{k-1},d^h_k).\Psi_v(d^v_{k-1},d^v_k).P(o_k|d_k)$$

avec ici :

$$\frac{\Psi_h(d^h_{i-1},d^h_i)}{\Phi_h(d^h_i)} = P_h(d^h_{i-1},d^h_i)$$
$$\frac{\Psi_v(d^v_{i-1},d^v_i)}{\Phi_v(d^v_i)} = P_v(d^v_{i-1},d^v_i)$$

Ou les exposants h et v désignent les liens horizontaux et verticaux dans le cas ou k représente notre variable à l'intersection (celle pour laquelle nous utilisons vraiment le modèle). Avec P_v, P_h et P, respectivement, nos probabilités apprises de transition verticale, horizontale et notre probabilité d'émission. Si par contre on souhaite obtenir cette probabilité en multipliant naïvement, on a alors :

$$\begin{aligned}
P(\{o^h\},\{d^h\},\{o^v\},\{d^v\}) &\neq P^h(\{o^h\},\{d^h\})P^v(\{o^v\},\{d^v\}) \\
&= \frac{\prod_{i}\Psi_h(d^h_{i-1},d^h_i)P(o^h_i|d^h_i)}{\prod_{i=1}\Phi_h(d^h_i)}.\frac{\prod_{i}\Psi_v(d^v_{i-1},d^v_i)P(o^v_i|d^v_i)}{\prod_{i=1}\Phi_v(d^v_i)} \\
&= \frac{\prod_{i\neq k}\Psi_h(d^h_{i-1},d^h_i)P(o^h_i|d^h_i)}{\prod_{i=1}\Phi_h(d^h_i)}.\Psi_h(d^h_{k-1},d^h_k).P(o_k|d_k) \\
&\quad .\frac{\prod_{i\neq k}\Psi_v(d^v_{i-1},d^v_i)P(o^v_i|d^v_i)}{\prod_{i=1}\Phi_v(d^v_i)}.\Psi_v(d^v_{k-1},d^v_k).P(o_k|d_k) \\
&= \frac{\prod_{i\neq k}\Psi_h(d^h_{i-1},d^h_i)P(o^h_i|d^h_i)}{\prod_{i=1}\Phi_h(d^h_i)}.\frac{\prod_{i\neq k}\Psi_v(d^v_{i-1},d^v_i)P(o^v_i|d^v_i)}{\prod_{i=1}\Phi_v(d^v_i)} \\
&\quad .\Psi_h(d^h_{k-1},d^h_k).\Psi_v(d^v_{k-1},d^v_k).P(o_k|d_k)^2
\end{aligned}$$

L'impacte de cette erreur se fait ressentir non pas dans la comparaison entre différentes valeurs d'émission (variation sur le bruit), mais sur l'importance que l'on donne à nos observations.

On accorde alors moins d'importance qu'initialement souhaité à la valeur d'observation (partie générative) et beaucoup plus aux transitions (partie jointe de notre modèle). En effet, notre valeur de probabilité étant comprise entre 0 et 1, en prendre le carré donne une valeur inférieure à la valeur souhaitée. On aurait donc trop tendance à suivre la probabilité de transition. La source de ce problème peut se comprendre intuitivement en considérant la figure 5.7.

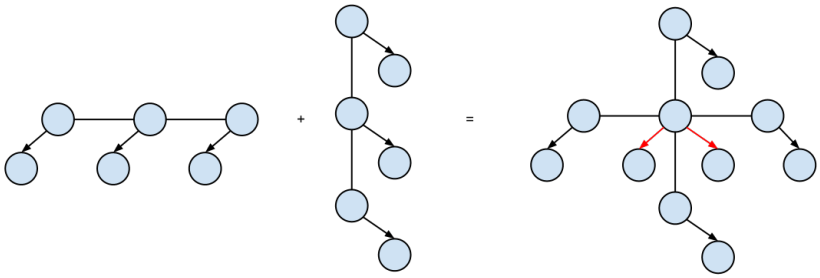

Figure 5.7 Illustration du comptage double.

5.2.3 Implémentation

Nous avons évoqué précédemment l'implémentation de l'algorithme du forward/backward en GPGPU. L'implémentation de ce modèle repose intégralement sur cet algorithme. Dans un premier temps, nous calculons deux tenseurs tridimensionnels contenant pour chaque position dans la carte de disparités, la distribution de probabilité pour les variables cachées dans les chaines horizontales et verticales. Un nouveau noyau permet alors de combiner chacune de ces distributions en enlevant le double comptage et de rechercher le maximum de probabilité. Ce maximum est alors assigné dans un nouveau tampon mémoire bidimensionnel qui contient, une fois l'exécution terminée, notre carte de disparités débruitée.

5.3 Coûts

5.3.1 Théorie

Notre utilisation des coûts lors de la construction d'une carte de disparités bruitée pose problème. D'abord, comme nous l'avons vu dans le chapitre trois les coûts peuvent être définis de multiple façons et aucune n'est optimale. Par ailleurs, il n'est pas rare lorsque notre meilleure disparité, celle minimisant le coût, n'est pas correcte que la vraie disparité ait un

coût seulement très légèrement supérieur, sans forcément être dans le voisinage de la valeur erronée. Il est possible d'avoir plusieurs correspondances de qualité similaire et choisir celle de coût minimal n'est pas forcément la meilleure solution. Une fois que nous avons produit notre carte de disparités, cette information est perdue et notre deuxième meilleure possibilité est irretrouvable. Enfin, nos modèles ne prennent jamais en considération la valeur du coût, que ce soit pour modifier la probabilité d'émission ou l'importance qu'on lui accorde.
Plusieurs approches, mentionnées dans la revue de littérature, essaient de résoudre ces problèmes en utilisant des techniques de programmation dynamique pour agglomérer les coûts sur un voisinage de façon empirique. Il est encourageant de voir que ces approches utilisant un descripteur plus global , au lieu d'un simple coût, améliorent grandement les résultats.
De façon similaire, notre comportement résulte du fait qu'au lieu de considérer nos observations comme étant l'ensemble des lignes de chaque image, nous considérons juste le résultat des comparaisons. Si par contre nous changeons légèrement notre modèle nous pouvons alors raisonner sur l'ensemble de ligne correspondante (voir figure 5.8).
Cependant, les approches mentionnées précédemment se basent sur des modèles empiriques

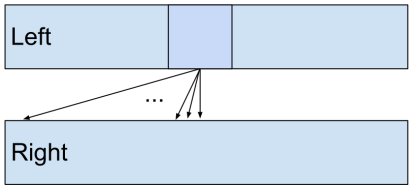

Figure 5.8 Observation des lignes correspondantes et découpage en patch pour faire les calculs (un point de la vu de gauche est à une position précédente dans la vue de droite).

et un recours à la programmation dynamique qui n'est justifiée qu'a posteriori. En partant de ces constats, nous avons souhaité construire un descripteur probabiliste local, reposant sur les lignes, pour remplacer le coût et utiliser un modèle d'inférence sur l'ensemble de ces descripteurs afin de pouvoir construire une carte de disparités de meilleure qualité ne souffrant pas des problèmes mentionnés plus haut. Cela revient donc dans la pratique à modifier notre modèle probabiliste pour l'un des deux présenté en figure 5.9. Ces deux modèles diffèrent sur le lien entre la variable cachée et nos observations de référence (ligne de gauche, L) qui deviennent alors une donnée à part (non-indépendante). Dans la pratique ce modèle (à gauche) reflète plus la réalité, mais peut être un peu plus confus. Par ailleurs, dans notre cas nous verrons qu'il est facile de voir l'équivalence.
Dans la suite nous utiliserons une notation vectorielle pour la comparaison de nos patchs d'images. Afin que la modélisation en variable vectorielle de nos images soit bien claire nous

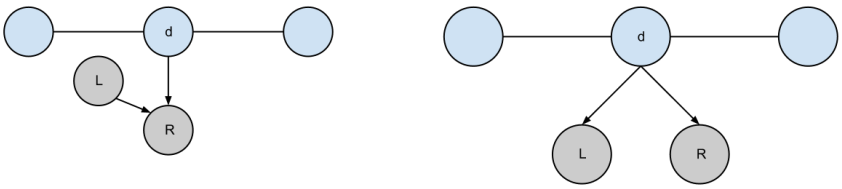

Figure 5.9 Modèles probabilistes possibles pour prendre en compte les bonnes observations (L pour la ligne de gauche, R celle de droite et d pour la disparité cachée).

illustrons la transformation dans le cas d'un bloc de taille trois par trois (voir figure 5.10). Cette transformation nous permet donc d'utiliser naturellement les probabilités sur des variables aléatoires vectorielles.

La littérature, aussi bien que nos expériences, montrent que la combinaison de différentes

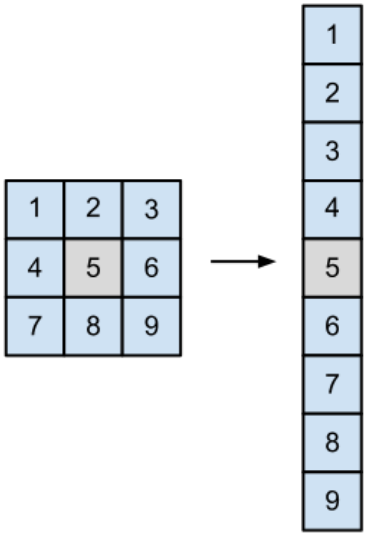

Figure 5.10 Transformation d'un bloc autour d'un pixel d'intérêt (gris) en vecteur.

métriques donne souvent de meilleurs résultats, la correspondance par bloc utilisant la différence au carré et le census par exemple sont de bon complémentaires. Pour les combiner, nous avons d'abord étudié leur répartition, les graphes présentés dans la figure 5.11 illustrent l'allure générale des coûts autour de la vraie valeur de disparité. Au vu de l'allure des répartitions des coûts, nous avons trouvé judicieux de modéliser les coûts comme l'ajout d'un bruit gaussien dans le cadre de la différence des carrés et comme un bruit suivant une loi binomiale pour le census qui est une variable discrète :

$$f_G(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-||x-\mu||^2}{2\sigma^2}}$$

$$\begin{aligned} f_B(x,p) &= \binom{n}{x} p^x (1-p)^{n-x} \\ &= \binom{n}{x} e^{ln(\frac{p}{1-p})x + ln(1-p)} \end{aligned}$$

Nous présentons ici la forme issue de la famille exponentielle de notre distribution afin d'illustrer les similitudes existantes entre nos deux densités. Par ailleurs n représente le nombre de tirage, dans notre cas il s'agit du nombre de bits total à comparrer, soit le nombre de bits par descripteur multiplié par le nombre de descripteurs (soit $n = 9 * 9 * 7 * (2 * 3 + 1) = 3969$ dans la configuration mentionnée au chapitre précédent).
Il est également intéressant de noter que nous approximons ici nos distributions grossièrement et que, plutôt qu'une gaussienne et une binomiale, il pourrait être avantageux d'envisager une mixture de deux distributions dans chacun des cas. Nous reviendrons sur cette idée dans la section travaux futurs de ce mémoire.
Où f_G représente la densité de probabilité d'un bruit Gaussien de moyenne μ et variance σ

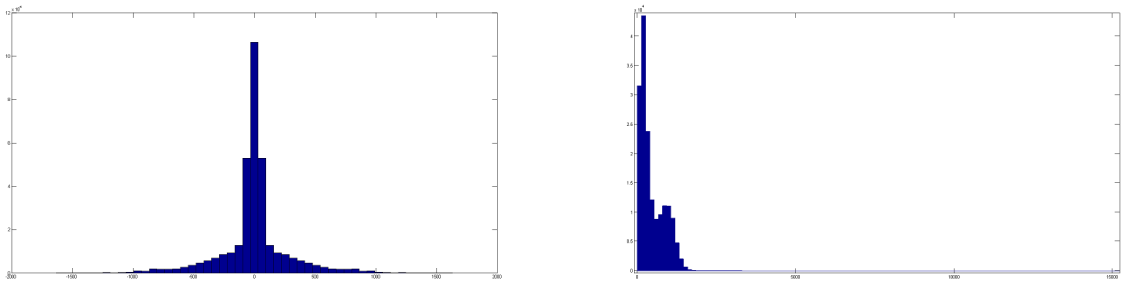

Figure 5.11 Répartition des coûts pour les valeurs des vraies disparités (calculé sur le dataset Middlebury), on observe un comportement gaussien pour les coûts de la correspondance par bloque (à gauche) et binomial pour la correspondance du Census (à droite).

et f_B la probabilité d'obtenir x en tirant une variable suivant une loi binomiale de probabilité p. Nous devons donc calculer les paramètres de ces fonctions avant de pouvoir transformer nos vecteurs de coûts en densité de probabilités. $||x - \mu||^2$ correspond déjà à notre coût dans le cadre de la correspondance par bloc. Il nous faut donc calculer σ, pour ce faire on utilise des cartes de vérité et des disparités de scènes similaires pour mesurer la variance des coûts pour la vraie disparité :

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=0}^{n} ||x_i - \mu_i||^2$$

À ce stade nous sommes donc en mesure d'expliquer la différence entre nos modèles. Dans le premier, nous considérons la probabilité :

$$P(X^R|\tilde{X^L}, d) = \mathcal{N}(X^R, \tilde{X^L}, \sigma)$$

Dans le second cas, nous considérons plutôt :

$$P(X^R - X^L|d) = \mathcal{N}(X^R - X^L, 0, \sigma)$$

Ces deux probabilités sont ici équivalentes et les deux modèles illustrent donc réellement les liens existants entre nos variables. Il est, toutefois, important de bien noter que si le lien aux observations change, la théorie présentée précédemment pour faire de l'inférence dans nos réseaux reste inchangée. Le même procédé nous permet de mesurer les variations existantes entre deux descripteurs de census et ainsi de calculer notre probabilité binomiale :

$$\hat{p} = \frac{1}{n} \sum_{i=0}^{n} |x_i - \mu_i|$$

Ou dans ce cas $|x - \mu|$ mesure la différence normalisée entre nos descripteurs census.
On peut alors transformer nos vecteurs de coûts en une densité de probabilité, on peut ensuite utiliser notre modèle d'inférences précédent en remplaçant la modélisation de notre probabilité d'émission pour chaque variable par la probabilité que nous venons de créer. Au lieu de paramétrer une probabilité d'émission a posteriori nous utilisons alors directement cette probabilité pour faire notre inférence. Nos observations ne sont alors plus l'ensemble des meilleures valeurs de disparités possibles, mais bien l'intégralité des coûts (pour toutes les valeurs de disparités possibles), notre probabilité d'émission devient alors la probabilité d'obtenir le meilleur coût sachant une disparité :

$$P_E(Cout|H) = f_G(SD(H), 0, \hat{\sigma}) f_B(C(H), \hat{p})$$

Avec $SD(x)$ le carré de la différence de nos blocs et $C(x)$ est le coût obtenu, dans le cas du census, pour la valeur de disparité x.
Afin de bien comprendre ce principe de tenseur de probabilité, tout en illustrant l'utilisation de lignes correspondantes à travers les prises de vue en tant qu'observation, nous présentons sur la figure 5.12 une visualisation d'une matrice de coûts extraite par cette approche, de deux lignes correspondantes dans nos images stéréoscopiques et la disparité reconstruite en prenant le maximum de probabilité. L'empilement de ces matrices pour chacunes des lignes nous donne notre tenseur.

Il est également intéressant de noter que dans cette matrice nous avons un triangle où il nous est impossible de calculer des probabilités (les valeurs sous la diagonale), en effet ceci correspond à des positions qui sortent du support de l'image, on ne peut obtenir ces disparités en utilisant juste les images fournies.
Une fois la probabilité d'émission obtenue pour l'ensemble des disparités nous pouvons donc

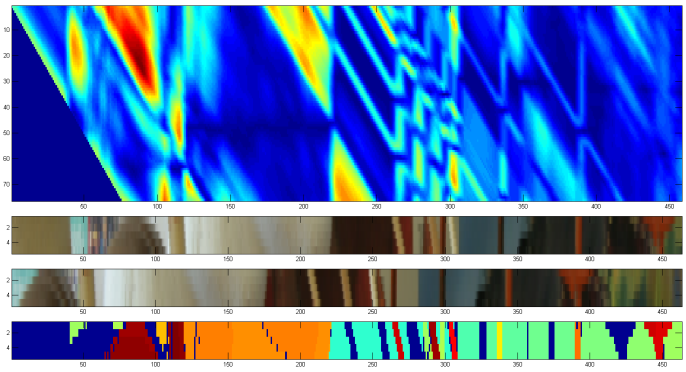

Figure 5.12 Coûts fonctions de la position et la disparité dans le cas de blocs de rayon 5. Les coûts (en haut, bleu étant le moindre coût) pour les deux bandes d'images (au centre) et la meilleure disparité en résultant(en bas).

obtenir la probabilité générale de notre modèle :

$$
\begin{aligned}
P(\{o^h\},\{d^h\},\{o^v\},\{d^v\}) &= \frac{\prod_{i\neq k}\Psi_h(d^h_{i-1},d^h_i)P(o^h_i|d^h_i)}{\prod_{i=1}\Phi_h(d^h_i)}.\frac{\prod_{i\neq k}\Psi_v(d^v_{i-1},d^v_i)P(o^v_i|d^v_i)}{\prod_{i=1}\Phi_v(d^v_i)} \\
&\quad .\Psi_h(d^h_{k-1},d^h_k).\Psi_v(d^v_{k-1},d^v_k).P(o_k|d_k) \\
&= \prod_{i\neq k}P^h_T(d^h_{i-1},d^h_i).P_E(o^h_i|d^h_i).\prod_{i\neq k}P^v_T(d^v_{i-1},d^v_i).P_E(o^v_i|d^v_i) \\
&\quad P^h_T(d^h_{k-1},d^h_k).P^v_T(d^v_{k-1},d^v_k).P_E(o_k|d_k)
\end{aligned}
$$

Ou P^h_T et P^v_T désignent, respectivement, les probabilités de transition horizontale et verticale et P_E la probabilité d'emission décrite plus haut.

5.3.2 Implémentation

Au vu de l'ensemble des changements présentés ici, il serait légitime de penser que nous devons apporter un grand nombre de modifications à l'implémentation de notre modèle probabiliste, ne serait-ce que pour faire de l'inférence a priori (sur les coûts) et non plus a posteriori (sur les disparités). Nous n'avons plus de variable observée scalaire, mais des vecteurs. Dans la pratique il n'y a en fait que peu de modifications à apporter.
Un détail de peu d'importance dans les implémentations précédentes, réside dans la façon dont nous avons modélisé les probabilités d'émission et de transition. Pour chacune de ces probabilités, nous passons à notre algorithme des matrices suivant la paramétrisation. La

matrice de transition contient ainsi la probabilité de passer de la valeur correspondante à la ligne à celle correspondante à la colonne. De même, la matrice d'émission contient la probabilité dans l'état correspondant à notre numéro de ligne de la matrice, d'observer l'état correspondant à la colonne. Si l'on n'avait pas de bruit dans nos observations cette matrice serait l'identité.
Lorsque nous faisons de l'inférence, nous raisonnons sur les colonnes : sachant que nous observons la valeur correspondante à la colonne, qu'elle est la probabilité que notre variable cachée corresponde à une des lignes ?
Cette modélisation classique est couteuse lorsque l'on considère la probabilité d'émission comme étant paramétrée par trois valeurs comme nous le faisions jusqu'ici : toutes les valeurs au-delà de la surdiagonale ou en deçà de la sous-diagonale sont identiques de même pour toutes les valeurs sur la diagonale ou sur les sur et sous-diagonale. Nous avions choisi d'utiliser ce modèle de matrice principalement dans le but de garder une implémentation générale et de pouvoir dans un second temps faire une recherche sur les paramétrisations.
Mais dans cette nouvelle modélisation, nous avons la possibilité d'exploiter cette matrice pour faire les calculs que nous souhaitons sans rien avoir à changer. En effet, si cette matrice ne représente pas la probabilité d'émission sur chaque ligne, mais que nous considérons chaque colonne comme étant notre probabilité pour une variable cachée sachant un vecteur de coûts, cette matrice devient alors identique au tenseur contenant l'ensemble des probabilités et il nous suffit de considérer chaque observation comme le numéro d'un vecteur de coûts afin de pouvoir continuer à utiliser exactement le même algorithme de forward/backward. Chaque colonne de la matrice d'émission n'est alors utilisée qu'une fois et correspond pour schématiser à une probabilité d'émission par variable cachée.
La version GPGPU de cet algorithme est donc essentiellement la concaténation des noyaux précédents et d'un noyau transformant le tenseur de coûts en tenseur de probabilités. Il faut cependant, dans ce but, ajouter un précalcul des paramètres σ et p, dans le cadre du jeu de données Middlebury nous calculons ces paramètres pour un couple d'images stéréo en utilisant tous les autres (leave one out). Comme mentionné au chapitre précédent, les noyaux sont déjà conçus de façon à pouvoir obtenir très facilement le tenseur des coûts pour chacune des variables. Nous avons donc également besoin d'ajouter un noyau qui prend en entrées ces coûts, les paramètres σ et p et calcul les probabilités correspondantes et les enregistre dans un nouveau tenseur. Ce noyau est extrêmement intuitif, la parallélisation est évidente, et ne peut pas être optimisé d'un point de vue mémoire, nous lisons deux valeurs en mémoire globale pour les combiner et enregistrer le résultat également en mémoire globale. Une fois ceci fait, nous avons l'ensemble des probabilités qui nous sont nécessaires pour faire de l'inférence.
Nous appelons alors une première fois nos noyaux de forward et de backward, en passant

notre tenseur comme matrice d'émission (matrice modélisant la probabilité d'émission), pour calculer les log-probabilités horizontales. Un deuxième appel à ces noyaux nous donne les log-probabilités verticales. Nous utilisons alors un dernier noyau prenant en entrée les log-probabilités dans les deux directions ainsi que les probabilités d'émission. Ce noyau ajoute alors les log-probabilités et soustrait le logarithme de la probabilité d'émission correspondante (en espace logarithmique une multiplication est transformée en somme et une division en soustraction). Nous obtenons ainsi les log-probabilités finales du modèle pour chacune des variables et pouvons alors faire une recherche de maximum. L'$argmax$ de cette recherche nous donne alors la carte de disparités débruitée.

5.3.3 Résultats

Nous pouvons utiliser de nouveau le jeu de données de Middlebury pour tester les performances de notre modèle. À fin de comparaison, nous avons également calculé un ensemble de résultats intermédiaires que nous présentons dans le tableau 6.1. Dans un premier temps, nous présentons les résultats obtenus en se contentant de la combinaison de nos coûts et en cherchant le coût minimal (BM-Census). On constate que cette combinaison améliore déjà la qualité de la carte de disparité ainsi produite . Nous présentons également les résultats obtenus en utilisant des MMC et l'algorithme de forward/backward de façon similaire à la section précédente (MMC et 2-MMC), mais en utilisant cette fois-ci la carte de disparité produite en utilisant les meilleurs de ces nouveaux coûts. Enfin, nous présentons les résultats de notre modèle (Étoile) utilisant l'intégralité des coûts et disparités possibles.
Ces résultats illustrent clairement la forte différence qui existe entre le raisonnement probabiliste a posteriori et notre approche. L'utilisation de notre modèle prenant en compte à la fois la cohérence spatiale bidimensionnelle et l'ensemble des coûts au lieu de se limiter seulement aux disparités. En effet, les résultats de 2-MMC correspondent à l'utilisation du même modèle, mais appliqué aux disparités (posteriori) là ou Étoile est utilisé à priori ce qui nous donne un gain de qualité de l'ordre de 8%.

Dans la pratique, il est difficile de juger de la qualité d'une carte de disparités en ne se basant que sur des observations. Nous fournissons tout de même à titre indicatif les résultats obtenus sur l'exemple Art du jeu de données Middlebury (voir figure 5.13). On constate toutefois la présence d'artefacts similaires sur les cartes utilisant des modèles probabilistes.

Tableau 5.2 Comparaisons de taux d'erreur des pixels visibles en pourcentage (%) des différentes combinaisons de méthodes

Méthode	μ	*Arts*	*Books*	*Dolls*	*Laundry*	*Moebius*	*Reindeer*	GPU time (ms)
BM	22.50	27.00	19.39	15.14	34.12	20.61	18.72	9.9
Census	13.95	15.44	13.16	8.033	24.94	12.78	9.341	30
BM-Census	12.74	13.00	12.63	6.733	23.72	12.18	8.502	54
MMC	11.85	12.67	11.85	6.329	21.06	11.76	7.420	277
2-MMC	11.26	11.92	10.17	6.065	20.56	11.55	7.345	467
Étoile	10.36	10.40	9.743	5.207	19.04	11.07	6.744	496

5.3.4 Discussion

Comme le montre la section précédente, cette approche de combinaison de coûts fournit de très bons résultats, mais il est également intéressant de noter que cette approche offre une très bonne modélisation des occlusions. En effet, la figure 5.14 montre les régions de notre image ou nous obtenons une faible probabilité pour toutes les valeurs de disparités possibles, la correspondance avec les régions occludées, calculées par cross corrélation de nos bases de vérité, de notre image est marquante. L'application de notre modèle aux données préserve cette propriété, mais nous n'utilisons pas encore cette information et cela pourrait être une bonne piste pour une éventuelle amélioration du modèle en attribuant la disparité en se basant sur de super pixels par exemple.

Ce chapitre souligne également l'importante différence existant entre la superposition de deux modèles unidirectionnels et l'utilisation d'un réel modèle bidirectionnel. On illustre, enfin, le principe de traitement a priori, directement sur les coûts et les correspondances, et son efficacité en comparaison des modèles a posteriori, raisonnants eux sur les disparités.

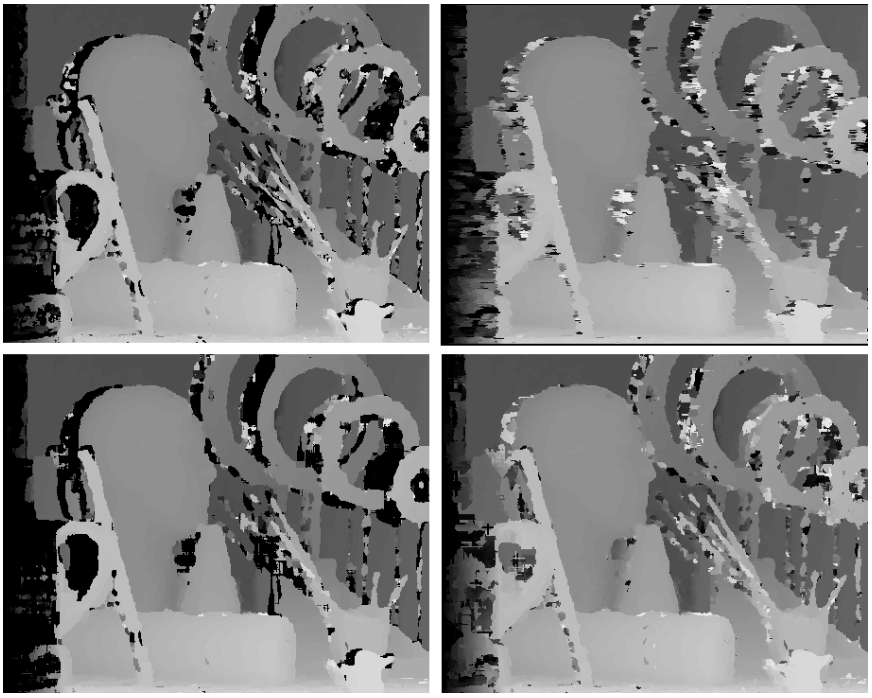

Figure 5.13 Comparaison des différents résultats obtenus à ce stade, la combinaison brute des coûts (haut gauche), utilisation du forward-backward horizontalement puis verticalement (haut droit et bas gauche) et enfin notre modèle en étoile (bas droit).

Figure 5.14 Détection des régions occludées (mise en évidence à gauche par validation croisée) par le modèle bi-dimensionnel(à droite).

CHAPITRE 6

Combinaison et Application

6.1 Introduction

Dans ce chapitre, nous présentons brièvement le principe de fonctionnement de la Kinect, permettant d'acquérir à faible coût et en temps réel des données tridimensionnelles. Nous évoquerons également brièvement les stratégies disponibles pour pouvoir la combiner efficacement avec d'autres systèmes de capture. Nous reviendrons également sur les possibilités de combinaisons des travaux présentés précédemment afin de réaliser des captures de bonne qualité dans le cadre de la numérisation de visage.

6.2 Kinect

En novembre 2010, Microsoft commercialisait la Kinect comme un périphérique de jeu pour sa console la xbox 360. La kinect comprend un système optique ainsi qu'un système acoustique permettant aux joueurs d'interagir naturellement avec leurs consoles (voir figure 6.1. Si la partie audio n'est pas révolutionnaire, la partie vidéo était une première en terme de capture 3D en temps réel pour un coût aussi minime. C'est cet aspect qui a très vite enthousiasmé le monde de la recherche qui s'est affairé à rendre le périphérique aussi accessible que possible.

Le principe de fonctionnement du périphérique n'est pas très éloigné de ceux présentés

Figure 6.1 Descriptif de la kinect.

dans ce mémoire dans la partie sur les méthodes actives. Un projecteur projette un schéma

pseudoaléatoire (voir figure 6.2) tandis qu'une caméra en capture les déformations. Ces déformations peuvent alors être transformées en profondeurs. Tout ceci se fait dans le domaine de l'infrarouge afin de ne pas altérer la scène ou l'expérience du joueur. Une autre caméra vient compléter le montage pour capturer les couleurs de la scène. Nos captures peuvent alors être combinées pour obtenir un rendu 3D en couleur de notre scène.

Cependant, la Kinect souffre de plusieurs défauts principalement dus à la contrainte bud-

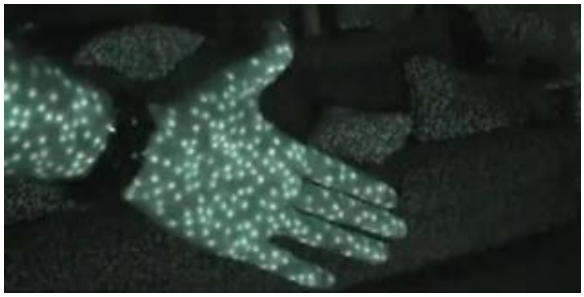

Figure 6.2 Schéma Infrarouge pseudoaléatoire projeté par la kinect (extrait de http ://www.youtube.com/watch ?feature=player_embedded&v=nvvQJxgykcU).

gétaire. La calibration, les lentilles et les capteurs sont de qualités somme toute moyennes ce qui impacte aussi bien la carte de profondeur que les images capturées ou la mise en correspondance des deux (voir figure 6.3). Il y a par ailleurs des défauts de stabilité et des variations dans la profondeur mesurée pour un même point au cours du temps. Enfin, on retrouve les défauts liés aux méthodes projectives : limites de profondeur ainsi que l'impacte des réflexions et occlusions.

La kinect offre toutefois l'avantage de fournir une carte de profondeurs de qualité suffisante

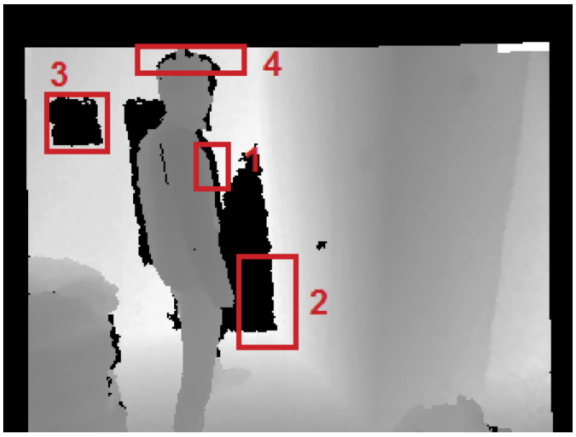

Figure 6.3 Illustration des défauts liés à la capture par une kinect : occlusion (1) distance limitée (2) réflexion (3) et zones trop petites (4).

pour le jeu vidéo à 30 images par secondes. Il peut être intéressant de l'utiliser tout en considérant bien qu'il ne s'agit que de données de faible qualité. Nous proposons dans ce chapitre trois façons de l'exploiter conjointement avec notre système de capture.

6.3 Schéma expérimental

Dans le but de réaliser notre propre jeu de captures, afin de pouvoir essayer divers modèles sur différentes situations, nous avons dû établir un schéma expérimental nous permettant d'obtenir toutes les données nécessaires et de pouvoir les combiner et les comparer. Nous revenons dans cette section sur les différentes contraintes que cela implique et détaillons le schéma expérimental auquel nous avons abouti.

6.3.1 Références

Comme nous l'avons mentionné précédemment, il est primordial que notre capture de données nous permette d'obtenir des références ; à savoir, une base de vérité ainsi qu'une valeur de base à laquelle nous pouvons nous comparer. Notre modèle probabiliste final décrit dans le chapitre précédent repose sur des captures stéréo standard, obtenir une référence de comparaison ne requiert donc pas d'acquisition supplémentaire. Pour l'obtention de la base de vérité nous avons cependant besoin d'utiliser une numérisation par lumière structurée qui vient se rajouter à notre capture.

6.3.2 Modélisation

Notre modèle repose sur une modélisation probabiliste de notre scène. Il nous est bien sûr possible d'en fixer une arbitrairement, mais une telle approche devra être adaptée pour chaque changement de scène et chercher une bonne modélisation arbitraire serait fastidieux. Une première possibilité serait d'utiliser la base de vérité capturée dans une situation similaire, pour une capture précédente de la même scène par exemple, mais cela requiert d'acquérir la base de vérité pour chaque légère modification de notre scène et nous pourrions alors utiliser directement la carte de vérité. Une seconde idée serait alors de construire notre modèle sur une première capture en utilisant une base de vérité puis de l'utiliser pour les captures suivantes d'une même scène, cette approche est dite du transfert d'apprentissage. Cette idée est valide même si elle requiert un temps de capture plus long pour notre configuration initiale. Par ailleurs, notre modèle risque d'être de moins en moins adapté au fur et à mesure

des déformations subies par notre scène. Nous pouvons alors projeter de temps en temps une texture comme discuté dans le chapitre trois. Cette texture nous fait perdre une trame, mais nous permet d'obtenir une modélisation de bonne qualité qui peut nous servir à améliorer les paramètres de notre modèle.

6.3.3 Ajout d'une Kinect

Une autre solution pourrait être d'utiliser une Kinect. La Kinect donne une carte de disparité de faible qualité, mais pour modéliser notre scène cela pourrait permettre d'obtenir une modélisation évitant le surapprentissage. Une bonne paramétrisation pourrait en effet utiliser une Kinect pour modéliser nos transitions.
Mais, à condition d'être calibré avec nos caméras stéréoscopiques afin de pouvoir tout projeter dans le même repère, notre Kinect pourrait aussi nous servir de repère de comparaison. Une autre application serait de l'utiliser pour restreindre notre champ de recherche de correspondance, pour accélérer les calculs, autour de la valeur de disparité indiquée par la kinect. L'idée est alors d'utiliser la calibration pour transformer la profondeur de la kinect en une profondeur pour chacune de nos caméras. Puis d'utiliser cette distance et les matrices intrinsèques (également obtenues par calibration) pour déterminer la position de chaque point sur nos capteurs. La différence de position nous donne alors une bonne idée de disparité autour de laquelle restreindre nos calculs.
Enfin une dernière utilisation potentielle de notre Kinect pourrait être de se rajouter comme observation dans notre modèle, on viendrait ainsi pondérer notre probabilité d'émission en fonction de la valeur observée par la Kinect. Cette dernière approche ressemble beaucoup à la précédente. Elle varie cependant un peu dans la mesure où le champ des valeurs possibles n'est alors pas restreint, mais juste pondéré, on accepte alors d'envisager que la Kinect puisse être complètement erronée.

6.3.4 Modèle expérimental final

Afin de tester et d'évaluer nos différentes stratégies, nous avons donc dû établir un modèle expérimental. Pour ce faire, nous avons dans un premier temps renoncé à considérer des scènes en mouvement. Notre schéma expérimental se décompose donc en deux phases que l'on va répéter autant de fois que voulu : capture de notre scène statique puis déplacement suivit d'une nouvelle capture, etc. Notre capture est composée de trois étapes : l'acquisition de la base de vérité en utilisant une approche par lumière structurée avec une caméra stéréosco-

pique, l'acquisition de données stéréoscopiques actives et enfin l'acquisition passive incluant une capture par une Kinect.
Dans le cadre d'un éventuel projet avec une compagnie nous nous sommes intéressés aux visages, pour assurer l'absence de mouvement nous avons envisagé l'utilisation d'un mannequin, cette approche introduit cependant un défaut de texture plus important que celui représenté par la peau. Nous avons donc également numérisé de vrais visages aux travers divers expressions et positions.

6.3.5 Système de capture

Notre montage final pour réaliser nos expériences se compose donc d'une caméra stéréo Sony Handycam qui présente l'avantage de fournir des images stéréo synchronisées rectifiées et alignées, d'une Kinect (Microsoft) et d'un projecteur Lightcommander fabriqué par Texas Instrument. Ce projecteur offre la particularité d'être programmable et d'offrir une excellente stabilité de trame ainsi que sans l'utilisation d'un système de couleur rotatif.

6.4 calibration

L'ensemble caméra-Kinect est préalablement calibré de façon à pouvoir projeter toutes les captures dans un même repère. Le principe consiste à détecter des points clés au travers les différentes vues et à utiliser des connaissances sur ces points pour déterminer les paramètres intrinsèques de nos caméras ainsi que, après mise en correspondance, les paramètres extrinsèques les reliant les unes aux autres. Ces deux matrices peuvent, en bonne première approximation, s'écrire sous la forme :

$$I = \begin{pmatrix} \alpha_x & \gamma & c_x \\ 0 & \alpha_y & c_x \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad E = \begin{pmatrix} & t_x \\ R_\theta & t_y \\ & t_z \end{pmatrix}$$

Où I est la matrice intrinsèque définie en fonction des longueurs focales α sur chacun des axes, du coefficient de distorsion γ et du centre optique sur le capteur c (idéalement le centre coïncide avec le centre du capteur, dans la pratique cela n'est jamais le cas). E la matrice extrinsèque définie en fonction d'une rotation tridimensionnelle, représentée par la matrice R_θ et d'une translation t. Ainsi, une fois ces matrices obtenues, un point en trois dimensions défini dans un repère de caméra peut être projeté pour obtenir la position sur le capteur grâce à notre matrice I ou, en utilisant sa composante homogène (pour la translation), être changé de repère, pour pouvoir être éventuellement projeté sur un autre capteur, par la matrice E.

$$P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \hat{C} = IP = \begin{pmatrix} u \\ v \\ z \end{pmatrix}, C = \begin{pmatrix} \frac{u}{z} \\ \frac{v}{z} \end{pmatrix}$$

$$P_K = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, P_C = E_{KC} P_K = R_\theta \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T$$

Avec P un point projeté à la position C sur notre capteur et P_K un point dans le repère d'une seconde caméra qui peut être transformé en P_C dans le repère de notre première caméra. Cettre transformation est obtenue en utilisant la matrice extrinsèque E_{KC} liant nos deux caméras.

La matrice intrinsèque réalise une projection, on perd donc une dimension et il est impossible sans connaitre la profondeur d'un unique point de la calculer. Calculer les matrices intrinsèques et extrinsèques d'un système de caméras à partir de prise de vue peut donc sembler impossible. Mais si l'on utilise un ensemble de points formant une géométrie connue (la distance entre chaque point est donnée) on peut utiliser cette distance et les déformations subies par nos points lors des captures pour calculer notre matrice intrinsèque.

La matrice extrinsèque comporte douze inconnues, elle projette d'un espace à trois dimensions et une composante homogène dans un espace à trois dimensions il peut sembler plus simple de résoudre ce système à l'aide de seulement quatre points. Mais dans la pratique une fois encore nous n'avons pas la profondeur, mais seulement la projection, la prise de vue de nos points. On a donc une fois encore recours aux déformations subies par un ensemble de points dans nos prises de vue pour déterminer chacun des paramètres de notre matrice extrinsèque. Ces calculs reposent sur de l'optimisation bilinéaire ainsi que sur les calculs effectués précédemment pour obtenir nos matrices intrinsèques. Il est intéressant de noter qu'il est facile d'inverser notre matrice extrinsèque pour avoir le lien réciproque entre nos caméras.

L'algorithme consiste donc à capturer un ensemble de points ayant des positions connues dans l'espace à utiliser leur déformation dans chacune des prises de vue pour estimer les paramètres intrinsèques des deux caméras. Puis d'utiliser ces valeurs intrinsèques et une mise en correspondance des points aux travers les vues pour calculer les paramètres extrinsèques. De nombreuses implémentations optimisées de cet algorithme sont déjà disponibles. Nous effectuons donc dans un premier temps plusieurs captures d'échiquiers (voir figure 6.4). Nous utilisons ensuite la librairie OpenCV pour détecter ces échiquiers et obtenir les matrices intrinsèques et extrinsèques de notre système. Le problème qu'il nous reste cependant à résoudre dans le cadre de la calibration est d'obtenir des images synchronisées. En effet, nous souhaitons utiliser des échiquiers pour réaliser la calibration afin d'avoir un objet facilement

détectable et identifiable. La mise en correspondance aux travers nos différentes prises de vues est alors assez intuitives. Mais ces correspondances ne seront vraies que si notre échiquier ne bouge pas entre nos prises de vues. Or tous nos appareils de capture ne peuvent pas forcément être synchronisés. Le problème vient du fait que notre caméra Sony ne peut pas être contrôlée par ordinateur pour ne prendre qu'une photo stéréoscopique en même temps que notre Kinect. Nous sommes obligés d'enregistrer sur la mémoire interne puis de récupérer nos vidéos ultérieurement.

Nous souhaitons tout de même conserver cette caméra pour les avantages qu'elle offre en

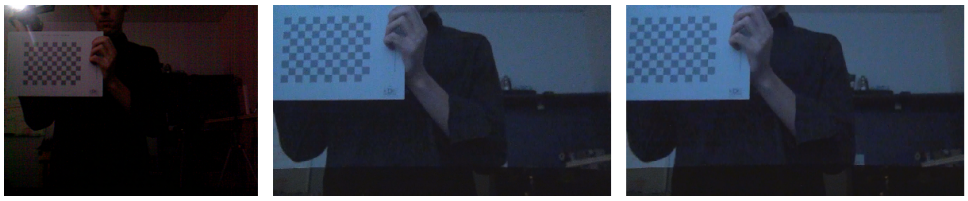

Figure 6.4 Capture d'échiquiers "flashé" par la kinect (à gauche), le capteur stéréo de gauche (au centre) et celui de droite (à droite).

terme de capture stéréoscopique son très bon taux de frame et la résolution des images alignées et rectifiées qu'elle permet d'obtenir. Nous avons donc besoin d'une synchronisation pouvant être capturée. Ne voulant pas rajouter de capture d'une bande-son qui aurait pu être légèrement déphasée avec l'une de nos captures nous avons opté pour une synchronisation visuelle. Après avoir envisagé plusieurs possibilités (entre autres, projection d'un chronomètre dans la zone de capture, ou utilisation d'une tablette avec une horloge et un échiquier visible à l'écran), la solution que nous avons adoptée a finalement été de projeter un flash de lumière sur notre scène pour détecter les trames de nos captures qui correspondent (voir figure 6.4). Une étude de l'histogramme au cours du temps nous permet de détecter les trames d'intérêt.

6.5 Conversion de la kinect

Une fois ce principe de synchronisation trouvé et quelques subtilités inhérentes à l'utilisation d'OpenCV et au fait que les images fournies par la Kinect sont non seulement à l'envers (la Kinect étant bas en haut pour être le plus proche possible de notre caméra stéréoscopique), mais aussi miroir (la droite est à gauche) obtenir nos matrices intrinsèques et extrinsèques ne pose pas de réelle difficulté. Nous sommes alors en mesure de capturer un grand nombre d'informations sur les scènes que nous souhaitons numériser (voir figure 6.5). Dans la suite de cette section nous utiliserons les notations suivantes : C_L indiquera la vu de gauche de

notre caméra Sony, C_R celle de droite, K sera utilisé pour désigner la Kinect, I_X désigne la matrice intrinsèque pour la caméra X (ainsi I_{C_L} représente celle de la vue de gauche), enfin E_{XY} représente la matrice extrinsèque transformant notre repère de caméra X en celui de la caméra Y (par exemple E_{KC_L} pour passer de la Kinect à la vue de gauche). On notera par ailleurs que $E_{XY}.E_{YZ} = E_{XZ}$ cette égalité est assez notable sur nos jeux de données du au fait que notre caméra stéréoscopique produit des images déjà alignées et rectifiées.

L'utilisation de l'ensemble de ces données dans le même référentiel nécessite cependant un

Figure 6.5 Ensemble des données capturées pour une scène numérisée, nos deux vues prisent par la Sony à 60 images par secondes (gauche et centre gauche) et la kinect, à 30 images par secondes, en couleur (centre droit) et en profondeur (droite).

prétraitement des données qu'il est intéressant de détailler. La Kinect nous permet d'obtenir à trente images par seconde une prise de vue ainsi qu'une carte de profondeurs toutes deux de piètre qualité. La carte de profondeurs encode pour chaque pixel la distance séparant l'objet perçu à cette position et la Kinect elle-même. Nous possédons donc, une fois chacune des positions de cette carte multipliées par la matrice E_{KC_L}, respectivement E_{KC_R}, une carte de profondeurs dans le repère de notre vu de gauche, respectivement de droite (voir figure 6.6).

On peut alors projeter ces points sur le capteur de chacune des caméras en utilisant la ma-

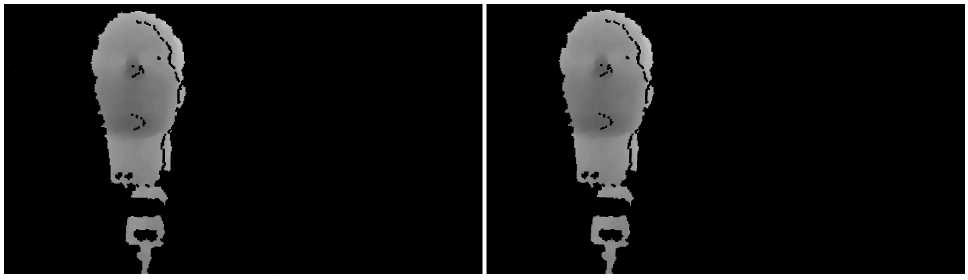

Figure 6.6 Carte de profondeurs de la kinect transformée dans le repère de notre vue de gauche (à gauche) et de droite (à droite).

trice intrinsèque correspondante (I_{C_L} pour la gauche et I_{C_R} pour la droite). On obtient alors les résultats présentés en figure 6.7, nous avons alors la carte de profondeur grossière de la Kinect pour chacune de nos prises de vues de bonne qualité. Cependant, il s'agit de carte de profondeurs et non de carte de disparités et nous ne pouvons pas directement les utiliser

pour comparer nos résultats.

Nous n'allons pas utiliser ces cartes modifiées pour retrouver des correspondances, sur des

Figure 6.7 Cartes de profondeurs projetées dans la vue de gauche (à gauche) et de droite (à droite).

valeurs variant aussi peu et dans un intervalle si restreint les résultats ne seraient probablement pas bons. Mais si l'on remonte notre processus et que l'on considère maintenant pixel par pixel notre carte de profondeur issue de la kinect. Chaque point est en effet envoyé à une position dans la vue de gauche et une autre dans celle de droite. De la différence de positionnement sur l'axe horizontal nous pouvons retrouver notre disparité sans réellement construire la projection de la carte de profondeurs dans chacune de nos vues (voir figure 6.8). Si par contre on ne s'intéresse qu'à la comparaison de nos résultats avec la kinect il est aussi possible de transformer la carte de disparités finale en carte de profondeur en utilisant la matrice $E_{C_L C_R}$ et ainsi comparer directement la profondeur.

La comparaison de la carte de disparités ainsi produite et de celle construite en utilisant

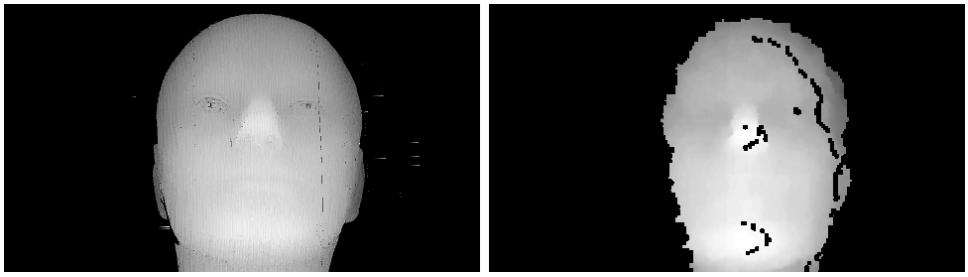

Figure 6.8 Cartes de disparités construite par lumière structurée (à gauche) comparrée à celle extraite de la transformation de la carte de profondeurs de la kinect (à droite).

l'approche de lumière structurée nous permet de juger de la qualité de notre calibration (les régions identiques se superposent). Mais cette superposition illustre surtout les lacunes de la Kinect, avec notamment d'importants défauts sur les bords (une partie du visage est manquant), des zones manquantes et des aplats qui ne devraient pas exister.

6.6 Deux approches

6.6.1 Cohérence temporelle et transfert d'apprentissage

Notre première idée, pour exploiter le montage et le modèle expérimental résultant de nos recherches, était de considérer notre scène au cours du temps et de voir dans quelle mesure notre apprentissage à un temps donné pouvait se transférer au temps d'après.
Pour ce faire, nous avons donc dans un premier temps capturé des scènes statiques (visages immobiles), en prenant à la fois la base de vérité par lumière structurée et les captures stéréoscopiques nécessaires pour construire nos cartes de disparités. Nous avons ensuite appris les paramètres de notre modèle en étoile sur ces données comme indiqué au chapitre précédent. L'étape suivante consiste à numériser une autre pose de notre même scène, également avec la base de vérité, mais cette fois uniquement à des fins de comparaisons et d'évaluations. Nous sommes alors en mesure d'utiliser notre modèle en étoile sur les données obtenues en utilisant les paramètres appris au cours de la numérisation précédente.
Nous présentons en figure 6.9 et 6.10 des exemples de résultats obtenus par cette approche. Un premier constat est que la combinaison de coûts, et plus généralement la détection de correspondance est de moins bonne qualité que sur les données utilisées au chapitre précédent. Cela s'explique par la grande différence existante entre nos scènes d'intérêts ici et les scènes du jeu de données Middlebury. Ici nous ne numérisons qu'un visage en gros plan et nous avons donc beaucoup moins de détails susceptibles d'améliorer les correspondances. La simple mise en correspondance, utilisant la combinaison du census et de la correspondance par bloc et pas de modèle probabiliste nous donne des résultats de l'ordre de 37% (pour les exemples fournis les valeurs sont de 36.23 lorsque σ et p de notre modèle sont appris sur la base de vérité correspondante, 6.9, et 37.13% lorsqu'on utilise les paramètres appris au cours de la numérisation précédente, 6.10).
Sans grande surprise lorsqu'on utilise la base de vérité pour paramétrer notre modèle nous sommes en mesure d'améliorer drastiquement la qualité de nos cartes de disparités. Passant de 37% à un peu moins de 23% d'erreur, soit un gain de l'ordre de 38% (notre erreur passe de 36.23 a 22.93% sur la première numérisation présentée en exemple et de 37.13 à 22.80 sur la seconde).
Mais la paramétrisation ainsi obtenue semble par ailleurs bien se porter au fil du temps. En effet, en utilisant les valeurs obtenues lors de l'apprentissage sur notre première numérisation pour débruiter notre seconde numérisation, nous avons pu améliorer la qualité de nos cartes de disparités pour passer de 37% d'erreur à près de 24% (sur l'exemple numéro deux nous passons ainsi de 37.13% à 24.11% d'erreur).

6.6.2 Stéréoscopie multimodale

Faute de temps nous n'avons pu explorer toutes les combinaisons possibles utilisant la Kinect. Nous nous sommes donc restreints à son utilisation en tant qu'observations additionnelles servant à la paramétrisation de notre matrice de transition. Nous souhaitions également l'utiliser pour déterminer les paramètres nécessaires à la transformation des coûts de notre modèle (σ et p) mais cette approche détériore les résultats et devras être explorée plus en détail ultérieurement.
Une fois la calibration effectuée nous sommes en mesure de transformer les données de la Kinect et d'évaluer la qualité de la carte ainsi produite par rapport à la base de vérité. Nous obtenons alors une erreur de l'ordre de 35% (sur les exemples fournis : 34.01 dans le premier cas et 36.12 dans le second avec d'importants problèmes sur la région des cheveux).
Nous pouvons alors procéder exactement comme expliqué au quatrième chapitre pour établir notre matrice de transition et ensuite appliquer notre modèle. Notre paramétrisation accorde peu d'importance à la valeur en tant que telle pour ce qui est de la transition, nous avons en tout trois paramètres. Aussi, du moment que les données de la Kinect sont cohérentes et dans la mesure où les transitions sont réalistes, l'utiliser pour calculer la matrice de transition semble tout à fait raisonnable. Et, en effet, nous avons pu observer un fort gain de qualité de l'ordre de 30% en passant de 37% à 26% d'erreur (de 36.23% à 25.12% quand σ et p sont appris sur la base de vérité de de 37.13% à 26.18% quand on transfert les paramètres au fil du temps).

6.6.3 Résultats

L'ensemble des résultats obtenus par ces approches est détaillé au fil des explications précédentes, toutefois afin de permettre au lecteur de bien les cerner dans leur ensemble nous présentons ici un tableau récapitulatif. Les lignes indiquent si les paramètres σ et p de notre modèle sont obtenus par utilisation de la base de vérité ou s'ils sont passés d'une numérisation précédente similaire. Chaque colonne indique le pourcentage d'erreur à un pixel pour chacune des approches utilisées. "Meilleur coût" désigne l'approche de combinaison du census et de la correspondance par bloc sans utilisation d'inférence. "Modèle Kinect" désignant l'utilisation de la Kinect pour obtenir la matrice de transition et "Modèle b.d.v." désignant l'utilisation d'une base de vérité pour ce faire (celle de la capture précédente dans le cas du transfert).
À des fins de visualisation du type de cartes de disparités produites nous présentons ici deux figures, présentant chacune des approches dans le cas où nous n'utilisons pas le transfert de paramètres 6.9 et dans le cas où nous y avons recours 6.10.

Tableau 6.1 Récapitulatif des résultats obtenus en utilisant des approches de stéréoscopie multimodale (pourcentage d'erreur de reconstruction à un pixel)

σ p	kinect	Meilleur coût	Modèle kinect	Modèle b.d.v.
base de vérité	35.06	36.23	25.12	22.87
transfert	35.06	37.13	26.18	24.11

6.7 Discussion

Ce chapitre illustre à la fois le bon fonctionnement de notre modèle même lorsque les correspondances sont moins bonnes, dû à la complexité de nos numérisations et au faible texturage. Mais il illustre aussi le bon fonctionnement du transfert d'apprentissage appliqué à notre modèle et le fait qu'il est possible de se contenter d'évaluation de nos paramètres seulement une fois par type de scènes numérisées. Enfin, ce chapitre pose les bases de l'introduction d'une kinect dans notre système de capture et illustre une bonne utilisation qu'il est possible d'en faire.

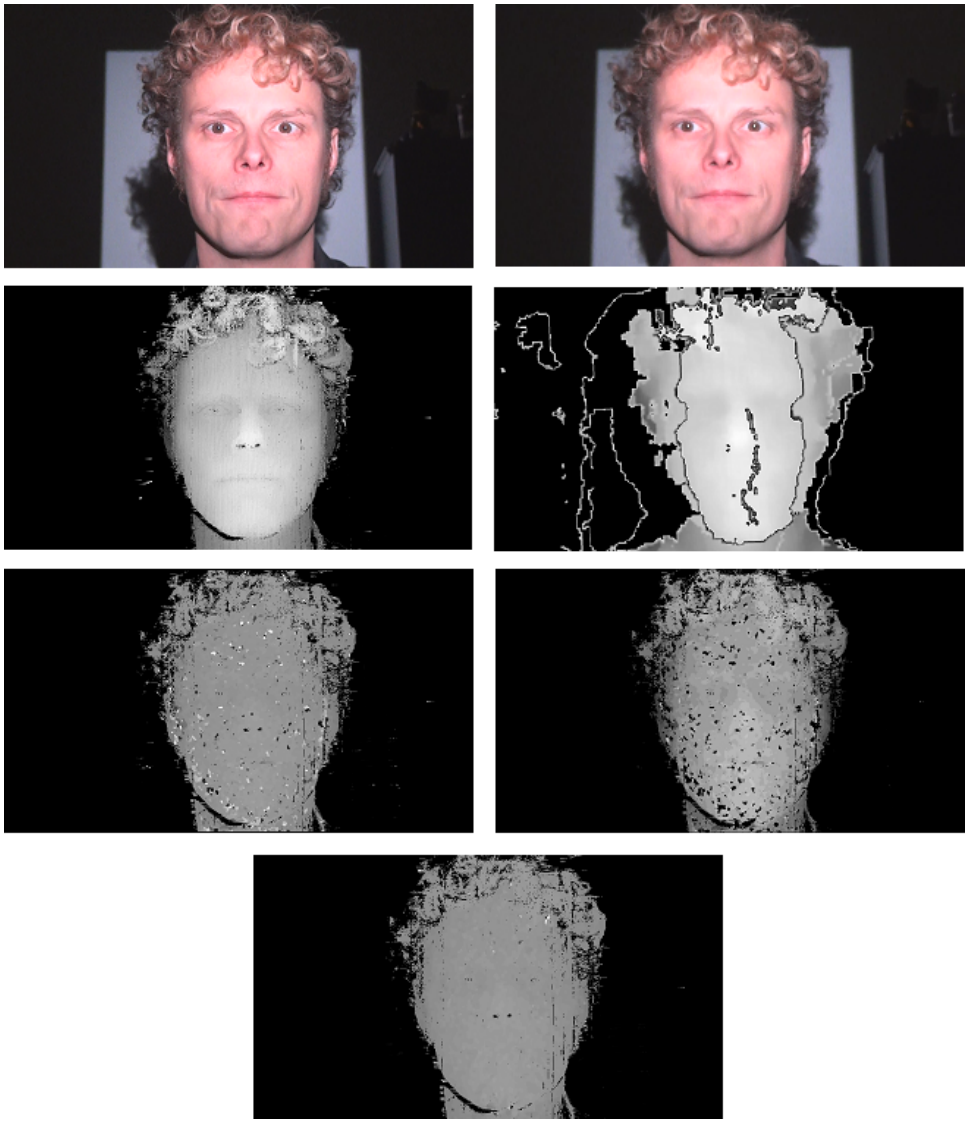

Figure 6.9 Résultats obtenus sans transfert de paramètres en appliquant notre modèle aux prises de vues de gauche et droite (en haut à gauche et à droite) en utilisant la base de vérité et la kinect (deuxième ligne première et seconde colonne) pour produire une carte de disparité en combinant les coûts et en utilisant la kinect pour évaluer les transitions et faire de l'inférence pour débruiter (troisième ligne) ou en utilisant uniquement notre base de vérité pour paramétrer notre modèle (en bas).

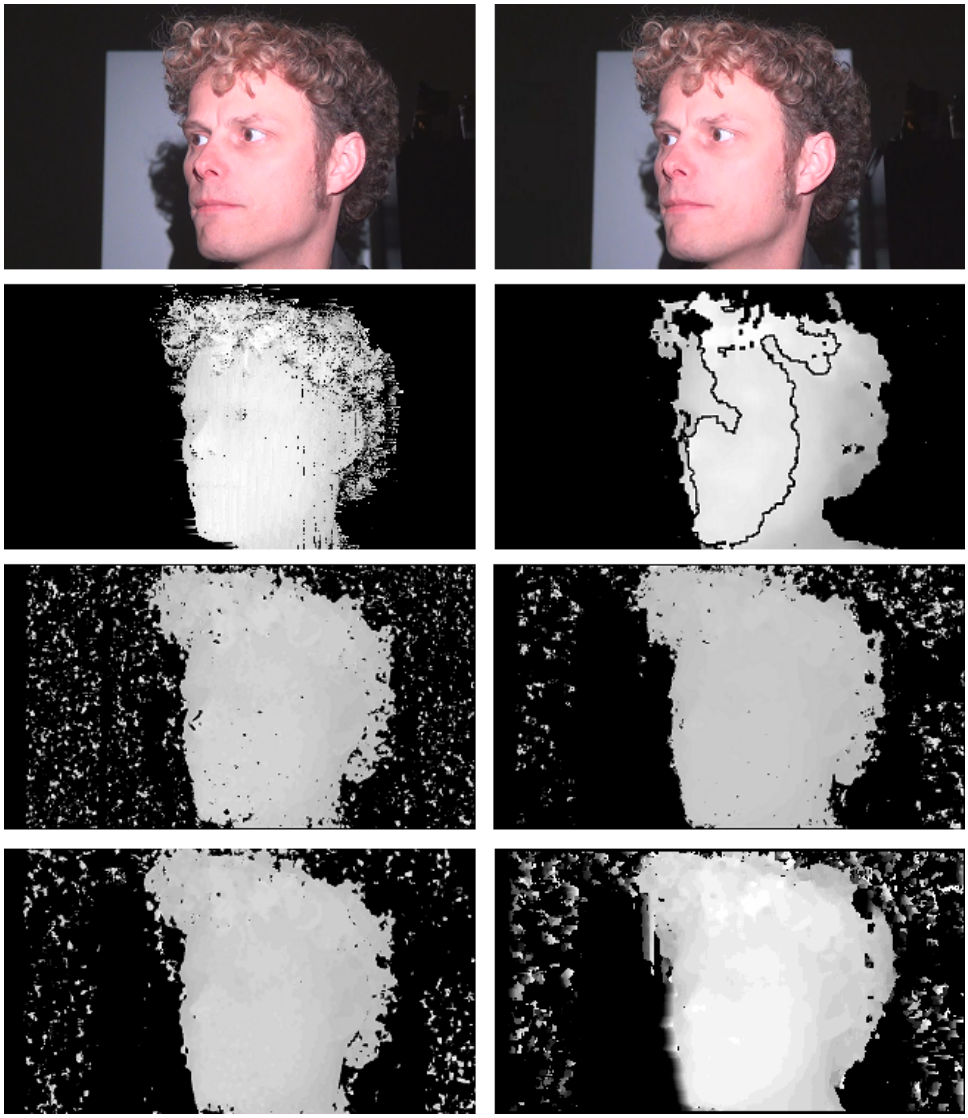

Figure 6.10 Résultats obtenus par transfert de paramètres en appliquant notre modèle aux prises de vue de gauche et droite (en haut à gauche et à droite) en utilisant la base de vérité et la kinect (deuxième ligne première et seconde colonne) pour produire une carte de disparité en combinant les coûts et en utilisant la kinect pour évaluer les transitions et faire de l'inférence pour débruiter (troisième ligne) ou en utilisant uniquement la base de vérité précédente ou la base de vérité courante pour paramétrer notre modèle (en bas).

CHAPITRE 7

CONCLUSION

La vision par ordinateur, et plus précisément la numérisation 3D à partir de captures stéréoscopiques est un vaste champ de recherche et il aurait été illusoire d'espérer le résoudre en une maîtrise. Toutefois, nous pensons avoir couvert d'intéressantes pistes de recherche et nous proposons un modèle innovant pour tenir compte de la cohérence de nos scènes.

7.1 Synthèse des travaux

Au fil de cette maîtrise nous avons étudié et approfondi différents concepts, en partant de la base de la vision stéréoscopique pour nous intéresser à des techniques avancées d'apprentissage machine en passant par des techniques de rendu graphique.
Nous avons essayé de retranscrire cette démarche à travers ce mémoire en mentionnant dans un premier temps les techniques couramment utilisées pour faire de la stéréoscopie active et étudiée comment obtenir une base de vérité à l'aide de lumière structurée aussi bien que comment il serait envisageable d'accélérer une approche similaire reposant sur le multiplexage en couleur.
Nous avons ensuite vu les deux approches principales de la stéréoscopie passive, à savoir la correspondance par bloc et le census. Les résultats obtenus par ces approches n'étant pas satisfaisant nous avons alors étudié différents modèles probabilistes et divers paramétrisations de nos données pour améliorer la qualité des cartes de disparités produites.
Un des principaux axes allant dans ce sens est, notamment, de mettre l'accent sur le traitement a priori et l'agrégation de coûts plutôt que le débruitage. Cela requiert alors d'obtenir des connaissances sur notre scène. En d'autres mots d'apprendre des paramétrisations pour nos coûts de correspondances afin de pouvoir exploiter au mieux nos modèles probabilistes.
Le second grand axe d'amélioration de nos modèles consistait à avoir recours à un modèle réellement bidirectionnel afin de mieux prendre en compte la cohérence spatiale de nos scènes. Cette approche se rapprochant de l'agrégation de coûts souvent faite en programmation dynamique a donc ici été considérée d'un point de vue probabiliste paramétrisable et mathématiquement moins arbitraire.
Enfin, nous nous sommes intéressés à la stéréoscopie multimodale afin d'essayer de comparer différents modèles aussi bien que d'obtenir autant d'information que possible en un temps de capture restreint.

Par ailleurs, nous avons également porté une grande attention au temps requis par nos calculs. En cherchant autant que possible à optimiser et réutiliser les calculs d'une part, mais également en utilisant au mieux l'ensemble des composants à notre disposition en ayant recours au GPGPU notamment et en choisissant soigneusement la mémoire utilisée.
En résumé, nous avons donc développé les différentes étapes permettant d'aboutir non seulement à un schéma expérimental permettant la comparaison et l'évaluation de cartes de disparités. Mais également proposé notre propre modèle basé sur des implémentations hautement parallélisées de différentes stratégies de calculs de correspondance et sur la transformation en modèle probabiliste d'approche de programmation dynamique.

7.2 Limitations de la solution proposée

Une importante limite de notre modèle vient du fait que certains paramètres tels que la taille de nos blocs ou de nos descripteurs sont déterminés sur le jeu de données Middlebury qui ne ressemble pas forcément aux scènes que nous souhaitons numériser. Il faudrait donc prévoir une plus grande probabilité et un apprentissage de ces paramètres en plus des autres.
Par ailleurs, nous sommes limités dans notre numérisation par lumière structurée qui requiert une importante immobilité de la scène que nous désirons numériser.
Enfin, notre schéma expérimental et notre montage ne couvre qu'une faible zone de l'espace et ne nous permettent de numériser qu'à une certaine distance de la Kinect (au-delà du seuil limite pour avoir une profondeur) et suffisament proche de la caméra pour être en mesure de capturer nos scènes avec une grande définition.

7.3 Améliorations futures

Au long de ce mémoire, nous avons mentionné plusieurs pistes envisageables pour poursuivre ce travail et éventuellement améliorer les performances. Malheureusement, faute de temps ces pistes ont dû être laissées de côté pour le moment et devraient être développées ultérieurement. Comme mentionné dans la section précédente le mouvement de sujet numérisé et la synchronisation entre notre caméra et notre projecteur posent d'importants problèmes dans notre approche de numérisation par lumière structurée. Une piste d'amélioration possible serait donc d'envisager l'utilisation de schémas sinusoïdaux qui requièrent moins de projections ainsi qu'un projecteur physiquement synchronisé avec les caméras.
Une autre amélioration possible serait d'étudier l'impacte de la taille des blocs sur des données similaires à nos visages numérisés. En effet, les scènes de Middlebury représentent des

portions d'espaces meublés qui diffèrent de nos numérisations, ce qui explique en partie la piètre qualité de nos numérisations dans la dernière section.
Une autre idée d'optimisation serait d'utiliser des détecteurs de points clés sur un visage et d'utiliser alors des matrices de transitions et d'émissions spécialisées dans certaines régions du visage.
Il serait également intéressant d'explorer d'autres pistes d'utilisation de la kinect et de la stéréoscopie multimodale. Telles que par exemple considérer les données obtenues comme des observations supplémentaires pour notre modèle probabiliste.
Enfin, notre paramétrisation de la distribution des coûts pourrait également être revue pour utiliser un mélange de gaussiennes ou de lois binomiale pour mieux représenter la répartition de nos correspondances. Les paramètres d'un tel modèle pourraient alors être appris en utilisant l'algorithme de maximisation d'espérance.

RÉFÉRENCES

AGARWAL, S., SNAVELY, N., SIMON, I., SEITZ, S. et SZELISKI, R. (2009). Building rome in a day. *Computer Vision, 2009 IEEE 12th International Conference*, 72– 79.

ANSAR, A., CASTANO, A. et MATTHIES, L. (2004). Enhanced real-time stereo using bilateral filtering. *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium*, 455 – 462.

AYACHE, N. et HANSEN, C. (1988). Rectification of images for binocular and trinocular stereovision. *Pattern Recognition, 1988., 9th International Conference*, 1, 11 –16.

BASRI, R. et JACOBS, D. (2001). Photometric stereo with gen- eral, unknown lighting. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 374–381.

BATTLE, J., MOUADDIB, E. et SALVI, J. (1998). Recent Progress In Coded Structured Light As A Technique To Solve The Correspondence Problem : A Survey. *Pattern Recognition*, 31, pp. 963—982.

BHOLE, C., MORSILLO, N. et PAL, C. (2011). 3d segmentation in ct imagery with conditional random fields and histograms of oriented gradients. *Proceedings of the Second international conference on Machine learning in medical imaging*, 326–334.

BIBER, P., ANDREASSON, H., DUCKETT, T. et SCHILLING, A. (2004). 3d modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference*, 4, 3430 – 3435.

BISHOP, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

CAMILLI, F. et PRADOS, E. (2006). Shape-from-shading with discontinuous image brightness. *Numerical methods for viscosity solutions and applications*, 1225–1237.

DU, H., GOLDMAN, D. et SEITZ, S. (2011). Binocular photometric stereo. *Proceedings of the British Machine Vision Conference*, 84.1–84.11.

FORSYTH, D. A. et PONCE, J. (2002). *Computer Vision : A Modern Approach*. Prentice Hall, première édition.

HIRSCHMULLER, H. (2008). Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 30, 328 –341.

HUMENBERGER, M., ENGELKE, T. et KUBINGER, W. (2010). A census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching

quality. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference*, 77 –84.

ISGRO, F., TRUCCO, E., KAUFF, P. et SCHREER, O. (2004). Three-dimensional image processing in the future of immersive media. *IEEE Transactions on Circuits and Systems for Video Technology*, 14.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A. et FITZGIBBON, A. (2011a). Kinectfusion : real-time 3d reconstruction and interaction using a moving depth camera. *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 559–568.

IZADI, S., NEWCOMBE, R. A., KIM, D., HILLIGES, O., MOLYNEAUX, D., HODGES, S., KOHLI, P., SHOTTON, J., DAVISON, A. J. et FITZGIBBON, A. W. (2011b). Kinectfusion : real-time dynamic 3d surface reconstruction and interaction. *SIGGRAPH Talks*, 23.

JENSEN, F. V. (1993). *Introduction to Bayesian Networks*. Hugin Expert A/S.

KANG, S. B., WEBB, J. A., ZITNICK, C. L. et KANADE, T. (1995). A Multibaseline Stereo System with Active Illumination and Real-time Image Acquisition. *Proceedings of the Fifth International Conference on Computer Vision (ICCV '95)*, pp. 88–93.

KONOLIGE, K. (2010). Projected texture stereo. *Robotics and Automation (ICRA), 2010 IEEE International Conference*, 148 –155.

KOSCHAN, A. (1993). Dense stereo correspondence using polychromatic block matching. D. Chetverikov et W. Kropatsch, éditeurs, *Computer Analysis of Images and Patterns*, Springer Berlin / Heidelberg, vol. 719 de *Lecture Notes in Computer Science*. 538–542.

KOSCHAN, A., RODEHORST, V. et SPILLER, K. (1996). Color stereo vision using hierarchical block matching and active color illumination. *Pattern Recognition, 1996., Proceedings of the 13th International Conference*, 1, 835 –839.

KRAMER, J., BURRUS, N., ECHTLER, F. et PARKER, M. (2012). *Hacking the Kinect*. Apress Publishers.

LANMAN, D. et TAUBIN, G. (2009). Build your own 3d scanner : 3d photograhy for beginners. *SIGGRAPH '09 : ACM SIGGRAPH 2009 courses*, 1–87.

LEICA (2012). http ://hds.leicageosystems.com/en/leicascanstationc10_79411.htm. *leicageosystems.com*.

LIM, J. (2009). Optimized projection pattern supplementing stereo systems. *Robotics and Automation, 2009. ICRA '09. IEEE International Conference*, 2823 –2829.

LOWE, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 91–110.

MA, L. Y. (2012). *Traitement et analyse d'images stéréoscopiques avec les approches du calcul générique sur un processeur graphique*. École Polytechnique de Montréal.

MARICHAL-HERNANDEZ, J. G., PEREZ NAVA, F., ROSA, F., RESTREPO, R. et RODRIGUEZ-RAMOS, J. M. (2006). An integrated system for virtual scene rendering, stereo reconstruction and accuracy estimation. *Proceedings of the conference on Geometric Modeling and Imaging : New Trends*, 121–126.

MARR, D. (1982a). http ://www.princeton.edu/∼gdetre/notes/neuro_generals/marr.html. *princeton.edu*.

MARR, D. (1982b). *Vision : A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman and Company.

MEI, X., SUN, X., ZHOU, M., JIAO, S., WANG, H. et ZHANG, X. (2011). On building an accurate stereo matching system on graphics hardware. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference*, 467 –474.

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHLI, P., SHOTTON, J., HODGES, S. et FITZGIBBON, A. W. (2011). Kinectfusion : Real-time dense surface mapping and tracking. *ISMAR*, 127–136.

PAGÈS, J., SALVI, J., COLLEWET, C. et FOREST, J. (2005). Optimised de bruijn patterns for one-shot shape acquisition. *Image and Vision Computing*, 23, 707 – 720.

PRADOS, E. et FAUGERAS, O. D. (2005). Shape from shading : A well-posed problem ? *CVPR*, 870–877.

RIPLEY, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

RUSSELL, S. J. et NORVIG, P. (2003). *Artificial Intelligence : A Modern Approach*. Pearson Education.

SALVI, J., PAGÈS, J. et BATLLE, J. (2004). Pattern codification strategies in structured light systems. *Pattern Recognition*, 37, 827 – 849.

SCHAEFFER, M. et PARR, R. (2006). Efficient selection of disambiguating actions for stereo vision. *UAI'06*.

SCHARSTEIN, D. et SZELISKI, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47, 7–42.

SCHARSTEIN, D. et SZELISKI, R. (2003). High-accuracy stereo depth maps using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 195 – 202.

SCHINDLER, G. (2010). Photometric stereo via computer screen lighting for real-time surface reconstruction.

SZELISKI, R. (2006). Image alignment and stitching : a tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2, 1–104.

SZELISKI, R. (2010). *Computer Vision*. Springer.

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R. et FITZGIBBON, A. (2000). *Bundle Adjustment — A Modern Synthesis*, vol. 1883. Springer Berlin / Heidelberg.

VITERBI, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions*, 13, 260–269.

WEBER, M., HUMENBERGER, M. et KUBINGER, W. (2009). A very fast census-based stereo matching implementation on a graphics processing unit. *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference*, 786 –793.

WEINMAN, J. J., TRAN, L. et PAL, C. J. (2008). Efficiently learning random fields for stereo vision with sparse message passing. *Proceedings of the 10th European Conference on Computer Vision : Part I*, 617–630.

YIN, H. B., JIA, H., QI, H., JI, X., XIE, X. et GAO, W. (2010). A hardware-efficient multi-resolution block matching algorithm and its vlsi architecture for high definition mpeg-like video encoders. *IEEE Trans. Circuits Syst. Video Techn.*, 20, 1242–1254.

ZABIH, R. et WOODFILL, J. (1994). Non-parametric local transforms for computing visual correspondence. J.-O. Eklundh, éditeur, *Computer Vision — ECCV '94*, Springer Berlin / Heidelberg, vol. 801 de *Lecture Notes in Computer Science*. 151–158. 10.1007/BFb0028345.

ZHANG, L., SNAVELY, N., CURLESS, B. et SEITZ, S. M. (2004). Spacetime faces : High-resolution capture for modeling and animation. *ACM Annual Conference on Computer Graphics*, 548–558.

ANNEXE A

Résultats généraux

Nous présentons ici l'ensembles des résultats obtenus sur le jeu de données Middlebury ordonnés comme suit : en haut à gauche une vue de la scène, en haut à droite la base de vérité, deuxième ligne à gauche les résultats obtenus par correspondance par bloc et par census à droite, troisième ligne à gauche se trouve les résultat de l'utilisation de MMC horizontaux et verticaux en utilisant l'algorithme de viterbi sur les résultats du census, à droite se trouve les résultats de la combinaison de coûts, en bas à gauche se trouve le résultat de l'utilisation de l'algorithme du forward/backward sur la combinaison des coûts et en bas à gauche les résultats de notre modèle en étoile.

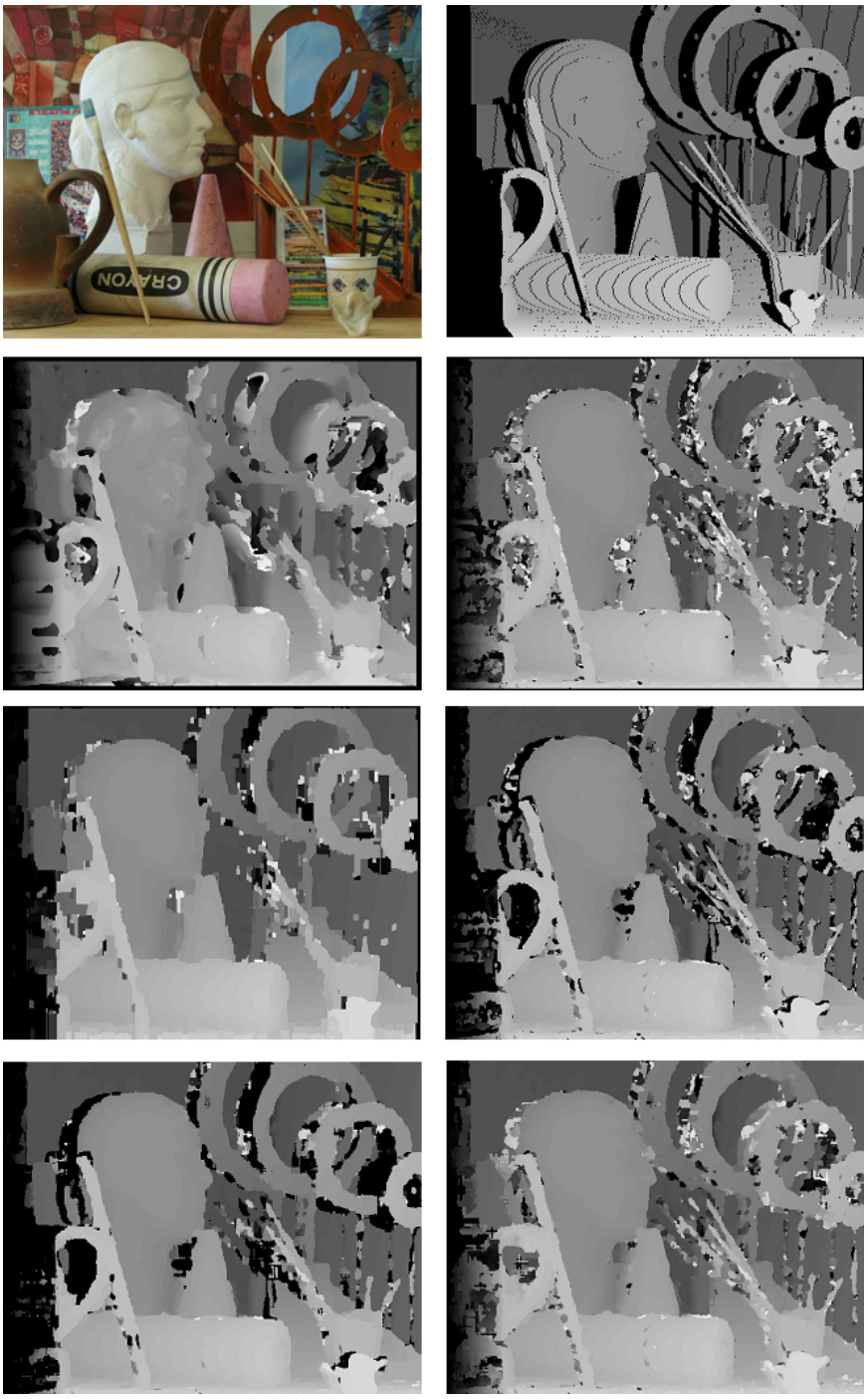

Figure A.1 Résultats : Art

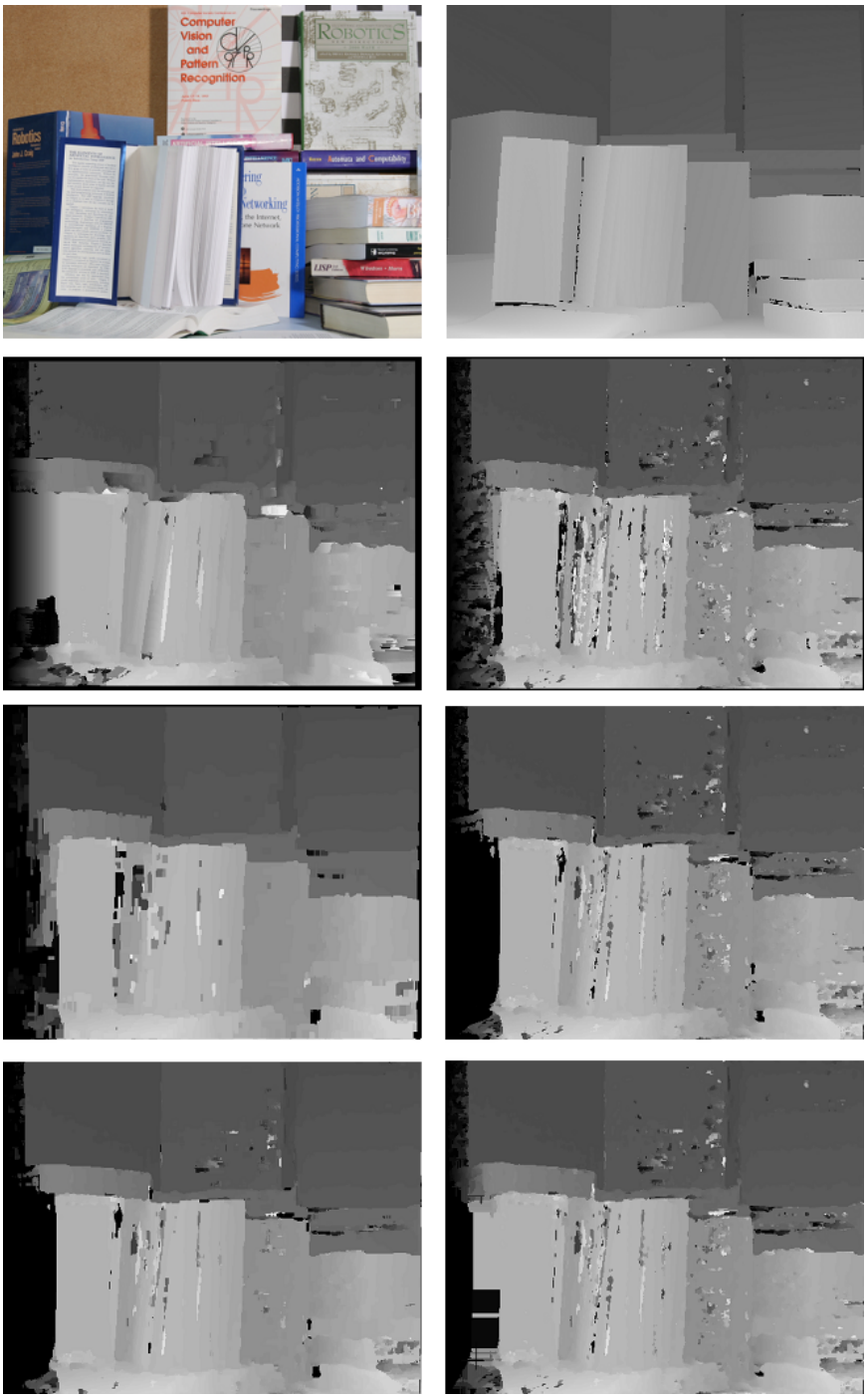

Figure A.2 Résultats : Books

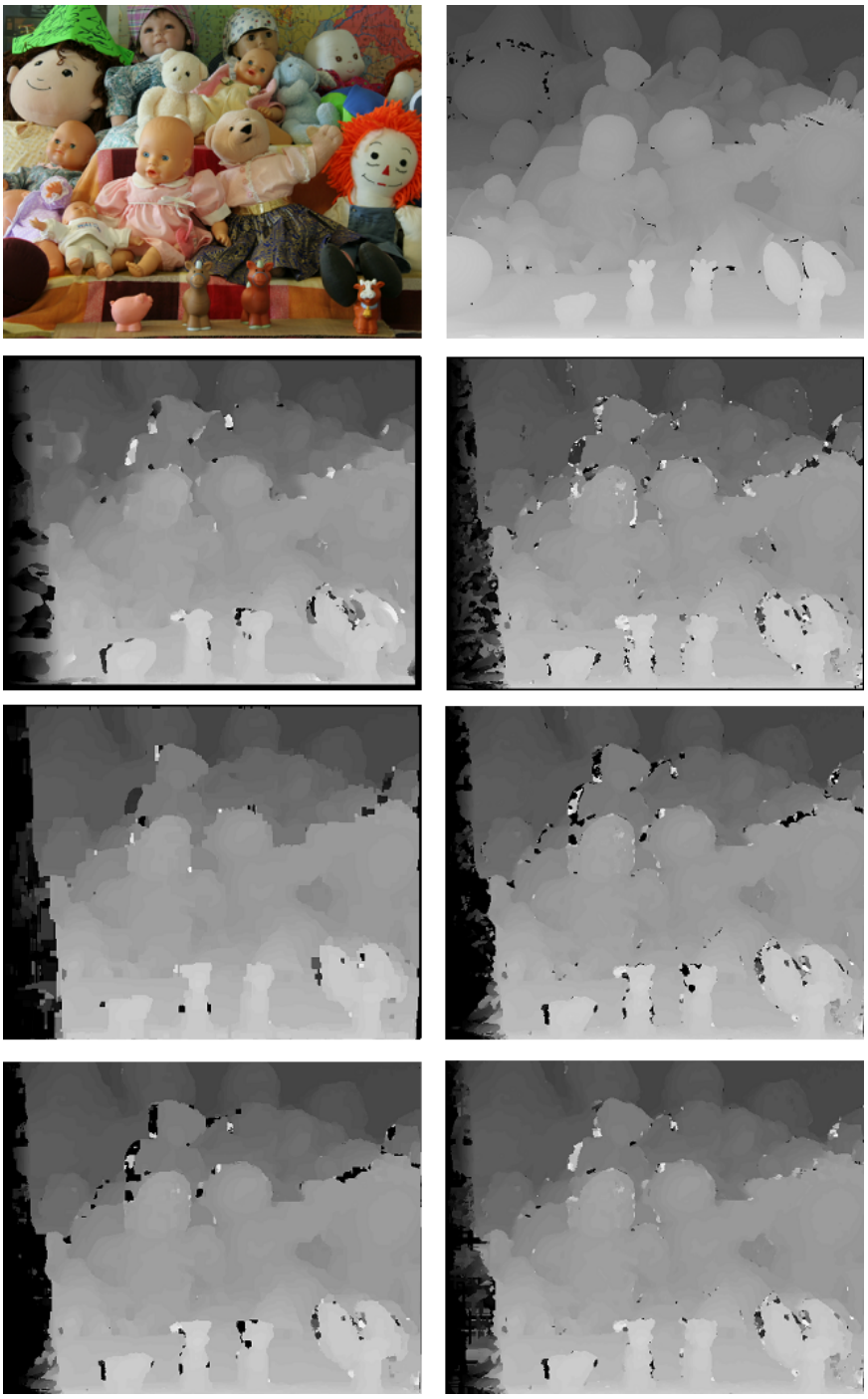

Figure A.3 Résultats : Dolls

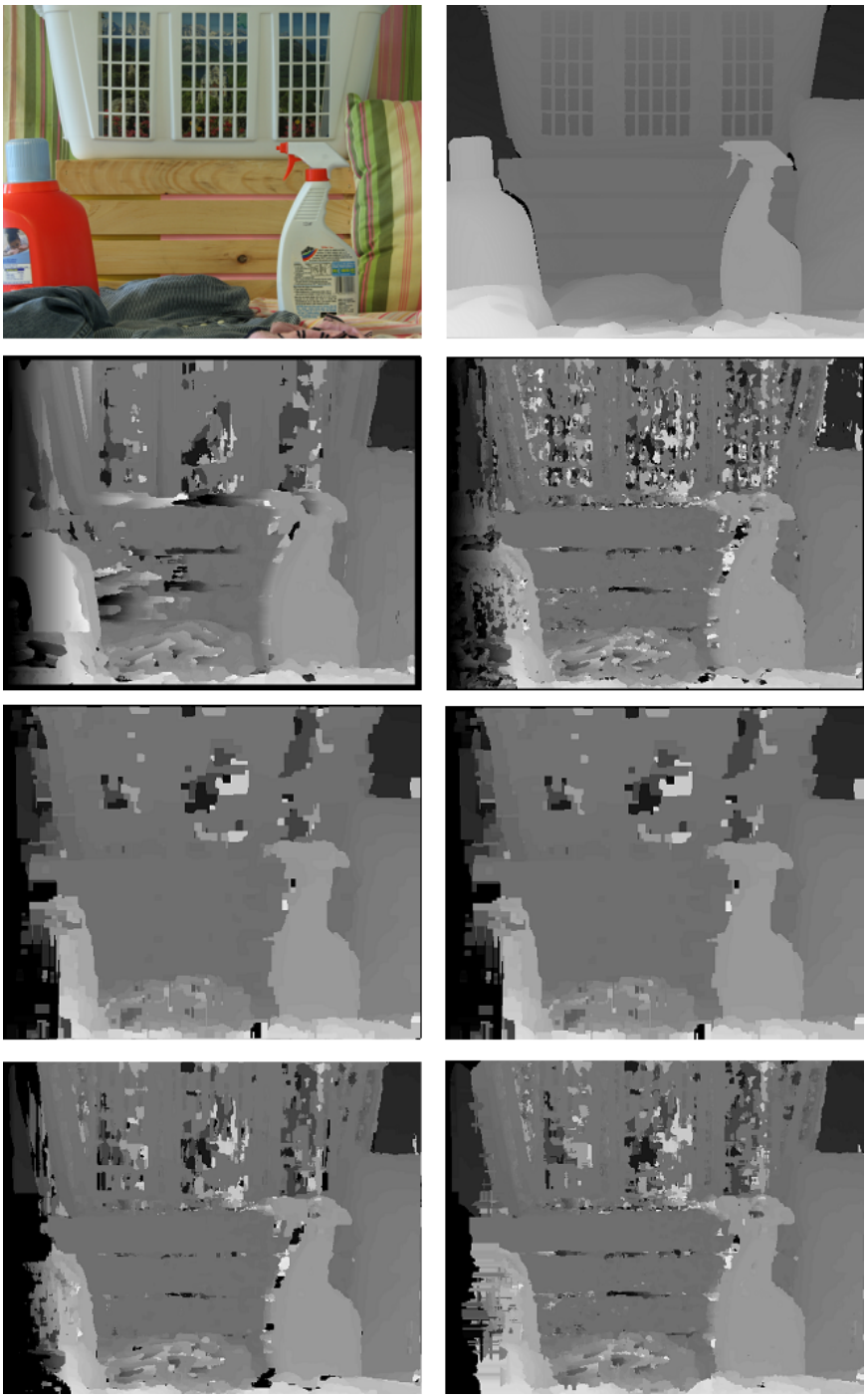

Figure A.4 Résultats : Laundry

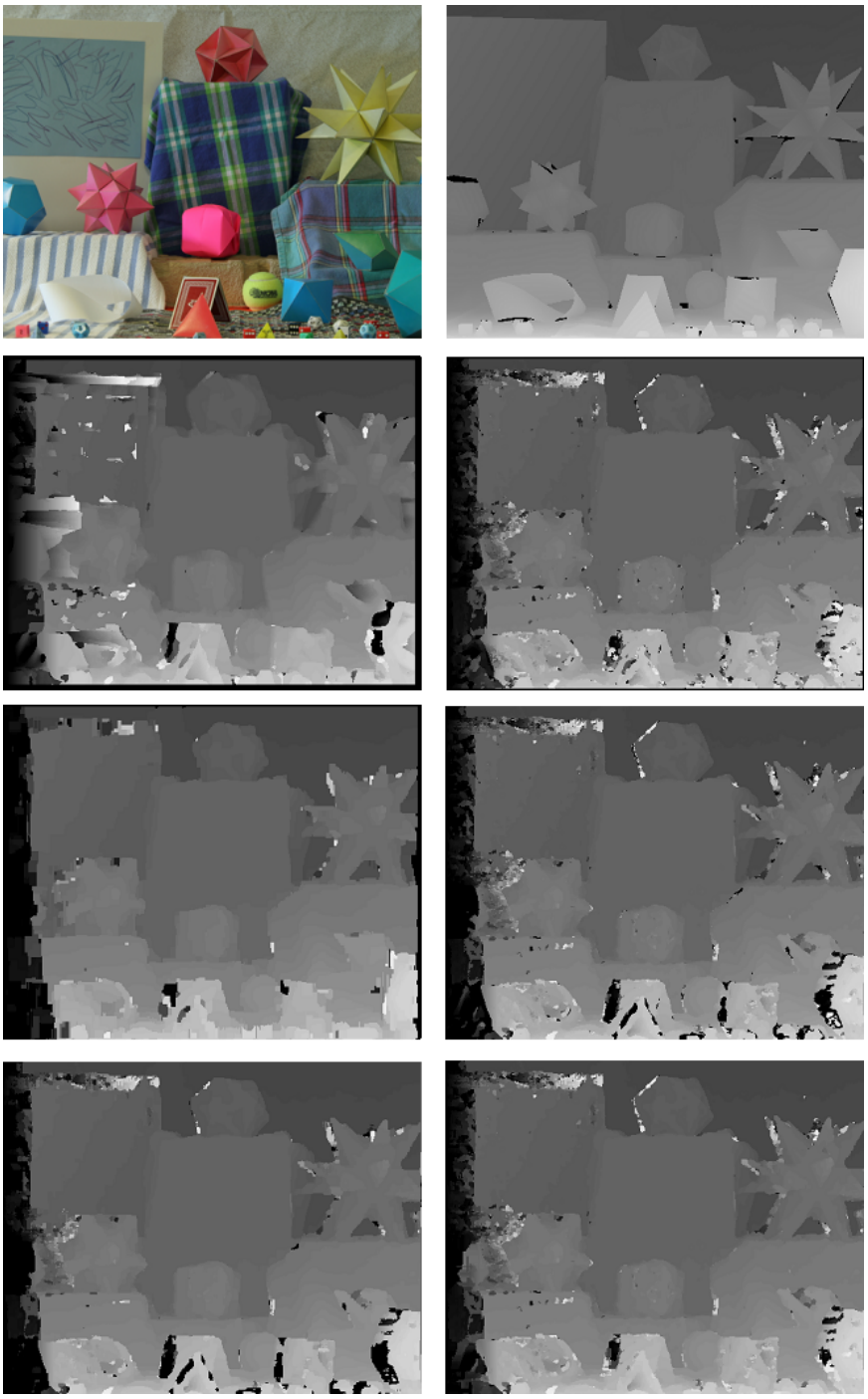

Figure A.5 Résultats : Moebius

Figure A.6 Résultats : Reindeer