Université de Montréal

# Réalité augmentée automatique à partir d'une séquence vidéo et utilisant la stéréoscopie dense

par

**Patrick Holloway**

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

avril, 2003



Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

# Réalité augmentée automatique à partir d'une séquence vidéo et utilisant la stéréoscopie dense

présenté par

Patrick Holloway

a été évalué par un jury composé des personnes suivantes:

Pierre Poulin
(Président-rapporteur)

Sébastien Roy
(Directeur de recherche)

Max Mignotte
(Membre du jury)

Mémoire accepté le 21 août 2003

# RÉSUMÉ

Ce mémoire s'intéresse à la réalité augmentée, c'est-à-dire à l'intégration d'objets virtuels dans des images réelles. La difficulté vient du fait que l'on veut que cette intégration de l'objet virtuel se fasse de façon réaliste, c'est-à-dire en tenant compte de la profondeur des objets réels. Les applications de réalité augmentée en industrie utilisent toutes des méthodes à forte interactivité humaine. La méthode que nous proposons dans ce mémoire est automatique et ne nécessite l'interaction d'un utilisateur que pour la prise de la séquence vidéo. En résumé, son fonctionnement est le suivant. On crée d'abord une carte de profondeur dense de la scène. Puis on intègre l'objet virtuel, dont on connaît la profondeur, à l'image originale aux endroits où il se trouve devant les éléments de la scène. Ceci se fait en comparant les deux profondeurs : celle de l'objet virtuel et celle de la scène. La carte de profondeur dense est fabriquée en utilisant l'algorithme de flot maximum (réf. [22]) et à l'aide de deux nouvelles techniques : la technique "max-gauche-droite" et la technique "consistance de profondeur". La technique "max-gauche-droite" permet de régler les problèmes liés aux demi-occlusions. Quant à la technique "consistance de profondeur", elle offre une solution à l'un des plus grands problèmes de la stéréoscopie dense : le choix du niveau de lissage.

Dans le présent mémoire, tous les aspects de la méthode seront étudiés, de la prise du film à l'intégration des objets virtuels dans les images, en passant par la calibration, la stéréoscopie, et bien sûr, nos deux nouvelles techniques.

Mots clés : réalité augmentée, stéréoscopie, demi-occlusion, niveau de lissage.

# ABSTRACT

This thesis concentrates on augmented reality, which consists in the integration of virtual objects in real images. This integration must be realized in a realistic way, which creates a difficulty because of the depth of the real objects. All industrial applications of augmented reality require a strong human interactivity. The method we propose in this thesis is automatic and only needs a human interaction for the movie capture. First, a dense depth map of the scene is created from the image sequence. After that, we integrate the virtual object, whose depth is known, to the original image where it is in front of the scene elements. This is done by comparing two depths : the depth of the scene and the depth of the virtual object. The dense depth map is created by the max-flow algorithm (ref. [22]), which is combined to two new techniques : the "max-left-right" and the "depth consistency". The "max-left-right" technique solves the half-occlusions problem. The "depth consistency" technique offers a solution to one of the biggest problem of dense stereoscopy : the choice of the level of smoothness.

In this thesis, all aspects of our method will be discussed, from the movie capture to the virtual object integration in the real images, as well as calibration, stereoscopy, and of course, our two new techniques.

Keywords : augmented reality, stereoscopy, half-occlusion, level of smoothness.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# REMERCIEMENTS

Je tiens à remercier tous ceux qui m'ont aidé à la création de ce mémoire, et plus particulièrement les personnes suivantes :

- Sébastien Roy ainsi que les autres membres du laboratoire de Vision 3D et de Traitement d'images, pour l'aide précieuse qu'ils m'ont apportée, et sans laquelle ce travail n'aurait pas vu le jour.
- Mes parents et ma soeur, que je ne remercierai jamais assez pour leur éternel support en mon endroit.
- Mes amis, pour ces parties de "Settlers of Catan" qui m'ont permis de garder le moral tout au long de cette maîtrise.

*À mes parents*

*"L'esprit n'est pas un récipient que l'on doit remplir,*

*mais un feu qu'il faut allumer."*

Plutarque

# Chapitre 1

# INTRODUCTION

La réalité augmentée s'intéresse à la combinaison d'images réelles et virtuelles en 3D et fera l'objet de ce mémoire. Il s'agit donc d'introduire des objets virtuels modélisés en 3D à des images réelles de façon réaliste. Ce processus est parfois appliqué en temps réel (réf. [2]). Les pilotes de F-22, par exemple, ont un système intégré à leur casque qui leur permet d'accéder à des boutons de commandes qui changent selon les besoins. Ces boutons virtuels n'existent qu'à travers leur visière. Mais dans le cas du présent mémoire, le traitement ne se fait pas en temps réel (réf. [28], [8], [33]). On parle ici de post-production, c'est-à-dire que l'insertion des images virtuelles se fait dans une étape ultérieure à l'acquisition de la séquence vidéo. Etant donné que le temps est moins une contrainte, le traitement peut être plus élaboré et cela conduit à un résultat final de meilleure qualité. Si le temps est "moins" une contrainte, il l'est néanmoins toujours un peu. Un système qui permettrait des images finales parfaites mais qui prendraient des années à s'exécuter ne serait pas intéressant. Et quand on parle d'images, le traitement est habituellement très long. C'est pourquoi le temps reste toujours une contrainte malgré tout. Les applications industrielles de ce système sont nombreuses. Il est utilisé, entre autres, pour des études d'impact visuel, en architecture particulièrement. Il est intéressant, avant de construire un gratte-ciel par exemple, de "voir" comment il va s'intégrer au décor de la ville et ce, en l'ajoutant virtuellement dans une photo. Apporter des ajustements à ce stade du projet est infiniment moins coûteux qu'une fois l'immeuble construit. Hydro-Québec fait présentement des études d'impacts pour savoir si l'ajout de nouveaux pilônes

électriques ne va pas trop nuire au paysage vu de l'autoroute. Mais pour l'instant, les principales applications dans ce domaine sont reliées aux effets spéciaux utilisés pour le cinéma et la télévision. Les méthodes traditionnelles pour créer des effets spéciaux sont les cascades et les maquettes. Les cascades sont souvent dangereuses et nécessitent une grande préparation, ce qui se traduit par des coûts très élevés. Les maquettes quant à elles, finissent par être très coûteuses si l'on désire une bonne qualité dans les détails. Et même dans ces conditions, il y aura toujours quelques imperfections qui trahiront leurs petites tailles. Les maquettes, tout comme les cascades, contiennent aussi des limitations physiques. Par exemple, pour un vaisseau spatial qui se déformerait sous l'intense gravité d'un trou noir, il est difficile de créer une maquette qui rendrait justice à la vision de l'auteur. Et il est évidemment impossible de créer cette cascade! Aujourd'hui, si les cascades et les maquettes sont toujours utilisées, elles sont remplacées peu à peu par des scènes "augmentées" d'objets virtuels. Ces scènes "augmentées" sont parfaitement sécuritaires, n'ont aucune limitation physique et les coûts ne sont habituellement pas trop élevés. Mais cela reste encore quand même assez coûteux avec les techniques actuelles. En effet, la méthode du masque manuel, qui est actuellement largement utilisée en industrie, reste assez rudimentaire. Après avoir créé l'objet virtuel à ajouter dans le film, il faut l'intégrer aux images réelles de façon réaliste, c'est-à-dire en tenant compte de la profondeur des objets de la scène afin de déterminer les occlusions. En effet, si on ajoute un avion virtuel qui passe derrière une montagne, la montagne doit cacher l'avion, mais ce dernier doit cacher le ciel derrière lui. Pour faire cela avec la technique du masque, on doit engager plusieurs artistes qui vont dessiner le masque en question par-dessus la montagne, de sorte que la montagne ne soit pas "effacée" par l'avion virtuel qui passe derrière. Il faut dessiner ce masque pour tous les éléments de la scène qui sont à la fois en contact avec l'objet virtuel et plus près de la caméra que ce dernier, et ce, pour toutes les images de la séquence vidéo. On peut résumer cette méthode en cinq étapes.

- Etapes pour la méthode du masque :
  1. Filmer la séquence originale
  2. Calibrer la caméra de façon automatique
  3. Modéliser l'objet virtuel
  4. Dessiner le masque
  5. Intégrer l'objet virtuel dans la scène aux endroits qui ne sont pas masqués.

La quatrième étape (dessiner le masque) exige un travail manuel fastidieux qui demande beaucoup de temps (et donc qui coûte cher). Pour éviter que ce travail prenne trop de temps, on va souvent se contenter de plans fixes ou contenant seulement une rotation, ce qui va permettre de "recycler" le masque pour plusieurs images. Mais pour les plans libres, où la caméra bouge à volonté dans la scène, les artistes doivent redessiner le masque à chaque image. Ce que nous proposons dans ce mémoire est une méthode complètement automatique pour introduire des objets virtuels dans une séquence vidéo. Cette méthode se décompose en cinq étapes.

- Etapes pour notre méthode :
  1. Filmer la séquence originale
  2. Calibrer la caméra de façon automatique
  3. Modéliser l'objet virtuel
  4. Fabriquer la carte de profondeur (depth map) de la scène filmée.
  5. Intégrer l'objet virtuel dans la scène filmée à partir de la carte de profondeur obtenue à l'étape #4.

Les étapes #1-2-3 sont identiques à la méthode par masque, mais les étapes #4-5 sont nouvelles. C'est sur l'étape #4 justement, la fabrication de la carte de profondeur, que s'est concentré le plus gros de nos efforts. La calibration et la fabrication de la carte de profondeur sont des problèmes auxquels de nombreux chercheurs ont apporté leurs solutions. Mais aucune de ces solutions n'est parfaite. Chacune de ces

méthodes possède sa source d'erreurs et la fabrication de la carte de profondeur étant tributaire de la calibration, les erreurs se multiplient et finissent par donner une image finale d'une qualité plutôt discutable. C'est pour cette raison, pour que l'intégration de l'objet virtuel se fasse sans anomalie géométrique avec la scène, que nous avons apporté nos propres améliorations : la technique "max-gauche-droite" et la technique "consistance de profondeur". Ces deux nouvelles techniques s'inscrivent dans l'étape #4 (fabrication de la carte de profondeur) et seront présentées un peu plus loin dans ce mémoire. Il a fallu aussi ajouter quelques contraintes sur le mouvement de la caméra et la nature de la scène afin de simplifier le problème, et ainsi, améliorer la qualité de la séquence vidéo finale. Le prochain chapitre traitera de ces étapes plus en détail.

Chapitre 2

# ETAPES DE LA MÉTHODE

Ce chapitre décrit, plus en détail, les cinq étapes de notre méthode.

## *2.1 Etape #1 : Filmer la séquence originale*

Ici seront définies les principales contraintes. D'abord, la séquence vidéo devra être filmée par une caméra vidéo digitale qui se déplacera assez lentement dans la direction à peu près perpendiculaire à l'axe de visée. Ce déplacement devra se faire sans retour en arrière. Ceci facilitera la calibration, et aussi, permettra de pouvoir bénéficier de la technique "max-gauche-droite" qui exige que des images soient d'un côté du "passé" et de l'autre côté du "futur" par rapport à l'image "présente". Cette technique sera expliquée plus en détail à l'étape #4. Autre contrainte importante : pour que la carte de profondeur représente bien la profondeur de la scène, seule la caméra doit bouger. Tous les objets de la scène doivent être immobiles et leur mouvement apparent ne doit être dû qu'au déplacement de la caméra vidéo. De plus, le film doit être composé d'images non-entrelacées. Les expériences nous ont prouvé que la fonction de coûts nécessaire à la fabrication de la carte de profondeur réagit très mal à l'entrelacement des images. L'entrelacement est une technique qui a été introduite à l'apparition de la télévision afin de permettre de regarder une émission à 30 images/seconde sans un agaçant clignotement ("flicker"). Pour empêcher ce clignotement, l'oeil humain a besoin que l'écran soit rafraîchi à 60 images/seconde. Mais les appareils de l'époque ne permettaient pas un tel niveau de rafraîchissement. Pour palier à ce problème, un standard a été instauré : l'entrelacement des images (réf. [13]). En ne rafraîchissant qu'une ligne sur deux dans une première passe (les lignes paires) et ensuite les lignes

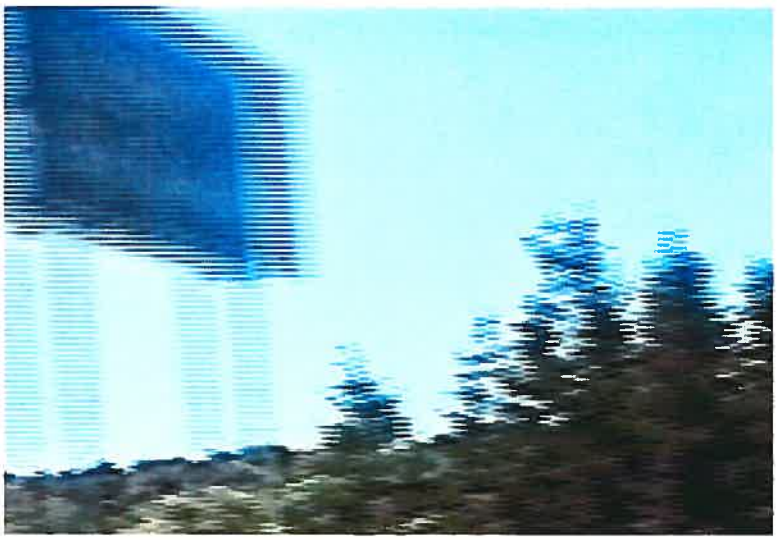


FIG. **2.1. L'entrelacement dans une image. On peut distinguer une pancarte (à gauche) et des arbres (à droite). La photo a été prise à l'intérieur d'une automobile roulant à 30 km/h.**

restantes dans une deuxième passe (les lignes impaires), on arrive à rafraîchir l'écran à 60 demi-images/seconde. L'oeil humain n'arrive pas à distinguer un clignotement dans ces conditions. Si cette technique résout vraiment le problème du clignotement, lorsque l'on regarde de très près les pixels, on peut observer, sur les objets qui se sont déplacés d'une image à l'autre, un décalage entre les lignes paires et impaires. Ceci est dû au fait que les lignes paires ont été dessinées (captées dans le cas d'une caméra vidéo) à un temps (t) différent des lignes impaires (t+1). Ce phénomène est surtout observable sur les grandes lignes frontalières verticales qui présentent une différence de couleurs marquée entre la gauche et la droite. On observe alors que cette ligne frontalière (entre les deux couleurs) apparaît en zigzag, alors qu'en réalité elle est vraiment droite (figure 2.1). Ce zigzag correspond au déplacement de l'objet entre le moment de la capture des lignes paires et celui des lignes impaires. Plus l'objet se déplace rapidement, plus le zigzag sera prononcé. Ce phénomène ne s'observe qu'au niveau pixel. C'est pourquoi cela ne dérange pas le cinéaste amateur, à moins qu'un objet

de la scène (ou lui-même) se déplace à très grande vitesse. Mais pour nos besoins, la fabrication de la carte de profondeur exige une précision de l'ordre du pixel. Alors là, l'entrelacement devient un problème. Quand l'algorithme essaie de mettre en correspondance des pixels d'une image à l'autre, en testant différentes profondeurs on tombe parfois sur des lignes paires, d'autres fois sur des lignes impaires. Etant donné que les lignes paires ont été prises à un moment différent des lignes impaires, ce décalage dans le temps se traduit par un décalage dans la disparité et par conséquent, un décalage dans la profondeur trouvée. Un seul pixel de différence peut équivaloir à des dizaines de mètres de différence de profondeur. L'entrelacement se traduit donc en une importante source d'erreurs. Nous avons essayé de nous débarrasser de l'entrelacement en divisant chaque image en deux, une avec seulement les lignes paires et l'autre avec seulement les lignes impaires. Cela nous donne donc deux fois plus d'images, mais avec la moitié des lignes (donc deux fois moins hautes). Pour combler les lignes manquantes, on fait la moyenne de la ligne précédente et suivante. Cela nous donne donc un film avec toujours deux fois plus d'images que le film original mais maintenant avec des images de la même hauteur et surtout, sans entrelacement. Cependant, ceci n'est pas une solution miracle. Quand on essaie de mettre en correspondance des pixels, inévitablement on tombe sur un pixel "fabriqué" par la moyenne des pixels du haut et du bas. Ces pixels "fabriqués" ne représentent pas vraiment la réalité, mais plutôt des trous espace-temps. A ce moment précis dans le film, la caméra, à cause de son faible taux de rafraîchissement, n'a pas pu capter ce qu'il y avait à cet endroit. Nous l'avons supposé en moyennant les autres pixels connus. Mais cela ne veut pas dire que c'était la réalité. Etant donné que pour chaque image, seule la moitié des lignes proviennent de l'image originale, alors seulement la moitié des pixels des images désentrelacées sont donc vraiment fiables. Par expérience, cette fiabilité est loin d'être suffisante quand vient le temps de fabriquer la carte de profondeur et mène parfois à des résultats erronés. Evidemment, cette technique marche bien dans le cas des scènes n'ayant que des objets grossiers sans détail. Mais dans les scènes

"normales", il y a toujours une multitude de détails qui apparaissent dans certaines images mais disparaissent dans d'autres parce qu'ils sont tombés dans un "trou" et qu'ils ont été "effacés" par moyennage, car ils n'étaient pas assez gros pour occuper aussi les pixels du haut et du bas. Dans ces scènes "normales" que l'on retrouve habituellement, cette technique de désentrelacement devient donc inadéquate. Il ne reste donc plus qu'une solution : régler le problème à sa source en filmant sans entrelacement. Mais la plupart des caméras vidéo bon marché ne permettent pas de filmer sans entrelacement. Heureusement, la dernière caméra vidéo que nous avons reçue au laboratoire permet de faire "à peu près" cela en la mettant en mode "PROGRESSIVE SCAN". En pratique, même si la caméra enregistre 30 images non-entrelacées par seconde, elle filme en fait 15 images impaires et chaque image paire est tout simplement une copie de l'image impaire qui la précède. Le taux de rafraîchissement de cette caméra ne permet pas un vrai 30 images/seconde désentrelacées. Ceci n'est pas vraiment souhaitable non plus mais la solution est triviale : il suffit de ne prendre que les images impaires, ce qui nous donne un film deux fois plus court mais avec exactement ce que nous recherchions : chaque image est différente et désentrelacée. Mais il a fallu prendre deux secondes d'images impaires pour obtenir une seconde de film à 30 images désentrelacées/seconde. Etant donné que nous travaillons en distances relatives, le fait que les objets se déplacent deux fois plus vite qu'en réalité ne dérange en rien l'algorithme de la fabrication de la carte de profondeur, puisque tout se déplace deux fois plus vite. Si par contre on travaillait en distances absolues, en mesurant la profondeur des objets en fonction de leur déplacement dans l'image versus la vitesse de déplacement mesurée de la caméra vidéo, alors évidemment il faudrait en tenir compte. J'ajoute quelques commentaires sur l'étape de la prise du film. D'abord, plus les images sont bruitées, plus les autres étapes vont en souffrir. La calibration sera moins bonne, la fabrication de la carte de profondeur aussi, et par conséquent le résultat final. Toutes les caméras vidéo introduisent un certain niveau de bruit dans les images mais ce niveau dépend de la qualité de la caméra vidéo. Plus

la qualité est basse, plus la caméra introduira de bruit dans les images. Pour réduire ce bruit, on peut toujours passer un filtre passe-bas sur les images mais cela contribue à rendre les contours des objets flous. Or, la calibration et la fabrication de la carte de profondeur se comportent bien quand il y a des contours nets où la disparité entre deux objets est claire. Surtout la technique de calibration utilisée dans notre méthode qui fait du suivi de points ("feature tracking"). Donc les filtres passe-bas ne sont pas très intéressants car ils n'enlèvent pas que le bruit mais aussi les contours nets que l'on voudrait conserver. La seule vraie solution au problème du bruit est de filmer la scène avec la meilleure caméra vidéo possible. Mais les caméras vidéo d'excellente qualité coûtent très chers. Pour les tests que nous avons effectués en laboratoire, nous avons utilisé une caméra vidéo SONY DCR-VX2000 qui est de bonne qualité mais n'a définitivement pas la qualité des grosses caméras vidéo utilisées dans les studios de télévision. Un dernier commentaire : plus la scène sera filmée avec un mouvement simple et à une vitesse assez basse et constante, plus cela facilitera la calibration. D'autant plus qu'à grande vitesse un autre problème survient : le flou de mouvement (réf. [10], [21]). La calibration étant assez difficile d'avance, il vaut mieux s'en tenir à des séquences vidéo simples.

## *2.2 Etape #2 : Calibrer la caméra de façon automatique*

Il y a deux étapes majeures qui composent le coeur de ce système. La première est la fabrication de la carte de profondeur. C'est sur cette étape que nous avons concentré nos efforts et c'est principalement sur cela que porte ce mémoire. L'autre grande étape essentielle à la fonctionnalité de ce système est la calibration. Une bonne calibration est d'une très grande importance. Si la calibration est mauvaise, la carte de profondeur, qui se base là-dessus, sera évidemment mauvaise et le résultat final le sera tout autant. Mais une bonne calibration est difficile à obtenir. Beaucoup de recherche s'est déjà faite et se fait encore là-dessus aujourd'hui pour la rendre complètement

automatique et la plus précise possible, tout en n'ayant aucun renseignement a priori sur la scène (réf. [20], [1], [29]). Travailler à améliorer la calibration dans notre système aurait constitué un projet de maîtrise en soi. Or, nous avons décidé de concentrer nos efforts sur la fabrication de la carte de profondeur et de nous en remettre, pour la calibration, à un logiciel commercial : MatchMover de RealViz (voir en annexe pour les détails complets sur la version du logiciel). Si les résultats obtenus par ce logiciel ne sont ni parfaits (il y a parfois des aberrations incroyables dans le "tracking" !), ni même prévus pour obtenir une précision au niveau pixel tel que nous avons besoin pour la fabrication de la carte de profondeur, il reste néanmoins que la calibration obtenue est assez bonne pour être utilisable. C'est le cas du moins si le mouvement de la caméra n'est pas trop compliqué. Il faut dire que la calibration automatique est très complexe et est encore en phase expérimentale (même si le logiciel est vendu commercialement !). Mais si le film a été pris selon les contraintes spécifiées à l'étape #1, alors le résultat de la calibration correspond assez bien à la réalité. MatchMover utilise la méthode de suivi de points ("feature tracking") pour essayer de déduire le mouvement de la caméra. Cette méthode sera décrite plus en détail au chapitre 3.

### *2.3 Etape #3 : Modéliser l'objet virtuel*

Il existe plusieurs moyens de modéliser un objet virtuel en 3D et d'en faire un rendu. Pour notre part, nous avons décidé d'utiliser le logiciel XSI de SoftImage (voir en annexe pour les détails complets sur la version du logiciel). La technique est simple. D'abord, il faut importer le mouvement de caméra déduit de la calibration par le logiciel MatchMover (voir étape #2). SoftImage étant un très gros joueur dans l'industrie de l'infographie, les fichiers .xsi (qui sont lus par XSI) sont devenus un standard dans lequel plusieurs compagnies de logiciels liés à l'infographie ont décidés d'exporter leurs propres fichiers. C'est notamment le cas de RealViz. MatchMover permet d'exporter le résultat de la calibration dans un fichier .xsi, lequel peut être

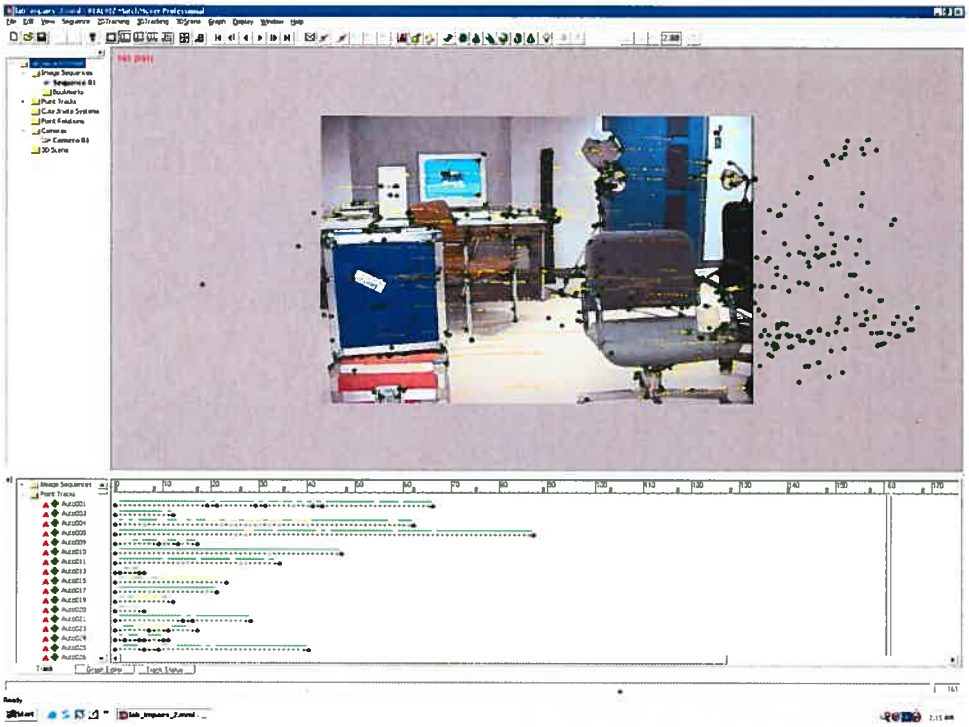


FIG. **2.2. Le logiciel "MatchMover".**

lu très facilement par XSI. A partir de là, il ne reste plus qu'à dessiner un modèle 3D de l'objet virtuel que l'on veut inclure dans la scène et à le positionner où on le veut par rapport à la position de la caméra vidéo. Pour aider à le positionner, on peut travailler en mode *Rotoscope* et en se fiant au suivi de points que nous a fourni MatchMover. Pour avoir une plus grande précision, il est possible de demander à MatchMover de suivre des points précis où sera placé l'objet virtuel pour ainsi pouvoir le positionner exactement à l'endroit souhaité. En mode *Rotoscope*, on obtient alors notre film original en arrière plan avec notre objet virtuel "collé" par-dessus. La scène est considérée comme un écran plat à une distance infinie. Puisque l'objet virtuel cache complètement la scène, si aucun élément de la scène n'est considéré en réalité comme étant devant l'objet virtuel, le traitement pourrait s'arrêter là. Un simple rendu à cette étape et le tour est joué. Dans notre méthode, il faut fabriquer une carte de profondeur (étape #4) dans le but de trouver les éléments de la scène

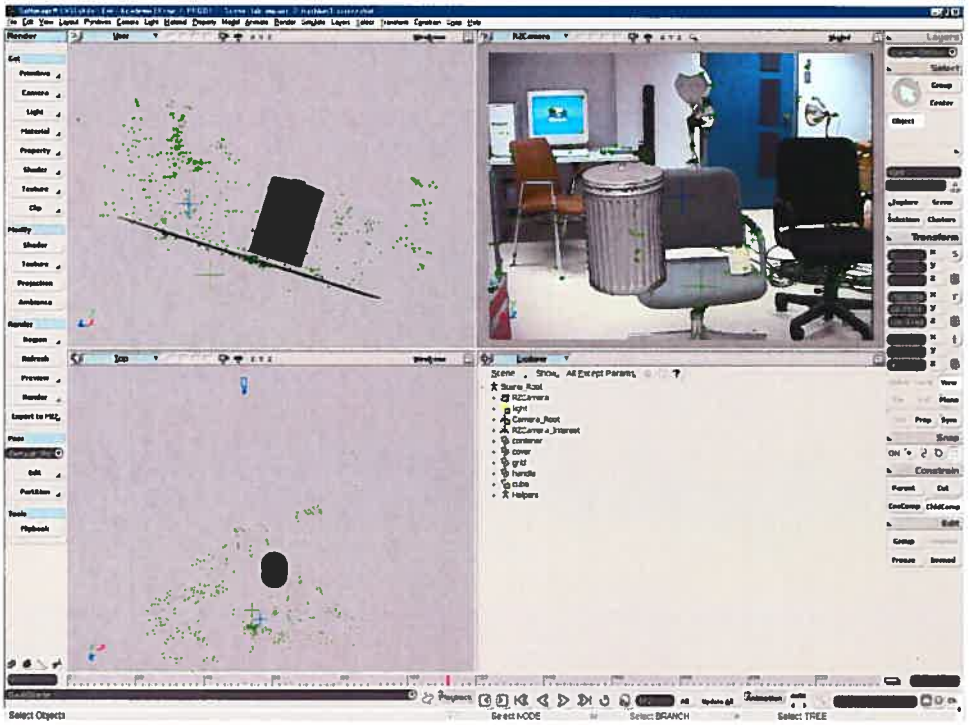


FIG. **2.3. Le logiciel "XSI".**

qui se trouvent devant l'objet virtuel pour ainsi ne pas les effacer par ce dernier. Avant de poursuivre, il faudra sortir, pour toutes les images de la scène, deux types de fichiers : le rendu de l'objet virtuel seulement (pas avec la scène à l'arrière) ainsi qu'un fichier .Zpic, lequel contient la profondeur par rapport à la caméra vidéo de tous les pixels du rendu. Si un pixel ne touche à aucun objet virtuel, il est considéré à une très grande profondeur, habituellement fixé à 10 000 par XSI. C'est la comparaison entre la profondeur lue dans ce fichier et celle obtenue par la carte de profondeur qui va déterminer, à l'étape de l'intégration (étape #5), lequel entre l'objet virtuel et la scène sera caché par l'autre.

### *2.4 Etape #4 : Fabriquer la carte de profondeur*

Cette étape constitue le coeur de notre travail et, évidemment, du présent mémoire. Les chercheurs à travers le monde ont exploré plusieurs avenues de recherche dans ce domaine. Certains ont travaillé sur des couches ("layers") (réf. [3], [14]). Ils considèrent que tous les éléments de la scène appartiennent à quelques plans 2D, lesquels sont positionnés dans la scène 3D. Cette technique a l'avantage de bien séparer les objets se trouvant à des profondeurs différentes. De plus cette technique assure l'intégrité des éléments : tous les pixels d'un élément associé à une couche sont considérés à la profondeur de cette couche. On n'aura donc pas de pixel isolé se trouvant à l'intérieur d'un objet de la scène et qui sera considéré à une profondeur différente du reste de cet objet. Ceci est un avantage considérable. Finalement, le temps de traitement pour cette technique est relativement court. Mais cette technique comporte aussi son lot de désavantages. Premièrement, le nombre de couches qui composent la scène doit être choisi par l'utilisateur, ce qui limite son côté "automatique". Si on décide de garder le même nombre de couches pour toutes les scènes (afin de rendre la technique vraiment automatique) cela va donner des résultats désastreux pour les scènes qui auraient un grand écart entre le nombre de couches idéal et le nombre de couches choisi. De toute façon, représenter une scène 3D par des couches ne représente en rien la réalité. De plus, cette technique limite les possibilités. Supposons que l'on veuille insérer un poteau virtuel qui passe entre les branches d'un arbre, de sorte que certaines branches cachent le poteau alors que d'autres sont cachées par celui-ci. Puisque l'on travaille par couches, l'arbre en entier sera considéré comme faisant partie d'une seule et même couche. Cette scène sera alors impossible à réaliser : le poteau cachera toutes les branches de l'arbre ou, au contraire, le poteau sera complètement caché par celles-ci. C'est pour ces raisons que nous avons décidé de ne pas retenir cette solution. Nous avons plutôt décidé d'y aller avec une méthode de stéréo dense, c'est-à-dire que l'on trouve une profondeur pour chaque pixel de l'image. Il existe

une multitude d'algorithmes de stéréo dense pour estimer la profondeur à partir de deux ou plusieurs images (réf. [22], [18], [19], [4]). Mais elles ont toutes un même désavantage : contrairement aux méthodes par couches, l'intégrité de la profondeur d'un élément de la scène n'est pas assurée. Avec la méthode de recherche directe (la plus simple), on se retrouve souvent avec des pixels au milieu de certains objets à des profondeurs complètement incompatibles. Certains pixels peuvent être considérés trop loin, comme s'il y avait un trou dans l'objet, d'autres trop près, comme s'il y avait une grosse bosse. Cela peut être dû au bruit dans l'image, à l'imprécision de la calibration, etc. Pour parer à ce problème, les autres méthodes (autres que la recherche directe) ont introduit une contrainte de lissage. Le lissage exprime l'hypothèse que la profondeur d'un pixel est similaire à celle de ses voisins. Quand on teste différentes profondeurs possibles pour un pixel, s'il y a une ambiguïté dans le choix de la profondeur, les pixels voisins détermineront ce choix de par leur influence. Le lissage c'est la quantité d'influence exercée par les pixels voisins. Plus le lissage est fort, plus l'influence sera grande. Néanmoins, si les coûts de correspondance ne suggèrent qu'une seule profondeur possible pour un pixel, il ne se laissera pas influencer par les pixels voisins. En fait, c'est plutôt ce pixel qui risque d'influencer ses pixels voisins sauf si la profondeur de ceux-ci est tout aussi fiable. Ceci est très important car il est bien possible que des éléments très éloignés dans la scène n'occupent qu'un seul pixel et donc, soient à une profondeur différente de ses voisins. Malheureusement, ce niveau de lissage doit être défini par l'utilisateur car il doit s'ajuster au contenu de l'image. Si le lissage est trop bas, il y aura des pixels incohérents. Si le lissage est trop fort, l'influence sera trop forte, et on perdra alors les détails dans l'image, c'est-à-dire que des petits éléments seront considérés à la même profondeur que leurs voisins même si ce n'est pas le cas en réalité. Le problème du choix du niveau de lissage est un problème ouvert de la vision par ordinateur. Une contribution de ce mémoire est de proposer une solution qui permet d'obtenir une carte de profondeur s'approchant du meilleur lissage obtenu en pratique, non pas en essayant tous les lissages possibles et en choi-

sissant le meilleur, mais en ne faisant que deux cartes de profondeur : une à un lissage minimal (1) et l'autre à un lissage maximal (1 000 000). Cette technique est parfaitement automatique, ne nécessitant aucune décision de l'utilisateur, et fonctionne avec tous les types de scènes sans informations a priori sur celles-ci. Nous avons nommé cette technique "consistance de profondeur" et les détails se trouvent au chapitre 6. Un autre désavantage des méthodes à stéréo dense est la lenteur du traitement. Cependant, il existe des moyens d'accélérer le processus, notamment par fenêtrage ("clipping") autour de l'objet virtuel et par parallélisme (chapitre 7). Au chapitre 4, nous verrons aussi trois méthodes de stéréo dense. D'abord la recherche directe, la plus simple et la plus rapide. Ensuite l'algorithme de programmation dynamique, lequel présente des résultats supérieurs à la recherche directe en introduisant un lissage sur les lignes (réf. [9], [24], [5]). Enfin, nous verrons l'algorithme de flot maximum qui impose un lissage sur les lignes et les colonnes en même temps (réf. [22]). L'algorithme de flot maximum permet d'obtenir de très bons résultats qui sont reconnus parmi les meilleurs en stéréo dense (réf. [26]). C'est pourquoi nous l'avons choisi pour notre méthode. Nous y avons apporté quelques améliorations. Il y a d'abord la technique "consistance de profondeur". Nous avons aussi développé la technique "max-gauche-droite", qui permet de mieux définir les contours des objets, au lieu du flou dû aux demi-occlusions ("half-occlusions") (réf. [7]). L'élaboration de ces deux techniques a été le coeur de cette maîtrise et elles seront définies en détail aux chapitres 5 et 6.

## *2.5 Etape #5 : Intégrer l'objet virtuel dans la scène filmée à partir de la carte de profondeur*

C'est dans cette étape que l'on produit l'image finale en décidant, pour chaque pixel de cette image finale, si on doit le colorer de la couleur du pixel de l'image originale, ou de la couleur du rendu de l'objet virtuel. L'élément de décision est la carte de profondeur. Si la profondeur trouvée sur la carte de profondeur est plus

grande que celle qui correspond au même pixel dans le fichier des profondeurs du rendu (.Zpic (voir étape #3)), alors on prend la couleur du pixel de la scène. Sinon, on prend la couleur du pixel du rendu. Il s'agit de la plus simple des étapes et peut, dans notre cas, être considérée comme étant sans faille. Si la carte de profondeur est parfaite, le résultat final sera parfait. Mais si la carte de profondeur contient certaines erreurs, elles ne seront pas réparées et se réflèteront dans le résultat final. Tout dépend donc des étapes antérieures.

Ceci est vrai dans notre cas puisque nous ne visons un réalisme que dans la précision géométrique et la relation de profondeur. Pour obtenir une image d'un meilleur réalisme, il faudrait utiliser des techniques d'"antialiasing", trouver avec précision l'éclairage de la scène, tenir compte de l'ombre projetée par l'objet virtuel, etc. Ceci aurait compliqué inutilement notre méthode car elle n'est pas destinée à produire une image finale prête à être insérée dans un film mais plutôt à produire un premier rendu qui sera retouché par un artiste. Le fait que notre méthode soit automatique devrait néanmoins permettre d'accélérer le travail de l'artiste de façon notable. Evidemment, il serait préférable que notre méthode produise des images finales sans avoir besoin d'être retouchées, mais obtenir les cartes de profondeur d'une qualité suffisante n'est pas encore possible avec les méthodes actuelles (même avec notre variante du flot maximum !).

Les chapitres suivants décrivent en détail les principales composantes de notre méthode de réalité augmentée : la calibration de la caméra (chapitre 3), la fabrication de la carte de profondeur par stéréo dense (chapitre 4), la technique "max-gauche-droite" (chapitre 5), la technique de "consistance de profondeur" (chapitre 6), et finalement, les résultats expérimentaux ainsi que diverses techniques pour accélérer le traitement (chapitre 7). Le tout sera suivi d'une conclusion sur notre méthode avec notamment, des suggestions de recherche afin de l'améliorer d'avantage.

Chapitre 3

# CALIBRATION DE LA CAMÉRA

Calibrer une caméra consiste à trouver tous les paramètres reliés à cette caméra. Ceci permettra d'établir la relation entre la caméra et la scène. Si on ne connaît pas les paramètres de la caméra, on ne pourra rien déduire de la scène. Mais si on connaît ces paramètres, on pourra trouver la projection sur l'image de chaque point 3D de la scène (réf. [30]). Ceci est absolument essentiel pour fabriquer la carte de profondeur. Ces paramètres se décomposent en deux parties : les paramètres internes et les paramètres externes.

Les paramètres internes représentent tout ce qui est relié à la caméra elle-même, sans tenir compte de sa position dans le monde. Ils comprennent la distance focale ($f$), les coordonnées pixels du centre de l'image ($o_x$,$o_y$), la taille des pixels ($s_x$,$s_y$) et, s'il n'est pas négligeable, le coefficient de distorsion radiale ($k$). En ce qui nous concerne, à cause de la lentille standard (qui déforme très peu l'image) de la caméra vidéo que nous avons utilisée, ce coefficient de distorsion radiale sera effectivement négligé. Habituellement, ces paramètres sont déjà connus avant de commencer à filmer. Le manufacturier fournit ces informations avec les caméras. Ils peuvent alors être considérés comme étant presque parfaits. Si nous n'avons pas les données techniques avec la caméra, on peut toujours les déduire en faisant quelques tests en laboratoire. Ceci pourrait théoriquement conduire à des sources d'erreurs mais étant donné que toutes les caméras sont construites selon l'un des quelques standards disponibles, il est facile de déduire quel standard a été utilisé pour sa fabrication. D'ailleurs, le logiciel MatchMover déduit ces paramètres par simple suivi de points ("feature tracking"). Les paramètres internes sont fixes tout au long de la séquence vidéo. Une seule exception : si on change le "zoom" de la caméra pendant la séquence, cela fait varier la

distance focale qui est un paramètre interne.

Les paramètres externes représentent la position et l'orientation de la caméra dans le monde. Ils comprennent le vecteur de translation et la matrice de rotation de la caméra par rapport au point de référence (0,0,0) de la scène. Contrairement aux paramètres internes, les paramètres externes varient tout au long de la séquence vidéo. Ce sont ces paramètres qui sont les plus difficiles à trouver et qui sont les sources de la majorité des erreurs de calibration.

MatchMover utilise une méthode de suivi de points ("feature tracking") pour calibrer la caméra. Cette méthode consiste à trouver les points les plus facilement identifiables dans la scène, à les suivre et, en tenant compte de tous ces points saillants ("feature points") et de leur déplacement d'une image à l'autre, déduire les paramètres de la caméra (réf. [30]).

D'abord, il faut trouver les points les plus facilement identifiables dans la scène, les fameux points saillants ("feature points"). Pour être facilement identifiable et être facile à suivre d'une image à l'autre, un point doit se trouver à la frontière d'une forte différence de couleurs dans l'image. Par exemple, tout le contour d'un objet noir sur un fond blanc sera facile à suivre, et plus particulèrement les coins de cet objet. Un coin dans la texture d'un objet peut aussi faire un excellent point saillant. La qualité de ces points saillants dépendra de la facilité à les suivre d'une image à l'autre. Ici encore, comme dans tous les algorithmes de vision et de traitement d'images, le bruit est un gros problème. Le bruit peut engendrer des fortes différences de couleurs là où il n'y en a pas en réalité. Ou, au contraire, le bruit peut réduire une différence de couleurs bien réelle. On peut néanmoins supposer à juste titre que le bruit n'engendrera pas les plus fortes différences de couleurs de l'image, ni n'annulera complètement les plus fortes différences de couleurs de la scène. Sinon, on ne pourra pas tirer grand chose de ces images. Donc si cette supposition est juste, il suffit de conserver les meilleurs points saillants. Une des techniques les plus simples que l'on peut employer est de passer un filtre (Laplacien, Roberts ou Sobel (réf. [31], [32])) dans l'image et de ne

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

FIG. **3.1. Filtre Laplacien 3x3**

conserver que les points qui auront répondu le plus fortement à ce filtre. L'application de ce filtre sur l'image par convolution (réf. [32], [31]) va produire une valeur pour chaque pixel. Etant donné que le signe du coefficient du pixel central (le pixel que l'on traite) est différent du signe des coefficients de ses pixels voisins, plus le résultat (en valeur absolue) sera élevé, plus ce pixel sera différent de ses voisins. Ce sont les pixels ayant obtenu le plus haut résultat qu'il faudra conserver comme points saillants. Pour ce faire, il faudra tester ces valeurs par rapport à un seuil pré-déterminé. Si le seuil est trop haut, il n'y aura alors pas assez de points conservés. Si le seuil est trop bas, des mauvais points, dus au bruit par exemple, seront conservés, ce qui est pire encore. Il vaut mieux donc de mettre ce seuil à un niveau relativement élevé.

La méthode précédente pour trouver les points saillants fonctionne très bien pour une image. Cependant, notre but est en fait de trouver des points saillants stables dans le temps, c'est-à-dire que l'on peut suivre d'une image à l'autre. Un excellent point saillant qui ne serait détecté que dans une seule image n'aurait aucune valeur. Il faut donc que les points soient facilement identifiables dans plusieurs images consécutives. Pour suivre un point d'une image à l'autre, il faut d'abord le trouver comme point saillant dans une première image et ensuite le suivre en essayant de prédire son emplacement dans la prochaine image et ce, jusqu'à temps que le point ne soit plus détectable (réf.[30]). Supposons que nous avons trouvé un pixel correspondant à un point saillant dans l'image #1. Il faudra chercher ce point saillant dans l'image #2 à l'endroit où il est probablement en tenant compte du mouvement précédent de ce point saillant (s'il a été trouvé dans les images précédentes). Pour le chercher, on va

tester la couleur du pixel de l'image #1 avec la couleur du pixel de l'image #2 en faisant la somme des différences carrées (réf. [30], [22]). Avec des images couleurs, cela va donner :

$$\sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \tag{3.1}$$

où $r_1, g_1, b_1$ et $r_2, g_2, b_2$ sont les couleurs $RGB$ des pixels des images #1 et #2 respectivement.

Si le résultat ne dépasse pas un certain seuil, on considère alors que l'on a retrouvé le point saillant. Si par contre ce résultat dépasse le seuil fixé, on cherche autour de ce pixel en retestant avec l'équation 3.1. La caméra peut avoir changé de direction ou de vitesse et c'est pour cela que l'on doit tester dans le voisinage de l'endroit présumé. Si on ne trouve toujours pas de résultat inférieur au seuil après un nombre maximum d'essais prédéterminé, on considère ce point saillant comme étant caché, dû à une occlusion. On va quand même continuer de tester ce point saillant sur les prochaines images à l'endroit où il devrait être en tenant compte d'un mouvement de caméra constant. Si après plusieurs images on ne l'a toujours pas retrouvé, on le considère alors perdu. C'est pourquoi, pour faciliter la calibration, il faut que la caméra ait un mouvement plutôt continu, c'est-à-dire une translation toujours dans la même direction avec une vitesse constante. MatchMover suit les points saillants sur une moyenne d'environ 5 à 30 images avant de les perdre. Mais à chaque image, il essaie de trouver de nouveaux points saillants qui vont constituer des points de départ pour des nouveaux suivis de points. Il arrive parfois qu'il suit un même point saillant comme étant deux points saillants à deux moments différents du vidéo parce qu'il l'a perdu au cours d'une série d'images centrales. La corrélation qu'il ne fait pas entre les deux traces l'empêche alors de profiter de cette information pour calibrer la caméra plus précisément par la suite. Mais il arrive parfois qu'il suit un point saillant d'image en image alors qu'il se trompe complètement et est en train de suivre différents points saillants en les considérant comme s'ils étaient un seul et même

point. Ceci est beaucoup plus grave car cela produit une trace aberrante ("outlier") qui peut complètement dérégler les calculs pour la calibration. On préfère toujours peu de points fiables à de nombreux points peu fiables. C'est pourquoi le seuil (pour tester si on a retrouvé le point saillant) doit être élevé. Ce phénomène arrive souvent quand la scène n'a pas assez de texture dans l'image, le logiciel abaisse alors ce seuil en essayant de trouver des traces et c'est alors que la calibration obtenue est très mauvaise. Les reflets sont aussi une autre cause de ces phénomènes car ils vont suivre l'angle formé par la caméra et la source de lumière, au lieu des objets sur lesquels ils se reflètent. Comme dans la plupart des algorithmes de vision, on fait l'hypothèse que les objets de la scène sont opaques et mats (Lambertiens (réf. [10])). On fait aussi l'hypothèse qu'un pixel est suffisamment petit pour ne contenir qu'un seul élément de la scène et donc qu'une seule profondeur lui est associée. Mais dans la pratique, dans les scènes "normales", ces hypothèses sont habituellement fausses.

Une fois que toutes les traces ont été obtenues, on peut évaluer la position relative des points saillants entre eux et avec la caméra. Un point saillant se déplaçant rapidement d'une image à l'autre sera considéré comme étant près de la caméra, alors qu'un point saillant restant immobile d'une image à l'autre sera considéré comme étant très éloigné de la caméra. Mais il faut éviter de tenir compte des possibles aberrations ("outliers"). C'est après une analyse des traces que l'on peut déterminer la position et l'orientation de la caméra et la position des points saillants trouvés dans la scène. Evidemment, ces distances sont relatives. On ne peut pas savoir combien de mètres séparent le point #1 de la caméra. Mais on peut savoir, par exemple, que le point #2 est deux fois plus éloigné de la caméra que le point #1. Pour avoir des distances absolues, il suffit d'avoir une seule distance que l'on connaît entre un point saillant et un autre point saillant ou la caméra, et on peut alors déduire toutes les autres distances. Evidemment, pour que les points saillants soient évalués à leur vraie distance relative, seule la caméra doit se déplacer d'une image à l'autre. Si un objet se déplace dans le même sens que la caméra, il sera considéré comme étant plus loin qu'en réalité, alors

que s'il se déplace dans le sens contraire à la caméra, il sera considéré plus près qu'en réalité.

### *3.1 Conversion du .xsi en matrice de projection*

MatchMover retourne un fichier .xsi qui contient le résultat de la calibration de la caméra avec les éléments suivants : position de la caméra, position du point d'intérêt et roulis de la caméra. Avant de passer à l'étape de la fabrication de la carte de profondeur, nous avons besoin de la matrice de projection pour passer d'un point 3D de la scène vers un pixel 2D de l'image (figure 3.2 et réf. [30]), tel qu'exprimé par la relation

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = M \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

où $X$, $Y$, $Z$ sont des coordonnées 3D du monde, $u/w$ la coordonnée $x$ de l'image (coordonnée pixel), $v/w$ la coordonnée $y$ de l'image et $M$ la matrice de projection 4x4.

Il faut donc déduire cette matrice $M$ à partir des informations contenues dans le fichier .xsi fourni par MatchMover. D'abord, on peut diviser la matrice $M$ en un produit de deux matrices

$$M = M_{ext} \cdot M_{int} \tag{3.3}$$

où $M_{ext}$ est la matrice des paramètres externes et $M_{int}$ la matrice des paramètres internes.

Afin de trouver les axes du système de coordonnées de la caméra, il faut avoir défini un vecteur $V_{up}$. Ce vecteur est défini par MatchMover comme étant

$$V_{up} = Y_{world} \tag{3.4}$$

où $Y_{world}$ est l'axe $Y$ du monde.

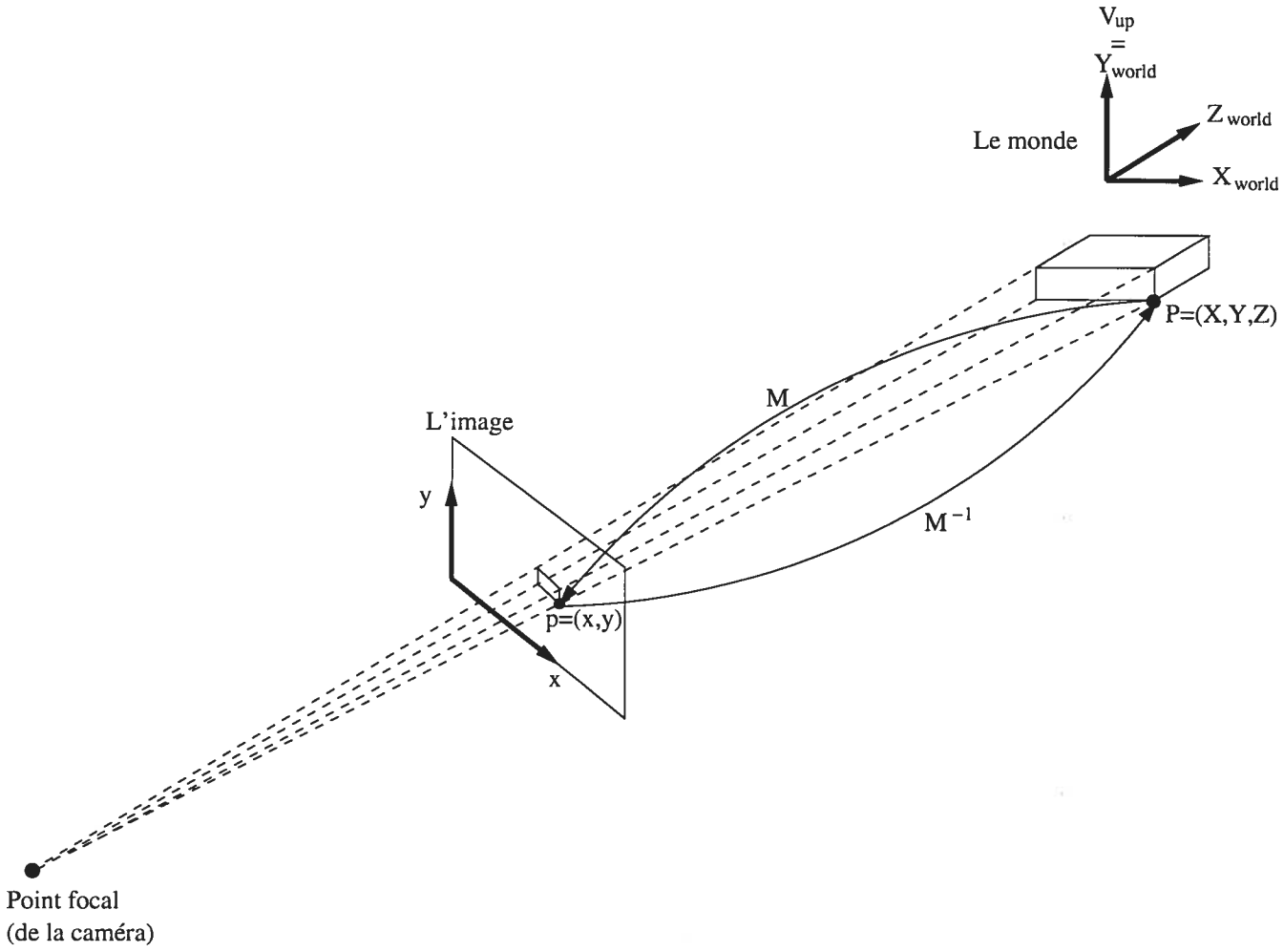



FIG. 3.2. **Systèmes de coordonnées. La matrice $M$ permet la projection du point $P$ sur l'image (pour obtenir le point $p$), alors que la matrice $M^{-1}$ permet de passer d'un point de l'image vers le monde.**

Maintenant il faut calculer $M_{ext}$. Cette matrice se divise à son tour en un produit de deux matrices

$$M_{ext} = M_{rot} \cdot M_{trans} \tag{3.5}$$

où $M_{rot}$ est la matrice de rotation de la caméra et $M_{trans}$ la matrice de translation.

Commençons par trouver la matrice $M_{rot}$ qui représente la rotation de la caméra selon le système de coordonnées du monde. Comme il a été mentionné précédemment, le fichier .xsi nous a fourni la position de la caméra dans le monde, la position du point d'intérêt et le roulis de la caméra. Le roulis $\theta$ s'effectue autour de l'axe $Z$ de la caméra. Il faut donc effectuer le roulis autour de l'axe $Z$ du monde (car il est beaucoup plus facile d'effectuer une rotation d'un certain angle sur les axes du monde) et calculer la matrice $R$ qui va permettre d'aligner le nouveau système d'axes avec roulis sur le système d'axes de la caméra sans roulis. Nous aurons alors le vrai système de coordonnées de la caméra (incluant le roulis) lequel devra être exprimé en une matrice de rotation en fonction du système de coordonnées du monde ($M_{rot}$).

La matrice de rotation autour du $Z$ du monde est la suivante (réf. [10]) :

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

où $\theta$ représente, dans notre cas, l'angle de roulis.

Les axes du nouveau système de coordonnées correspondant au monde avec roulis seront :

$$X_{roll} = R_z \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ 0 \\ 0 \end{bmatrix} \tag{3.7}$$

$$Y_{roll} = R_z \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta) \\ 0 \\ 0 \end{bmatrix} \tag{3.8}$$

$$Z_{roll} = R_z \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.9}$$

Nous allons maintenant trouver la matrice $R$ qui permettra d'aligner ces axes sur les axes du système de la caméra avant le roulis. Les axes du système de coordonnées de la caméra avant roulis sont définis comme suit :

$$Z_c = \frac{P_{cam} - P_{inter}}{\|P_{cam} - P_{inter}\|} \tag{3.10}$$

$$X_c = \frac{V_{up} \times Z_c}{\|V_{up} \times Z_c\|} \tag{3.11}$$

$$Y_c = \frac{Z_c \times X_c}{\|Z_c \times X_c\|} \tag{3.12}$$

où $P_{cam}$ est la position de la caméra, $P_{inter}$ est la position du point d'intérêt et $\times$ est le produit vectoriel.

Calculons ensuite la matrice $R$. Puisque c'est une matrice orthogonale, on peut utiliser la propriété suivante : la matrice $R$ est tout simplement formée des trois axes de la caméra mis en colonnes (réf. [10]).

$$R = \begin{bmatrix} & & & 0 \\ X_c & Y_c & Z_c & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Les axes du vrai système de coordonnées de la caméra (avec le roulis) peuvent alors être calculés.

$$X_{cam} = R \cdot X_{roll} \tag{3.14}$$

$$Y_{cam} = R \cdot Y_{roll} \tag{3.15}$$

$$Z_{cam} = R \cdot Z_{roll} \tag{3.16}$$

Ainsi la matrice de rotation de la caméra exprimée selon le système de coordonnées du monde ($M_{rot}$) peut alors être calculée en utilisant encore les propriétés de la matrice orthogonale.

$$M_{rot} = \begin{bmatrix} \begin{bmatrix} X_{cam}^T \end{bmatrix} \\ \begin{bmatrix} Y_{cam}^T \end{bmatrix} \\ \begin{bmatrix} Z_{cam}^T \end{bmatrix} \\ 0 \quad 0 \quad 0 \quad 1 \end{bmatrix} \tag{3.17}$$

Maintenant que nous avons trouvé la matrice de rotation de la caméra ($M_{rot}$), il faut trouver la matrice de translation de la caméra ($M_{trans}$)

$$M_{trans} = \begin{bmatrix} 1 & 0 & 0 & -Px_{cam} \\ 0 & 1 & 0 & -Py_{cam} \\ 0 & 0 & 1 & -Pz_{cam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

où $Px_{cam}$, $Py_{cam}$, $Pz_{cam}$ sont les coordonnées $x$, $y$, $z$ de la position de la caméra dans le monde.

Et finalement, nous avons tous les éléments pour calculer la matrice des paramètres externes ($M_{ext}$) selon l'équation 3.5.

Maintenant il faut calculer la matrice des paramètres internes de la caméra ($M_{int}$)

$$M_{int} = \begin{bmatrix} -\frac{focal}{s_x} & 0 & o_x & 0 \\ 0 & -\frac{focal}{s_y} & o_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

avec les éléments suivants :

$focal$ = la distance focale de la caméra.

$s_x$, $s_y$ = la taille en largeur ($s_x$) et en hauteur ($s_y$) d'un pixel. Mais puisque l'on travaille en distance relative, on peut fixer $s_x$ à 1 et $s_y$ devient alors équivalent au ratio ($pix\_ratio$) de la taille d'un pixel (hauteur / largeur). Ce ratio est fourni dans les spécifications techniques de la caméra.

$o_x$, $o_y$ = la coordonnée pixel en $x$ et $y$ du centre de l'image. Ce centre est la moitié du nombre de pixels en largeur ($Xsize$) et en hauteur ($Ysize$) de l'image.

Sachant cela, la matrice des paramètres internes devient donc :

$$M_{int} = \begin{bmatrix} -focal & 0 & \frac{Xsize}{2} & 0 \\ 0 & -focal * pix\_ratio & \frac{Ysize}{2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

Finalement, la matrice $M$ qui permet de passer d'un point 3D à une coordonnée pixel de l'image est calculée par l'équation 3.3.

## *3.2 La matrice de dé-projection*

L'algorithme pour calculer la carte de profondeur a aussi besoin de la matrice de dé-projection pour fonctionner, c'est-à-dire la matrice qui permettra de passer d'un pixel de l'image à un point 3D du monde. En fait, un pixel d'une image correspond à une infinité de points 3D du monde qui forment la droite "point_focal - pixel". Tout ceci sera expliqué en détail au chapitre 4. La matrice de dé-projection ($M_{deproj}$) est tout simplement l'inverse de la matrice $M$ trouvée plus tôt. Elle pourrait être calculée en faisant tout simplement :

$$M_{deproj} = M^{-1} \tag{3.21}$$

Seulement, calculer l'inverse d'une matrice n'est pas facile et demanderait beaucoup de temps de calcul à l'ordinateur. Or, il existe une façon beaucoup plus simple de calculer $M_{deproj}$ :

$$M_{deproj} = M_{ext}^{-1} \cdot M_{int}^{-1} \tag{3.22}$$

Commençons par calculer $M_{ext}^{-1}$.

$$M_{ext}^{-1} = M_{rot}^{-1} \cdot M_{trans}^{-1} \tag{3.23}$$

Puisque la matrice des paramètres externes $M_{rot}$ est orthogonale (réf. [15])

$$M_{rot}^{-1} = M_{rot}^{T} \tag{3.24}$$

où $M_{rot}^{T}$ est la transposée de la matrice $M_{rot}$.

Pour la matrice de translation inverse ($M_{trans}^{-1}$), il suffit de faire la translation qui permet de passer de l'origine à la position de la caméra (pour $M_{trans}$, la translation était négative, on partait de la caméra pour aller à l'origine)

$$M_{trans}^{-1} = \begin{bmatrix} 1 & 0 & 0 & Px_{cam} \\ 0 & 1 & 0 & Py_{cam} \\ 0 & 0 & 1 & Pz_{cam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

où $Px_{cam}$, $Py_{cam}$, $Pz_{cam}$ sont les coordonnées $x$, $y$, $z$ de la position de la caméra dans le monde.

Maintenant que $M_{ext}^{-1}$ peut maintenant être calculée à partir de l'équation 3.23, il faut trouver $M_{int}^{-1}$. Le centre $(\frac{Xsize}{2}, \frac{Ysize}{2})$ de l'image est situé sur la colonne de translation. Il faut donc inverser le signe des éléments de cette colonne. De plus, la diagonale de la matrice 4x4 contrôle l'agrandissement du point avec lequel on effectuera le produit scalaire. Il faut donc aussi inverser l'agrandissement de la matrice

$M_{int}$.

$$M_{int}^{-1} = \begin{bmatrix} -\frac{1}{focal} & 0 & \frac{(\frac{Xsize}{2})}{focal} & 0 \\ 0 & -\frac{1}{focal*pix\_ratio} & \frac{(\frac{Ysize}{2})}{focal*pix\_ratio} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

où $focal$ est la distance focale, $Xsize$ et $Ysize$ la taille de l'image et $pix\_ratio$ le ratio de la taille d'un pixel (hauteur / largeur).

Enfin, nous pouvons calculer $M_{deproj}$ avec l'équation 3.22.

La calibration est alors terminée. Ainsi, nous obtenons à la sortie une matrice de projection et une matrice de dé-projection. Le calcul de ces deux matrices est effectué pour chaque image de la séquence vidéo. Cette séquence de matrices constitue en fait la "trajectoire" de la caméra et permettra de fabriquer la carte de profondeur, étape qui sera étudiée dans le prochain chapitre.

Chapitre 4

# FABRICATION D'UNE CARTE DE PROFONDEUR PAR STÉRÉO DENSE

Une image possède deux dimensions : la hauteur et la largeur. Lorsque nous regardons une photographie, nous déduisons la position 3D approximative des objets parce que nous connaissons, de mémoire, leurs tailles réelles. Mais il n'y a aucune information sur une seule image pour déduire cette troisième dimension qu'est la profondeur. Si nous étions confrontés à une seule photo d'une scène complètement insolite, on ne pourrait déduire ni la distance des objets par rapport à la caméra, ni même leurs tailles réelles (ambiguïté à savoir si un objet est petit et proche ou très gros et loin). Pour retrouver cette troisième dimension, il nous faut au moins une autre image de la même scène prise d'un endroit différent. Alors, par triangulation, nous pourrons déduire la profondeur des objets de la scène. De la même façon, le système visuel humain permet de voir en trois dimensions grâce à nos deux yeux, en percevant la scène de deux points de vue légèrement différents. Mais pour y arriver, le cerveau doit faire une quantité de calculs complexes afin de déduire la profondeur de la scène à partir des images captées par nos yeux. Même le cerveau humain, qui possède une capacité de calcul bien plus grande que les meilleurs ordinateurs actuels, se trompe parfois dans l'interprétation de la profondeur, notamment dans les cas d'illusions optiques. Ainsi, retrouver la profondeur d'une scène avec précision à partir d'images est extrêmement complexe et nombre d'algorithmes en vision par ordinateur ont été développés pour y arriver. C'est à cela que s'intéresse la stéréoscopie par ordinateur. Comme il a été mentionné au chapitre 2, nous nous sommes intéressés aux algorithmes de stéréo dense qui permettent d'estimer la profondeur de chaque pixel de l'image. Dans ce chapitre, trois algorithmes appliqués à la stéréoscopie par

ordinateur seront étudiés : la recherche directe, la programmation dynamique et le flot maximum.

## *4.1 La triangulation*

On appelle triangulation le calcul de la distance d'un objet à partir de deux points de vue différents qui forment un triangle avec cet objet. Par géométrie, on peut alors déduire la distance de l'objet en question (réf. [30]). Mais pour cela, il faut d'abord mettre en correspondance cet objet à partir des deux points de vue. Si par exemple on photographie une scène avec deux appareils photos placés l'un à côté de l'autre, les deux images obtenues présenteront quelques différences. Les objets qui sont pourtant fixes dans la scène se seront déplacés d'une image à l'autre, à cause du différent point de vue des caméras. Un objet placé près des caméras apparaîtra à deux endroits différents sur les images. Par contre, un objet très éloigné ne se sera pas déplacé d'une image à l'autre, c'est-à-dire qu'il occupera à peu près les mêmes coordonnées pixels dans les deux images. C'est donc en retrouvant les mêmes objets d'une image à l'autre et en mesurant leur déplacement dans l'image que l'on pourra déduire leur distance par rapport à la caméra de référence, et par conséquent, la profondeur de la scène.

## *4.2 La géométrie épipolaire*

Le grand défi de la stéréoscopie est de mettre en correspondance des pixels d'une image à l'autre. Si on connaît la position d'un pixel d'une image dans une autre image, on peut alors déduire sa position 3D dans la scène. Mais pour cela, il faut d'abord connaître la position et l'orientation relative des caméras entre elles. Celles-ci sont obtenues par la calibration (chapitre 3). Prenons par exemple deux caméras côte-à-côte (figure 4.1). Un pixel dans l'image #1 peut être la projection d'un ensemble de points 3D. Cet ensemble, si on le projette dans l'image #2, forme une droite plutôt

qu'un seul point. Cette droite est appelée "droite épipolaire" (réf. [30]). Partons d'un pixel de l'image #1. Ce pixel représente la projection d'un point d'un objet de la scène qui coupe la droite épipolaire de la caméra #1. Mais cet objet peut se trouver n'importe où le long de cette droite. Dans l'image #2, cette droite part du côté où se trouve la caméra #1 et s'achève au point de fuite. C'est le long de cette droite qu'il faut chercher le pixel dans l'image #2 correspondant à la projection du même point de l'objet qui s'est projeté sur l'image #1 (et qui est notre pixel de départ). Le déplacement du pixel dans l'image #2 détermine la distance de l'objet par rapport à la caméra #1. Pour un déplacement de caméras surtout translationnel, un grand déplacement observé correspond à un objet très près des caméras alors qu'un faible déplacement correspond à un objet très éloigné. Pour connaître la distance du point de l'objet par rapport à la caméra #1, il faut tester différentes profondeurs le long de la droite épipolaire et conserver la plus probable.

D'abord il faut dé-projeter le pixel de l'image #1 vers la scène 3D selon la profondeur à tester $z_1$ :

$$P' = \begin{bmatrix} X \\ Y \\ Z \\ H \end{bmatrix} = M_1^{-1} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \\ -\frac{1}{z_1} \end{bmatrix} \tag{4.1}$$

où $P'$ est le vecteur des coordonnées homogènes associées au point 3D $P$ de la scène, $M_1^{-1}$ est la matrice de dé-projection de l'image #1, $x_1$ et $y_1$ sont les coordonnées $x$ et $y$ du pixel de l'image #1 et $z_1$ est la profondeur par rapport à la caméra #1 que l'on veut tester. Le système de coordonnées de la caméra ayant un axe Z qui pointe vers le photographe, la profondeur doit être négative ( $-\frac{1}{z_1}$ et non pas $\frac{1}{z_1}$ ).

Pour obtenir la position 3D du point $P$, il suffit de diviser chacun des éléments

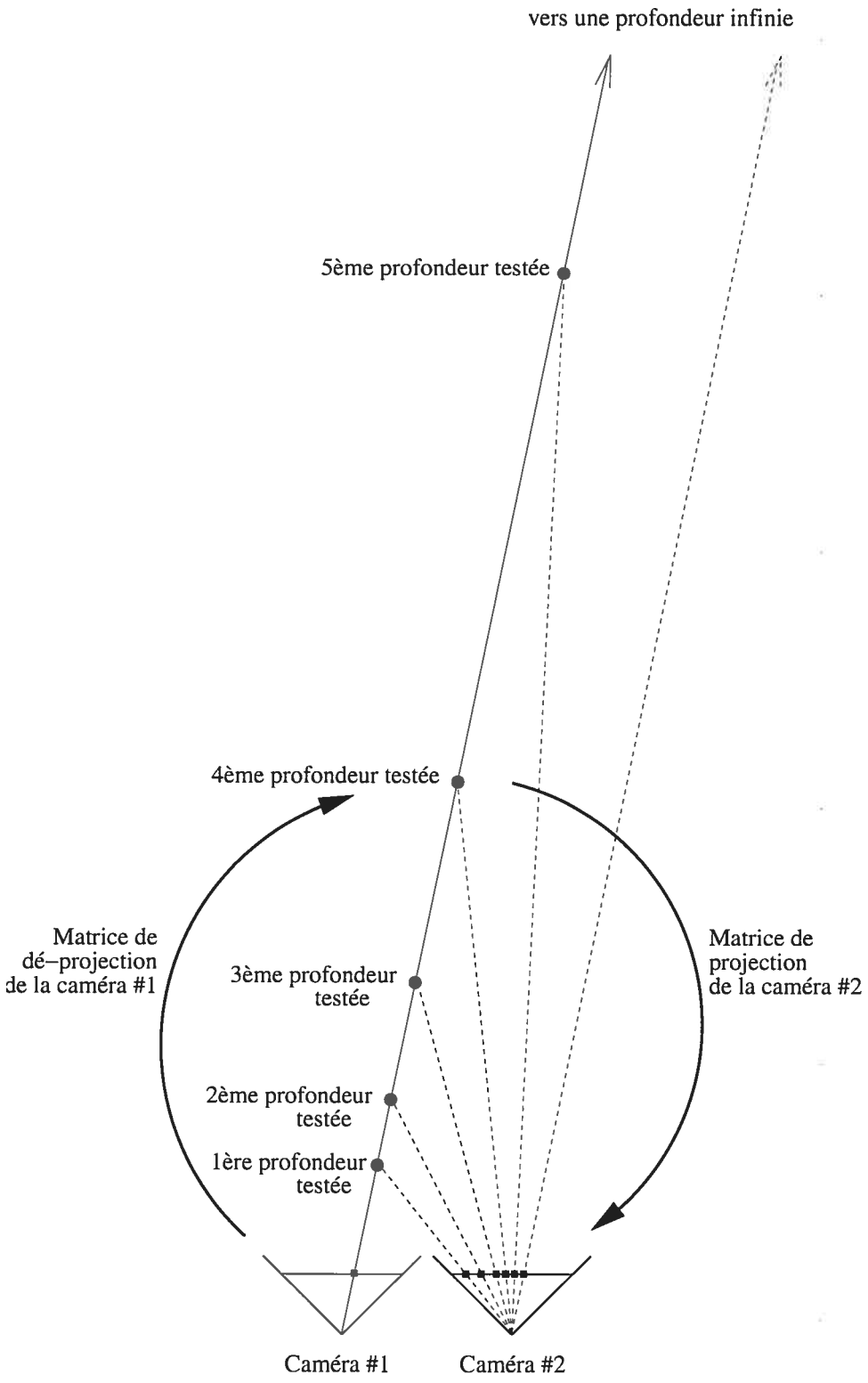



FIG. **4.1. La géométrie épipolaire à deux caméras.**

de $P'$ par le dernier élément de ce vecteur :

$$P = \frac{P'}{H} = \begin{bmatrix} X/H \\ Y/H \\ Z/H \\ 1 \end{bmatrix} = \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{4.2}$$

où $X_P$, $Y_P$ et $Z_P$ représentent les coordonnées 3D du point $P$ dans la scène. Le point $P$ représente le point sur la droite épipolaire qui correspond à la profondeur testée.

Il faut maintenant projeter ce point 3D dans l'image #2 :

$$p'_2 = \begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix} = M_2 \cdot P \tag{4.3}$$

où $p'_2$ est le vecteur des coordonnées homogènes associées aux coordonnées pixel dans l'image #2 et $M_2$ est la matrice de projection de l'image #2.

Pour obtenir les coordonnées pixel $x$ et $y$ de la projection du point 3D $P$ dans l'image #2, il suffit de diviser les 2 premiers éléments du vecteur $p'_2$ par le troisième élément de ce même vecteur :

$$x_2 = \frac{x}{z} \tag{4.4}$$

$$y_2 = \frac{y}{z} \tag{4.5}$$

où $x_2$ et $y_2$ sont les coordonnées pixel $x$ et $y$ correspondant au point $P$ projeté sur l'image #2.

Pour obtenir une carte de profondeur complète, il faut répéter ce calcul pour chaque profondeur à tester et ce, pour chaque pixel de l'image de référence (dans

l'exemple précédent, l'image de référence était l'image #1). Le choix de la meilleure profondeur dépend de la qualité de la mise en correspondance. Celle-ci est calculée en comparant les intensités de chaque couleur (RGB) des points correspondants.

### *4.3 L'interpolation bilinéaire*

Il y une infinité de points 3D sur une droite épipolaire, et donc une infinité de projections. Cependant, une image ne fournit les intensités que pour les positions entières. Les pixels de l'image sont donc la projection que de quelques points 3D le long de cette droite. Pour améliorer la précision, il faudra donc trouver un moyen d'estimer des points de l'image qui se trouvent entre ces pixels. Les lignes et les colonnes de l'image forment une grille où les intersections sont l'emplacement des pixels. La plupart du temps, un point projeté atterrira entre quatre pixels. Si nous nous contentons de prendre la couleur du pixel le plus près, nous perdons alors beaucoup de précision. Pour augmenter cette précision de la profondeur recherchée, on a recours à l'interpolation bilinéaire (réf. [16] [25]). L'interpolation bilinéaire permet d'approximer la couleur d'un point sur l'image qui se trouve entre ces quatre pixels. Cet algorithme fait une moyenne des couleurs des quatre pixels voisins en leur attribuant un poids proportionnel à la distance qui les sépare du point recherché. Par exemple, supposons que nous recherchons la couleur du point $P$ qui se trouve entre quatre pixels (figure 4.2). Il y a plusieurs façons de trouver la solution à ce problème. Voici celle qui a été utilisée dans ce projet. Notez que dans le cas des images RGB, il faudra exécuter cette méthode pour chacune des trois couleurs, rouge, vert et bleu, chacune étant traitée séparément.

Il faut d'abord trouver l'intensité (de la couleur traitée) des pixels voisins :

$$\begin{aligned} x &= \lfloor x' \rfloor \\ y &= \lfloor y' \rfloor \\ a &= I(x, y) \end{aligned}$$

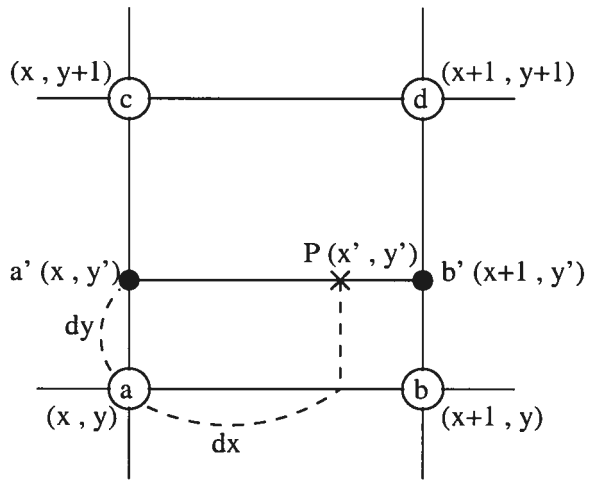



FIG. **4.2. Exemple d'interpolation bilinéaire. On cherche à interpoler le point P. Les pixels a,b,c et d sont connus.**

$$\begin{aligned} b &= I(x+1, y) \\ c &= I(x, y+1) \\ d &= I(x+1, y+1) \end{aligned}$$

où $x'$ et $y'$ sont les coordonnées du point $P$ et $I()$ est l'intensité (de la couleur traitée) de l'image.

Ensuite, il s'agit de faire l'interpolation bilinéaire, proprement dite, du point $P$ :

$$\begin{aligned} d_x &= x' - x \\ d_y &= y' - y \\ a' &= a + (c - a) * d_y \\ b' &= b + (d - b) * d_y \\ P &= a' + (b' - a') * d_x. \end{aligned}$$

En faisant ce calcul pour chacune des composantes RGB, on peut bien approximer la véritable couleur du point $P$. Evidemment, puisque qu'il n'y avait pas de pixel exactement à cet endroit, il se pourrait que cette approximation ne représente pas

tout à fait la réalité, mais c'est tout de même beaucoup plus précis que de prendre le pixel le plus près.

### *4.4 Le coût de mise en correspondance*

A la section 4.2, nous avons évoqué le fait qu'il fallait "tester" des profondeurs le long de la droite épipolaire et sélectionner la plus probable. En recherche directe, la profondeur la plus probable est celle dont la correspondance, entre le pixel de l'image de référence et le pixel de l'autre image, est la meilleure. Nous verrons plus loin que, dans le cas des algorithmes utilisant une contrainte de lissage, ce n'est pas toujours le cas. Afin de simplifier le problème, on suppose quelques hypothèses sur la scène : la calibration est parfaite et les objets sont opaques et mats (Lambertiens (réf. [10])). Donc, si ces hypothèses sont respectées, le pixel dans l'autre image correspondant à la profondeur recherchée a exactement la même couleur que le pixel de l'image de référence. En pratique, il est rare de trouver une correspondance "parfaite" entre les deux pixels. D'abord parce que les hypothèses que nous avons faites ne sont pas toujours respectées. Cela peut aussi être dû à l'interpolation bilinéaire, qui n'est pas toujours le reflet exact de la réalité. Mais l'une des principales raisons est le bruit. Le bruit modifie la couleur des pixels et affecte donc la mise en correspondance. En recherche directe, on suppose que le pixel recherché dans l'autre image est celui dont la différence de couleurs avec le pixel de référence est la moins élevée. Cette différence de couleurs est appelée "coût de mise en correspondance". Pour calculer ce "coût", il existe plusieurs formules possibles. Celle que nous avons utilisée (qui est possiblement la plus utilisée d'entre toutes) est la "somme des différences carrées" (réf. [30], [22]). Avec des images couleurs, cela donne :

$$\sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \tag{4.6}$$

où $r_1, g_1, b_1$ et $r_2, g_2, b_2$ sont les couleurs $RGB$ des pixels des images #1 et #2 respectivement.

Jusqu'à maintenant, toutes les explications et tous les exemples ont été donnés en fonction de deux images : une image de référence et une autre. Ce choix a été délibéré afin de simplifier la présentation. En réalité, quand il est possible de le faire, il est préférable d'utiliser le plus d'images possibles. Le traitement n'est pas vraiment plus compliqué mais les résultats sont améliorés. Il faut toujours choisir une image de référence sur laquelle on va se baser pour calculer la carte de profondeur. Cependant au lieu d'avoir une seule autre image pour faire la mise en correspondance, on peut en avoir autant qu'on veut. Il suffit alors de calculer le coût de mise en correspondance pour chacune des images (autres que l'image de référence) et d'en faire la moyenne. Cependant, il y a certaines contraintes qui limitent le nombre d'images à traiter. D'abord, plus le nombre d'images à traiter sera élevé, plus le temps de calcul sera élevé. Ensuite, il faut que toutes les images contiennent au moins quelques éléments de la scène qui sont projetés dans l'image de référence. Une image qui ne contiendrait aucun élément à mettre en correspondance avec l'image de référence serait parfaitement inutile. Evidemment, il faut aussi que les mêmes éléments soient de couleurs semblables dans les images. Ceci suppose donc que les objets de la scène ont une surface Lambertienne (réf. [10]) et que les caméras ont été ajustées pour donner les mêmes teintes. Dans le cas d'un vidéo, comme c'est le cas pour nos essais, il faut le considérer comme étant un ensemble d'images qui auraient été prises par différentes caméras. Il suffit de prendre un programme qui décompose le vidéo en images et traiter chaque image séparément. Par exemple, 10 secondes d'un vidéo pris à 30 images/seconde donnera 300 images. Le vidéo est donc une solution pratique et abordable pour obtenir un grand nombre d'images que l'on peut ensuite traiter. Un seul hic cependant, puisque les images n'ont pas été prises au même instant, il faut s'assurer que rien n'a bougé dans la scène durant toute la durée du vidéo. Lors de nos essais, nous avons traité jusqu'à 25 images de chaque côté de l'image de référence (25 images avant + 25 images après + l'image de référence = 51 images par carte de profondeur). Il est vrai que cela ralentit le traitement mais puisque la priorité était la

qualité des résultats, cela en valait la peine. De toutes façons, comme vous le verrez plus loin, c'est plutôt la résolution du flot maximum qui ralentit le plus le traitement.

Tous les calculs de mise en correspondance qui ont été décrits précédemment mettaient en relation un pixel d'une image avec un pixel de l'image de référence. En pratique cependant, il est préférable de mettre en relation des petits carrés de pixels. Typiquement, on utilise des carrés de 3x3 pixels (9 pixels au total) (réf. [18], [12], [34]). Le fait de tester plusieurs pixels permet de mieux résister aux aberrations (souvent dues au bruit) qui apparaissent parfois quand on travaille avec un seul pixel. Cela suppose évidemment que le déplacement entre les caméras est faible. Pour mettre en correspondance un carré de pixels, il suffit de considérer le pixel à traiter comme étant le pixel central. Une fois que l'on a dé-projeté et reprojeté ce pixel central dans l'autre image, il faut considérer le point obtenu comme étant le point central du carré dans l'autre image. Puis, il suffit de calculer le coût de mise en correspondance en comparant chaque pixel du carré de l'image de référence avec le point correspondant dans l'autre image (on suppose que la rotation est négligeable). Par exemple, le coin en haut à gauche du carré de l'image de référence doit être comparé avec le coin en haut à gauche du carré de l'autre image. Le coût de mise en correspondance pour ce pixel central (de l'image de référence), à cette profondeur, est la moyenne de tous ces coûts. Par exemple, pour un carré de 3x3 pixels, le coût final sera la moyenne de 9 coûts, à moins que le pixel central se trouve en bordure de l'image (c'est pour cela que l'on fait la moyenne et non la somme). Précisons que le point central obtenu dans l'autre image par reprojection n'est pas nécessairement au centre d'un pixel. Il faut donc faire l'interpolation bilinéaire, non seulement de ce point central, mais aussi de tous les points du carré, chaque point étant considéré à une distance de une coordonnée pixel des points qui lui sont adjacents. Par exemple, si le point central se trouve à la coordonnée (100.75, 50.10), la coordonnée du point à sa gauche sera (99.75, 50.10). Il est important de constater que la dé-projection - reprojection ne se fait que sur le pixel central seulement, pas sur les autres pixels du carré. Il faut aussi

noter qu'un carré de 3x3 pixels fonctionne très bien en pratique. Si on prend un plus grand carré, par exemple 5x5 pixels, non seulement le traitement devient vraiment plus long, mais surtout, il survient des problèmes à cause des demi-occlusions ( "half-occlusions" (réf. [7])) et aussi des erreurs dues à notre approximation de la projection du carré par un seul point. Avec des carrés de 3x3 pixels, ces problèmes sont limités, voire même négligeables en pratique. Nous n'avons pas fait d'études détaillées sur le sujet mais des carrés de 3x3 pixels nous ont donné les meilleurs résultats et sont souvent utilisés par les chercheurs (réf. [18], [12], [34]).

### *4.5 Le volume de mise en correspondance*

Théoriquement, pour obtenir une carte de profondeur parfaite, il faudrait tester toutes les profondeurs possibles, c'est-à-dire tous les points qui se trouvent sur la droite épipolaire. Evidemment, ceci est impossible à faire en pratique. Pour approximer la profondeur réelle, on doit se définir un espace, qui devra être assez grand pour contenir tous les éléments de la scène, et découper cet espace en autant de tranches que le nombre de profondeurs que l'on a décidé de tester. Cet espace est appelé : "volume de mise en correspondance" (réf [24]). Cet espace a un "avant" qui doit se trouver devant le plus près des éléments de la scène et un "arrière" qui doit se trouver derrière le plus éloigné des éléments. Dans notre cas, cet "avant" et cet "arrière" sont déterminés à la calibration par le programme MatchMover. Ils représentent le point saillant le plus près et le plus loin que le programme a rencontrés en faisant le suivi de points ("feature tracking"). On laisse généralement une marge de manoeuvre, car MatchMover n'a pas nécessairement suivi le point le plus rapproché et le plus éloigné dans la scène. Le nombre de profondeurs que l'on va tester (le nombre de subdivisions du volume de mise en correspondance) est déterminé par l'utilisateur. Plus on teste de profondeurs, plus le résultat de la carte de profondeur sera précis, mais plus le traitement sera long. Il existe plusieurs façons de subdiviser ce volume de mise en

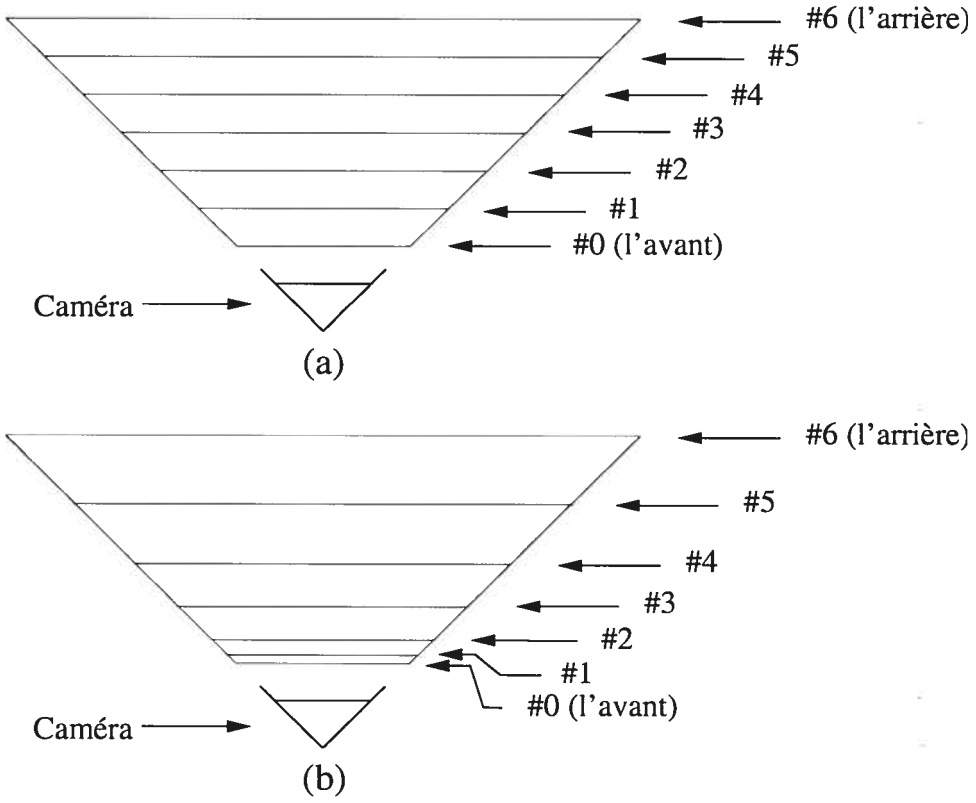



FIG. **4.3. Volume de mise en correspondance à 7 profondeurs. Chaque ligne correspond à une subdivision (qui est numérotée). (a) Subdivisions uniformes. (b) Subdivisions non-uniformes.**

correspondance. Il peut être subdiviser de façon uniforme, c'est-à-dire que la distance entre les subdivisions sera toujours la même (figure 4.3(a)). Cependant, les changements de profondeur n'ont pas un impact linéaire sur les déplacements observés dans l'image. Passer de 1 à 2 mètres de profondeur signifiera un bien plus grand déplacement en nombre de pixels dans l'image que de passer de 10 à 11 mètres. Le problème avec la subdivision uniforme est que l'on risque de tester le même point ou presque dans l'image (voir interpolation bilinéaire (section 4.3)) pour les subdivisions à "l'arrière", alors que, et c'est ça le plus grave, des pixels entiers se trouvant sur la droite épipolaire projetée n'auront pas été testés (même pas inclus dans aucune interpolation) à cause du manque de subdivisions à "l'avant" du volume de mise en

correspondance. Pour parer à ce problème, nous avons décidé de subdiviser le volume de mise en correspondance de façon non-uniforme (figure 4.3(b)). La formule utilisée pour déterminer la profondeur à tester est la suivante :

$$z_{cam} = z_{avant} + (z_{arriere} - z_{avant}) * \left(\frac{sub}{nb\_sub - 1}\right)^2 \tag{4.7}$$

où $z_{cam}$ est la profondeur recherchée, $z_{avant}$ et $z_{arriere}$ sont respectivement les profondeurs correspondant à "l'avant" et "l'arrière" du volume de mise en correspondance, *nb_sub* est le nombre de subdivisions totales du volume de mise en correspondance et *sub* est le numéro de la subdivision en partant de "l'avant" ($z_{avant}$ correspond à la subdivision #0). Notez que si on ne mettait pas le dernier terme de l'équation au carré $\left(\frac{sub}{nb\_sub-1}\right)$, on obtiendrait alors la subdivision uniforme.

### *4.6 La recherche directe*

L'algorithme de stéréo dense le plus simple (et le plus rapide) est la recherche directe. Il s'agit de trouver, pour un pixel, le coût de mise en correspondance pour chaque subdivision du volume de mise en correspondance. Ensuite, il suffit de conserver la profondeur qui correspond au coût le plus bas. Pour obtenir la carte de profondeur complète, il faut répéter ce processus pour tous les pixels de l'image de référence. Il n'y a aucun autre traitement à faire. C'est donc un algorithme très rapide mais qui possède son lot de problèmes. Il n'offre aucune résistance au bruit. Lorsqu'il y a une ambiguïté sur le choix d'une profondeur (un même coût pour deux profondeurs ou plus), l'algorithme va, aveuglément, toujours prendre la première des profondeurs équivalentes (ou la dernière selon la façon qu'il a été programmé), sans tenir compte de la profondeur des pixels voisins. Aucun algorithme de stéréo dense n'assure l'intégrité des éléments de la scène, mais la recherche directe est le pire dans ce domaine. On peut se retrouver avec des grandes différences de profondeur à chaque pixel alors qu'ils sont la projection d'un même objet dans la scène. C'est pourquoi des algorithmes plus élaborés ont été développés.

FIG. **4.4. La recherche directe appliquée sur la séquence "Tsukuba".**

## *4.7 La programmation dynamique*

La programmation dynamique est un algorithme très connu issu de la théorie des graphes (réf. [5]). Il résout globalement et efficacement (dans un temps polynômial) plusieurs problèmes de graphes. Par exemple, il est souvent utilisé pour trouver le chemin le plus court entre deux points. Pour pouvoir l'utiliser en stéréoscopie par ordinateur, il faut transformer la recherche de la carte de profondeur en un problème de chemin le plus court (réf. [9], [24]). Tout d'abord, pour contrer les problèmes de la recherche directe, il faut introduire une nouvelle contrainte : le lissage. Comme il a été mentionné au chapitre 2, le lissage exprime l'hypothèse que la profondeur d'un pixel est similaire à celle de ses voisins. L'algorithme de programmation dynamique appliqué à la stéréoscopie travaille sur une ligne de l'image à la fois. Tous les pixels de cette ligne sont, en quelque sorte, liés entre eux par le lissage. Ils s'influenceront mutuellement afin d'arriver à un "compromis" entre maintenir les coûts de mise en correspondance les plus bas possibles et garder une certaine constance dans la profondeur trouvée entre les pixels. Ce "compromis" est en fait le chemin le plus court. Prenons un exemple (figure 4.5). Le schéma représente une ligne d'une image de quatre pixels de large. On teste trois profondeurs possibles pour chaque pixel.

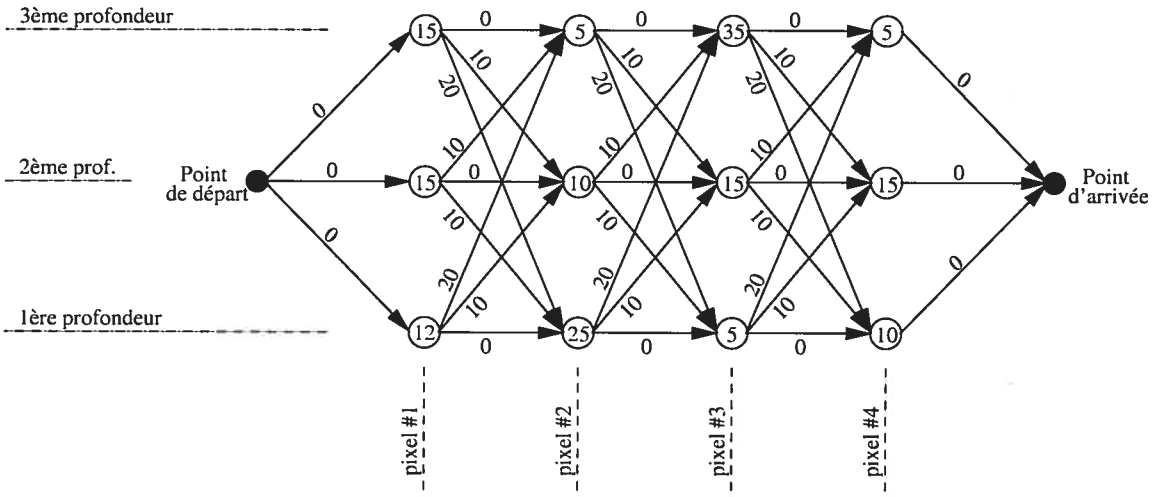



FIG. **4.5. Exemple de programmation dynamique. Exemple d'une ligne de 4 pixels de large avec 3 profondeurs possibles.**

L'algorithme procède d'abord exactement comme la recherche directe afin d'obtenir, pour le premier pixel, les coûts des mises en correspondance pour les trois profondeurs. Mais, au lieu de choisir tout de suite la profondeur qui correspond au coût le plus bas, on va conserver tous ces coûts dans un tableau. Puis on passe au pixel suivant, et ainsi de suite jusqu'au dernier pixel de la ligne. Nous aurons alors, dans un tableau à deux dimensions, tous les coûts de mises en correspondance pour toutes les profondeurs et ce, pour tous les pixels de la ligne (les coûts sont représentés dans les cercles sur le schéma). Maintenant, pour obtenir un graphe qui pourra être résolu par l'algorithme de programmation dynamique, il faut relier ces points par des arcs. Le coût d'un arc est lié à la différence de profondeurs entre les noeuds, multipliée par une constante de lissage. Par exemple, passer sur l'arc qui relie la 1ère profondeur du pixel #1 à la 3ème profondeur du pixel #2 coûtera deux fois la constante de lissage. Dans cet exemple, la constante de lissage est de 10. Une fois que tous les arcs ont été créés, il ne reste plus qu'à relier les noeuds du pixel #1 au point de départ avec des arcs d'un coût de 0 et relier les noeuds du dernier pixel au point d'arrivée des arcs d'un coût de

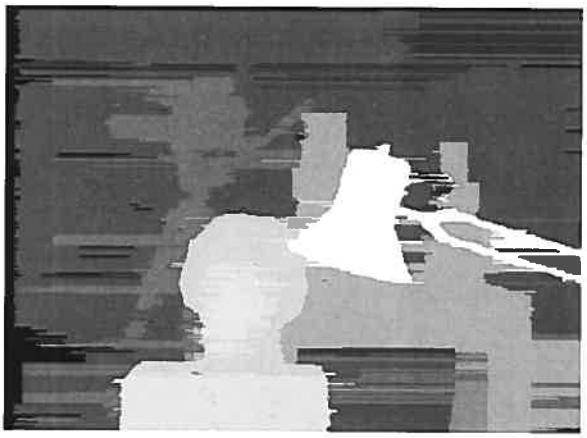


FIG. **4.6. L'algorithme de programmation dynamique appliqué sur la séquence "Tsukuba"**

0. Tout est maintenant en place pour résoudre le chemin le plus court entre le point de départ et le point d'arrivée par programmation dynamique. Cet algorithme ne sera pas expliqué dans ce mémoire. Pour avoir tous les détails, vous pouvez consulter (réf. [5]) ou tout autre bon livre sur la théorie des graphes. Il faut seulement faire attention, quand on calcule le coût d'un chemin, d'additionner les coûts des arcs et des noeuds utilisés. Pour l'exemple, les profondeurs retenues pour les pixels #1 à #4 sont : 2-2-1-1 (pour un coût total de $15 + 10 + (10) + 5 + 10 = 50$). Notez qu'un lissage extrême aurait donné comme résultat 1-1-1-1 (pour un coût total de $12 + 25 + 5 + 10 = 52$). La recherche directe (qui correspond à un lissage de 0) donne plutôt les profondeurs suivantes : 1-3-1-3 (pour un coût total de $12 + 5 + 5 + 5 = 27$).

L'algorithme de programmation dynamique donne de bien meilleurs résultats que la recherche directe. Cependant, cet algorithme ne permet de "lisser" que sur une seule dimension à la fois (une ligne (comme dans l'exemple précédent) ou une colonne). Idéalement, il faudrait "lisser" dans les deux dimensions. On pourrait penser à une méthode dans laquelle on traite d'abord l'image horizontalement (lignes), ensuite verticalement (colonnes) pour finalement jumeler les deux résultats ensemble afin d'obtenir la carte de profondeur finale. Le problème avec cette méthode est

que le lissage n'est pas fait dans les deux dimensions simultanément. En traitant les lignes et les colonnes séparément, on ne permet pas aux pixels voisins dans les deux dimensions d'opposer leur influence directement. Malheureusement, l'algorithme de programmation dynamique ne permet pas de traiter l'image dans son entier, avec ses deux dimensions, simultanément. Pour faire cela, nous avons besoin d'un algorithme encore un peu plus évolué.

### *4.8 Le flot maximum*

Le calcul du flot maximum est un problème très connu en théorie des graphes (réf. [5], [6]). Ce problème peut aussi être résolu globalement et efficacement. L'analogie qui est souvent employée pour visualiser cet algorithme est celle d'un réseau d'aqueduc. Imaginons un réseau de "tuyaux" avec un drain unique et une source unique d'où se propage un très grand débit (d'eau). L'algorithme de flot maximum permet de trouver les endroits dans ce réseau où il y a saturation de la capacité des "tuyaux". Ces endroits constituent la solution au problème de flot maximum et est aussi appelée "la coupe minimum". Dans notre cas, cette coupe minimum représentera la carte de profondeur recherchée. Mais d'abord, il faut transformer notre problème de stéréoscopie en un problème de flot maximum (réf. [22]).

Pour tester les différentes profondeurs possibles d'un pixel, tout en appliquant un lissage aussi bien vertical qu'horizontal, il faut que ce pixel soit un noeud 6-connexe : 2 connexions pour le lissage vertical, 2 connexions pour le lissage horizontal, et finalement, 2 connexions pour la profondeur, la profondeur juste avant et celle juste après. Les arcs correspondant aux connexions des pixels voisins de même profondeur sont appelés "arcs de lissage" alors que les arcs correspondant au changement de profondeur sont appelés "arcs de disparité". Si chaque pixel pour chaque profondeur à tester représente un noeud 6-connexe, l'image au complet sera alors représentée par

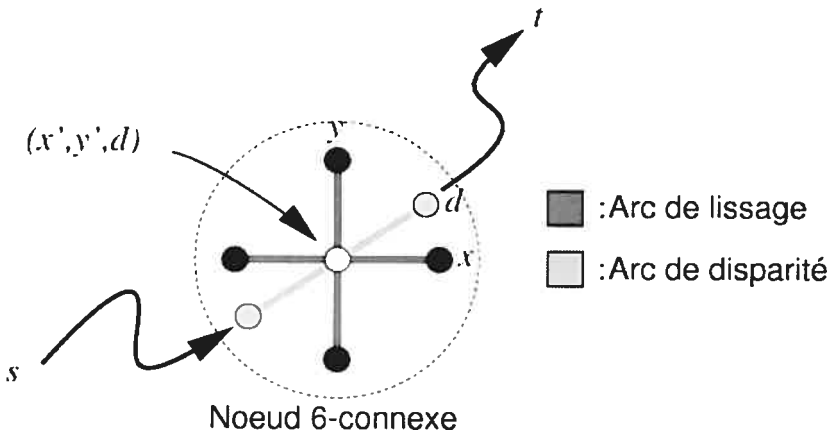



FIG. **4.7. Un noeud 6-connexe utilisé dans l'algorithme de flot maximum.**

un "cube" 3D aux dimensions suivantes :

$$cube = X_{size} * Y_{size} * (nb\_prof + 1) \tag{4.8}$$

où *cube* est le nombre de noeuds que contient le "cube", $X_{size}$ est la largeur de l'image en pixels, $Y_{size}$ est la hauteur de l'image en pixels et $nb\_prof$ est le nombre de profondeurs à tester.

Remarque : les trois dimensions du "cube" n'ont pas besoin d'être égales. Le "cube" doit être en fait un prisme rectangulaire. L'utilisation du terme "cube" est un abus de langage et son utilisation a pour but de faciliter la lecture.

Ce "cube" constitue le problème à résoudre. Une "coupe" séparant le cube en deux, de sorte que l'on retrouve un "avant" et un "arrière", constitue une solution au problème. Ce que l'on recherche, c'est la coupe minimum. Etant donné qu'il n'y a pas d'ordre strict pour bâtir la solution (on ne résout pas le problème dans l'ordre, du premier pixel au dernier pixel de l'image), on ne peut pas solutionner ce problème par l'algorithme de programmation dynamique. C'est pourquoi on utilise un algorithme de résolution du flot maximum.

Il faut d'abord initialiser ce "cube" avec les différentes valeurs obtenues par la

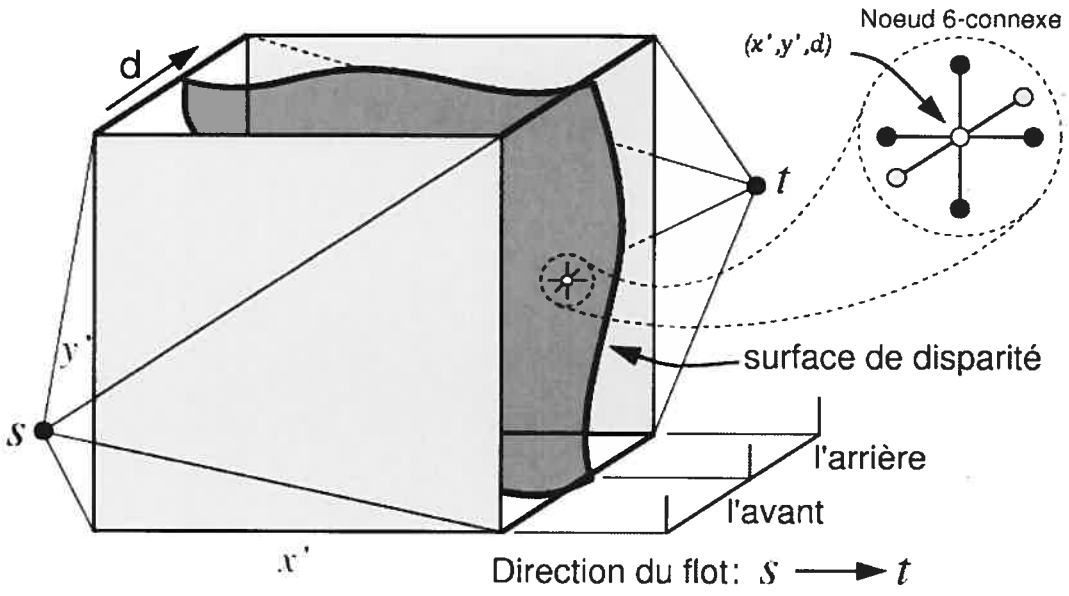



FIG. **4.8. Problème de flot maximum (le "cube").**

stéréoscopie. Chaque arc reliant deux noeuds représente un "tuyau" (selon l'analogie) d'une certaine capacité. Ces différentes capacités sont déterminées par les coûts de mise en correspondance et le degré de lissage.

$$c(u,v) = \begin{cases} 0 & \text{si } u \text{ ou } v \text{ n'appartient pas au "cube"} \\ \infty & \text{si } u \text{ ou } v \text{ est la source ou le drain} \\ c_{liss} & \text{si } (u-v) = (\Delta_x, \Delta_y, 0) \\ c_{disp} & \text{si } (u-v) = (0,0,\Delta_d) \end{cases}$$

où $u$ et $v$ représentent des noeuds, $c(u,v)$ représente la capacité entre les noeuds $u$ et $v$, $c_{disp}$ représente la capacité de l'arc de disparité reliant $u$ et $v$, et $c_{liss}$ représente la capacité de l'arc de lissage entre $u$ et $v$.

$$c_{liss} = \text{degré de lissage} \tag{4.9}$$

$$c_{disp} = \text{coût de mise en correspondance du pixel } (x,y) \text{ à la profondeur } min(u,v) \tag{4.10}$$

(4.11)

Remarquez que le coût de mise en correspondance de l'arc reliant $u$ et $v$ a été calculé à la plus petite profondeur entre $u$ et $v$. Afin de tester toutes les profondeurs, y compris la dernière, il faut que le "cube" ait une profondeur de un de plus que le nombre de profondeurs à tester (comme c'est le cas selon l'équation 4.8).

Puisque le coût de mise en correspondance est associé à la capacité d'un arc, plus la correspondance sera bonne, plus la capacité de l'arc sera petite, et donc, plus il y a de chances que cet arc soit saturé de flot et fasse partie de la coupe minimum. Evidemment, cela dépend aussi de l'influence simultanée apportée par les pixels adjacents (horizontaux et verticaux), par le biais des arcs de lissage. Toute l'image se résout donc d'un coup en retrouvant la coupe minimum. Etant donné que la coupe minimum est une solution globale, l'influence d'un noeud peut se propager aussi loin que nécessaire.

Une fois que le "cube" est complètement initialisé, il ne reste plus qu'à laisser "rouler" un algorithme de flot maximum sur ce problème. L'algorithme de flot maximum que nous avons implanté est le "preflow-push lift-to-front (réf. [6])". La coupe minimum obtenue représente la carte de profondeur recherchée.

Par le fait qu'il propage le lissage simultanément aussi bien à l'horizontal qu'à la vertical, l'algorithme de flot maximum colle beaucoup mieux à la réalité que l'algorithme de programmation dynamique. Ceci se reflète dans les résultats qui sont d'une bien plus grande qualité. Il n'y a plus ces grandes lignes de même profondeur dans la carte de profondeur qui sont typiques de la programmation dynamique et qui sont dues au lissage sur une seule dimension de l'image (figure 4.6). En fait, l'algorithme de flot maximum donne les meilleurs résultats connus pour un algorithme de stéréo dense. Malgré le fait qu'il soit polynômial, cet algorithme prend beaucoup plus de temps à s'exécuter que l'algorithme de programmation dynamique, qui était lui-même beaucoup plus lent que la recherche directe. De plus, la résolution du flot

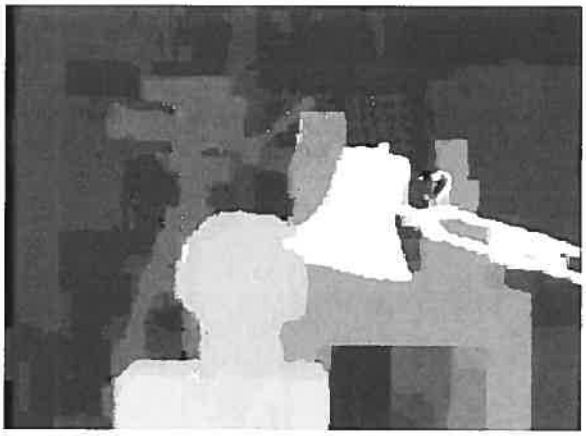


FIG. **4.9. Le flot maximum sur la séquence "Tsukuba"**

maximum requiert énormément de mémoire. On doit conserver un nombre de noeuds équivalents à la taille du "cube" (équation 4.8). Puisque le "cube" est un graphe régulier, il est possible d'avoir une représentation très compacte du "cube" avec des noeuds ne prenant que 36 octets. Mais même avec une représentation aussi compacte, si la carte de profondeur que l'on recherche est le moindrement grande, l'ordinateur doit posséder au moins 256MB de mémoire vive, préférablement 512MB. Pour nos séquences, la quantité de mémoire nécessaire variait entre 100MB et 500MB. Nous avons tout de même choisi l'algorithme de flot maximum pour notre méthode car il nous a permis d'obtenir les meilleures cartes de profondeur et que c'est la qualité du résultat final qui nous importait le plus (et aussi parce que nous possédons au laboratoire des ordinateurs assez puissants pour le faire "rouler" !).

Chapitre 5

# LA TECHNIQUE "MAX-GAUCHE-DROITE"

En stéréoscopie, comme il a été mentionné au chapitre 4, il faut mettre en correspondance des pixels (vrais ou interpolés) entre les images. Pour que la stéréoscopie fonctionne, il faut que les éléments qui apparaissent dans l'image de référence apparaissent aussi sur au moins une des autres images, lesquelles doivent avoir été prises d'un angle de vue différent de l'image de référence. C'est pour cette raison que, lorsque nous filmons une scène, nous déplaçons la caméra vidéo en essayant d'éviter les mouvements trop brusques. Ainsi nous obtenons des images qui se ressemblent mais qui sont toutes un peu différentes à cause du point de vue différent à partir duquel elles ont été prises. Dans les scènes typiques, les objets près de la caméra se déplacent (dans les images) plus vite que les objets éloignés. Evidemment, les objets qui sont près de la caméra cachent les objets qui sont derrière. En se déplaçant dans les images, les objets qui sont près de la caméra vont se trouver à cacher d'autres objets plus éloignés qui eux se déplacent plus lentement d'une image à l'autre. Le problème, c'est que certains éléments de la scène qui apparaissent dans l'image de référence n'apparaissent plus sur les autres images parce qu'un objet rapproché, à cause du différent point de vue, s'est glissé devant. Comment alors trouver la bonne profondeur associée à ces pixels de l'image de référence si nous n'avons aucune correspondance dans les autres images ? C'est ce qu'on appelle le phénomène des demi-occlusions ("half-occlusions" (réf. [7])). Les demi-occlusions sont parmi les plus importants problèmes en stéréoscopie par ordinateur. La technique "max-gauche-droite", que nous avons développée, s'attaque directement au problème des demi-occlusions et permet d'obtenir des résultats prometteurs.

D'abord pour que cette technique fonctionne, il faut qu'une contrainte, introduite

au chapitre 2, soit absolument respectée :
*"La séquence vidéo devra être filmée par une caméra vidéo digitale qui se déplacera assez lentement dans la direction à peu près perpendiculaire à l'axe de visée. Ce déplacement devra se faire sans retour en arrière."*

Si la caméra vidéo n'a pas fait de retour en arrière, cela veut dire que le groupe d'images est clairement divisé dans l'espace comme dans le temps. Prenons par exemple une caméra vidéo qui se déplace de droite à gauche (par rapport à la direction de l'axe de visée). Supposons que le film contient 300 images (#0 à #299) et que nous prenons l'image #150 comme image de référence. Toutes les images qui ont été prises "avant" dans le temps (#0 à #149) sont des images "à droite" de l'image de référence, c'est-à-dire qu'elles ont été prises d'un point de vue "à droite" par rapport au point de vue de l'image de référence. De même, toutes les images qui ont été prises "après" dans le temps (#151 à #299) sont des images "à gauche" de l'image de référence. Séparer les images en deux groupes, celles à gauche et celles à droite, est la clé de la technique "max-gauche-droite".

Dans une image, les demi-occlusions sont situées sur les côtés des objets, là où il y a une discontinuité dans la profondeur. Sur la carte de profondeur, une zone floue apparaît à ces endroits et forme la "bande d'occlusion" (figure 5.1(a)). Plus un objet est près de la caméra vidéo, plus la bande d'occlusion sera large. De même, plus la caméra vidéo se déplace rapidement, plus la bande d'occlusion sera large. La bande d'occlusion vient du fait qu'à cet endroit dans l'image il y a une grande ambiguïté sur la profondeur trouvée. La cause de cette ambiguïté est que les pixels à cet endroit dans l'image de référence n'ont pas de correspondance dans plusieurs des autres images. Il est très possible qu'il y ait une bonne correspondance des pixels dans plusieurs images, mais les images dans lesquelles il n'y a pas de correspondance viennent gâcher le résultat. Pour obtenir le bon résultat, il faudrait ne tenir compte que des images dans lesquelles il y a une bonne correspondance. C'est l'idée qui est derrière la technique "max-gauche-droite".

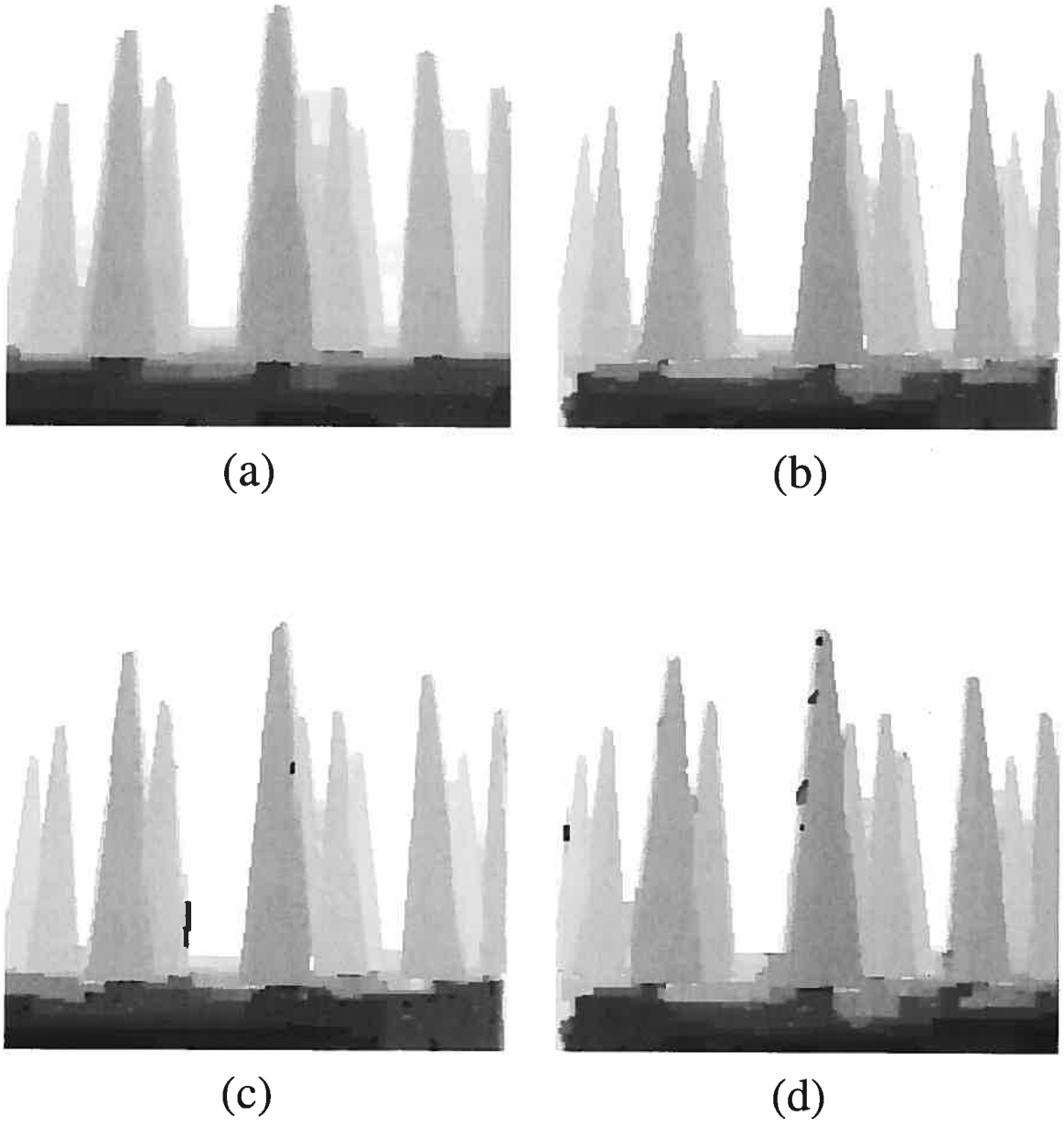



FIG. **5.1. La technique max-gauche-droite sur une séquence virtuelle. Les différentes cartes de profondeur ont été obtenues par : (a) la stéréoscopie "classique" (moyenne de toutes les images), (b) la technique max-gauche-droite, (c) les images à gauche seulement, (d) les images à droite seulement. Nombre d'images utilisées : 11 (5 à droite, 5 à gauche et l'image de référence).**

Prenons un exemple à trois caméras (figure 5.2). On essaie de reconstituer la carte de profondeur de la scène à partir de la caméra de référence. On considère pour l'instant que c'est la boule B1 (et non la boule B2) qui fait partie de la scène. On retrouve assez facilement la profondeur de la section A-B du mur arrière puisqu'elle est visible par toutes les caméras. Cependant, les choses se compliquent lorsque l'on essaie de retrouver la profondeur de la section B-C. Cette section, qui apparaît dans l'image de référence, est bien visible de la caméra de gauche mais pas de la caméra de droite. La boule B1 au centre de la scène cache cette partie du mur du point de vue de la caméra de droite. Mais l'algorithme stéréo de mise en correspondance ne le sait pas et essaie de trouver tout de même une correspondance avec les points qui sont visibles dans la caméra de droite. En testant la vraie profondeur de cette section B-C dans l'image de droite, on tombe sur les pixels de la boule B1 qui est vraisemblablement d'une couleur différente du mur. Cela donne donc un coût de mise en correspondance très élevé. Même si le coût de mise en correspondance avec la caméra de gauche était tout près de 0, puisqu'en stéréoscopie "classique" on fait la moyenne des coûts de mise en correspondance de toutes les images, le coût final sera assez élevé. Donc cette profondeur, qui est la vraie, ne sera pas retenue. La profondeur qui obtiendra probablement un meilleur coût de mise en correspondance sera plutôt celle qui correspondra à un point du mur qui se retrouve aussi dans l'image de droite. Deux sections du mur sont visibles de la caméra de droite, la section A-B et la section F-I. Mais il est impossible géométriquement qu'une droite, reliant le point focal de la caméra de droite à la section F-I, croise la droite épipolaire de la caméra de référence. Si le mur était à une distance infinie, le pixel à tester dans l'image de droite serait celui correspondant à la projection du point P2. Plus on rapproche le mur, plus on déplace le pixel à tester vers la gauche dans l'image de droite. Quand les axes de visée sont parallèles, il est impossible géométriquement qu'un point soit plus à droite dans une image qui a été prise à droite de l'image de référence. La seule possibilité pour que l'on retrouve le mur dans l'image de droite, à la droite de la boule (qui

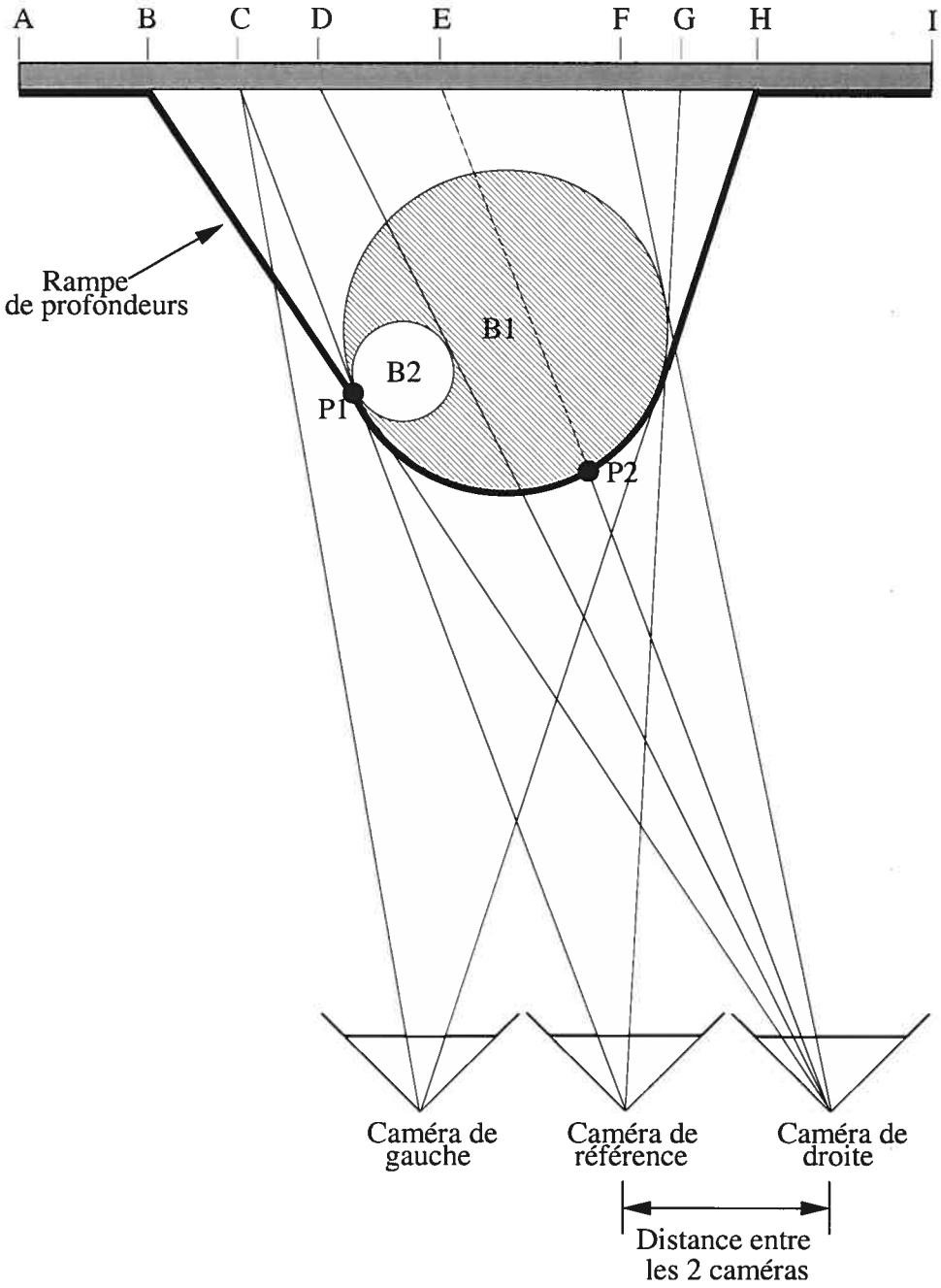



FIG. **5.2. La géométrie des demi-occlusions en stéréoscopie. Les cas de demi-occlusion sont situés à la section B-C (caméra de droite) et la section G-H (caméra de gauche).**

crée la demi-occlusion), c'est que cette boule soit moins large que la distance qui sépare la caméra de droite de la caméra de référence. Si c'était la boule B2 qui faisait partie de la scène (au lieu de la boule B1), on tomberait, en testant les différentes profondeurs, sur la section D-E du mur qui se trouve à la droite de l'objet qui crée la demi-occlusion, du point de vue de la caméra de droite. Donc, pour pouvoir se limiter à la section A-B, il faut que l'objet qui crée la demi-occlusion soit plus large que la distance qui sépare les centre optiques des caméras. En pratique, puisque nous avons dit précédemment que la caméra vidéo doit se déplacer assez lentement, la distance entre les captures d'images est suffisamment petite pour que l'on considère que tous les objets de la scène sont suffisamment larges. Donc, on peut toujours considérer que l'algorithme de stéréoscopie va trouver des points à la gauche de l'objet qui crée la demi-occlusion (section A-B dans l'exemple). Ceci est extrêmement intéressant car cela veut dire que si l'algorithme de mise en correspondance se trompe à cause d'une demi-occlusion, il a toujours choisi une profondeur moins grande que la véritable profondeur de l'objet. Si on introduit un certain degré de lissage à l'algorithme de stéréoscopie, le résultat est à peu près le même. Les pixels qui font partie de la demi-occlusion (section B-C) n'auront probablement pas de bons coûts de mise en correspondance, peu importe la profondeur testée. Ces pixels seront donc considérés ambigus. A cause du lissage, ces pixels se laisseront influencer, proportionnellement à leur proximité, par les pixels voisins vraiment fiables, c'est-à-dire ceux qui se trouvent tout juste à l'extérieur de la bande d'occlusion (points B et P1). Mais même si, par une curieuse coïncidence, certains pixels à l'intérieur de la bande d'occlusion obtiennent un très bon coût de mise en correspondance pour une certaine profondeur (ce qui les rendrait fiables), ce sera toujours pour une profondeur moins grande que la profondeur réelle. Dans tous les cas, cela va créer un effet de "rampe de profondeurs". Cette rampe de profondeurs correspond au flou que l'on retrouve sur les côtés des objets dans les cartes de profondeur typiques (figure 5.1(a)).

L'idée de la technique "max-gauche-droite" nous est venue au moment où nous

avons fait la constatation que si l'algorithme de stéréoscopie se trompe à cause d'une demi-occlusion, il choisit toujours une profondeur moins grande que la profondeur réelle. Or, les demi-occlusions sur le côté gauche des objets ne sont créés que sur les images des caméras situées à la droite de la caméra de référence, et les demi-occlusions sur le côté droit des objets ne sont créés que sur les images des caméras situées à la gauche de la caméra de référence. Donc, il suffit de construire deux cartes de profondeur séparément, une pour les images de gauche et une pour les images de droite, et de prendre la profondeur maximum entre les deux. Voyons cette technique plus en détail :

- Etapes pour la technique "max-gauche-droite" :
  1. Fabriquer une carte de profondeur à partir des images des caméras situées à la gauche de la caméra de référence seulement.
     - Pas de demi-occclusion sur le côté gauche des objets (section B-C dans l'exemple).
     - Demi-occlusions sur le côté droit des objets (section G-H).
  2. Fabriquer une carte de profondeur à partir des images des caméras situées à la droite de la caméra de référence seulement.
     - Pas de demi-occclusion sur le côté droit des objets (section G-H dans l'exemple).
     - Demi-occlusions sur le côté gauche des objets (section B-C).
  3. Créer la carte de profondeur finale en prenant, pour chaque pixel, la profondeur maximum entre celle obtenue sur la carte de profondeur des images de gauche et celle obtenue sur la carte de profondeur des images de droite.

La technique "max-gauche-droite" est vraiment très simple. Le traitement est un peu plus long que de faire de la stéréoscopie "classique" (moyenne de toutes les images), mais elle est tellement efficace pour régler les problèmes de demi-occlusions que ça vaut la peine de patienter un peu. On peut constater la différence entre la carte

de profondeur obtenue par la technique "max-gauche-droite" (figure 5.1(b)) et celle obtenue, à partir de la même image, par la stéréoscopie "classique" (figure 5.1(a)). On peut voir que sur l'image obtenue par la technique max-gauche-droite, le flou sur les côtés des objets a presque complètement disparu. On constate la même chose en comparant les figures 5.3(a) et 5.3(b). Cependant, il arrive parfois, quand il n'y a pas assez d'images d'un côté ou de l'autre, que l'on obtienne un peu plus d'aberrations ("outliers") dans la carte de profondeur par rapport à celle obtenue par stéréoscopie "classique". Ceci pourrait être un problème, mais il a été neutralisé par l'autre technique que nous avons développée et qui fait le sujet du prochain chapitre : "la consistance de profondeur".

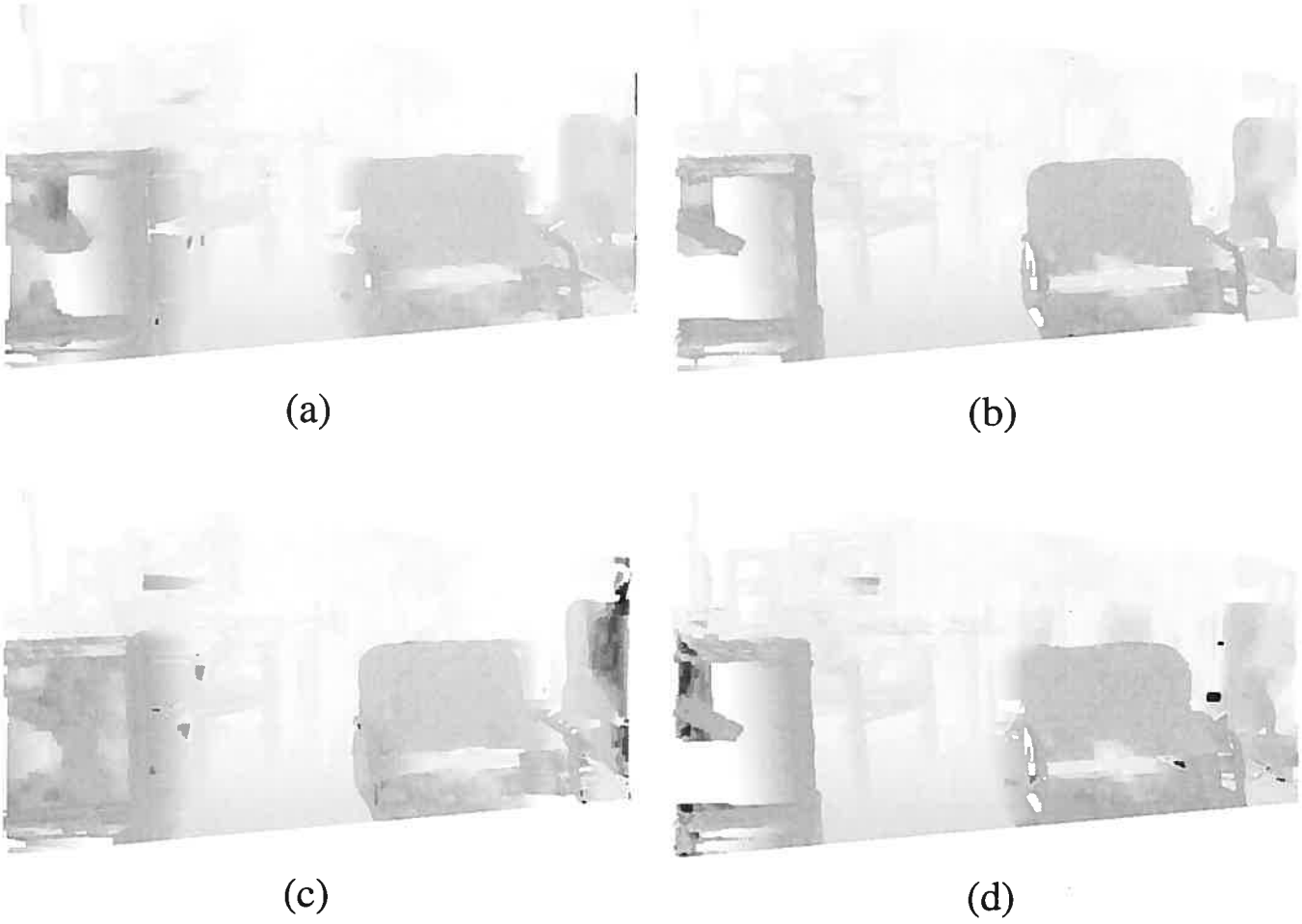



FIG. **5.3. La technique max-gauche-droite sur la séquence du laboratoire. Les différentes cartes de profondeur ont été obtenues par : (a) la stéréoscopie "classique" (moyenne de toutes les images), (b) la technique max-gauche-droite, (c) les images à gauche seulement, (d) les images à droite seulement. Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

Chapitre 6

# LA TECHNIQUE "CONSISTANCE DE PROFONDEUR"

Comme nous l'avons vu au chapitre 4, les algorithmes de stéréo dense qui sont les plus évolués et qui donnent les meilleurs résultats (programmation dynamique, flot maximum) imposent tous une contrainte de lissage. Ceci permet de réduire le nombre d'aberrations ("outliers") dans la carte de profondeur que l'on fabrique. Mais quel niveau de lissage doit-on employer ? Si le lissage est trop faible, il risque d'avoir des variations de profondeur à chaque pixel. C'est particulièrement le cas pour les grandes surfaces de couleur uniforme. Ce genre de surface ne présente pas de différences de couleur entre les pixels et donc, le coût de mise en correspondance sera le même pour plusieurs profondeurs testées. Il en résultera donc une certaine ambiguïté. Seuls les pixels sur les bords de la surface seront vraiment fiables puisqu'ils feront contraste avec le restant de la scène. Si le lissage est assez fort, les pixels fiables qui se trouvent sur les bords vont influencer tous les autres pixels de la surface. Mais si le lissage est trop faible, l'influence sera trop faible et l'algorithme va trouver plusieurs profondeurs erronées au centre de la surface. De plus, un lissage trop faible sera très sensible au bruit dans l'image. Le lissage doit donc être suffisamment fort pour obtenir une bonne carte de profondeur. D'un autre côté, un lissage trop fort n'est pas vraiment mieux. Si la scène comporte des petits détails qui ne prennent pas beaucoup de pixels dans l'image, ces détails risquent d'être "effacés" sur la carte de profondeur (même si les coûts de mise en correspondance sont assez bons), à cause de l'influence trop forte exercée par les pixels voisins. Un lissage extrême tend à uniformiser la carte de profondeur à la profondeur à laquelle adhèrent le plus grand nombre de pixels dans l'image. Il faut donc choisir un niveau de lissage moyen. Mais lequel exactement ? S'il existait un niveau de lissage idéal pour toutes les images possibles, le problème serait

réglé. En réalité, le lissage doit être adapté au contenu de l'image. S'il y a beaucoup de petits détails dans l'image, il faut choisir un lissage plutôt faible, alors que s'il y a surtout des grandes surfaces de couleur uniforme, il faut choisir un lissage plutôt fort. Pour compliquer les choses un peu plus, certaines images présentent à la fois des grandes surfaces de couleur uniforme et plusieurs petits détails, ce qui suppose que l'on devrait adapter localement le lissage dans l'image. Le choix du lissage est un des problèmes les plus importants en vision par ordinateur et la seule solution qui a été proposée jusqu'à maintenant est de tester plusieurs niveaux de lissage différents et de choisir celui qui donne la meilleure carte de profondeur. Ceci est évidemment extrêmement long et nécessite qu'un utilisateur juge de la qualité des cartes de profondeur. Nous nous sommes attaqués à ce problème et la solution que nous proposons permet de fabriquer une carte de profondeur d'une qualité s'approchant du lissage "idéal", tout en ne fabriquant que deux cartes de profondeur au total : une à un lissage minimal et l'autre à un lissage maximal. Cette technique, que nous avons appelée "consistance de profondeur", est rapide, ne nécessite aucune intervention humaine, et donne des résultats intéressants peu importe la scène.

Remarque : l'expresion *lissage "idéal"* ("idéal" entre guillemets) employé tout au long de ce mémoire désigne le niveau de lissage qui nous a permis d'obtenir les meilleurs résultats dans notre contexte expérimental. Cela ne signifie pas que ces résultats sont parfaits. Il signifie seulement que pour une séquence d'images donnée, en appliquant une seule valeur de lissage pour fabriquer toute la carte de profondeur, c'est ce niveau de lissage qui nous a permis d'obtenir le résultat qui s'approchait le plus du résultat parfait (en comparant les résultats de façon subjective (à l'oeil)). C'est ce que nous avons appelé *lissage "idéal"*.

Si on fabrique deux cartes de profondeur de la même scène, une avec un lissage très faible et l'autre avec un lissage très fort, on obtiendra deux résultats très différents. La carte de profondeur obtenue par le lissage très faible (figure 6.1(b)) présentera une profondeur pour tous les éléments de la scène, même les plus petits détails. Par

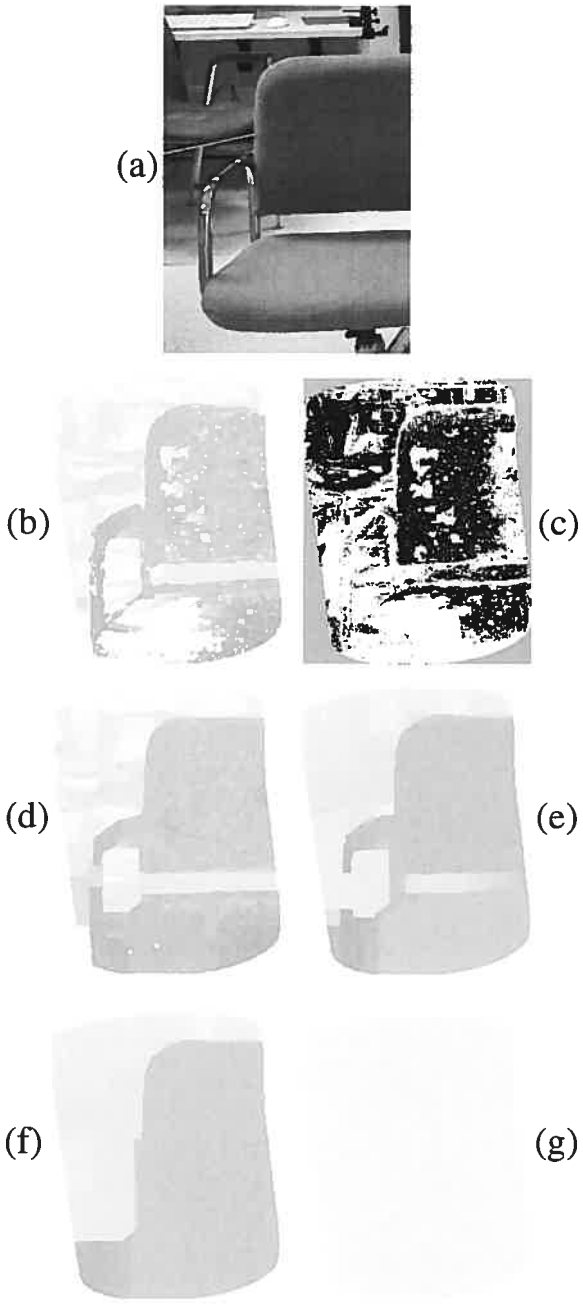



FIG. **6.1. La technique consistance de profondeur sur la séquence du laboratoire. (a) image du laboratoire, (b) carte de profondeur avec un lissage de 1, (c) carte de consistance associée à (b) (les pixels noirs sont valides), (d) carte de profondeur finale obtenue par la technique "consistance de profondeur" (lissage de 1 000 000 avec verrouillage de profondeur), (e) carte de profondeur obtenue d'un lissage "idéal" (1 200), (f) carte de profondeur obtenue d'un lissage trop fort (5 000), (g) carte de profondeur obtenue d'un lissage extrême (1 000 000) sans verrouillage de profondeur. Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

contre, cette carte de profondeur sera "bruitée", c'est-à-dire qu'il y aura beaucoup de différences de profondeur entre pixels adjacents, notamment là où la couleur est uniforme dans l'image. D'autre part, la carte de profondeur obtenue par le lissage très fort (figure 6.1(f)), présentera plutôt une sorte de division de l'image en quelques grands "morceaux", tous les pixels d'un même "morceau" étant définis à la même profondeur. Cette carte de profondeur ne contiendra aucun bruit. Par contre les petits détails qui se trouvent à des profondeurs différentes du reste de l'image auront été "effacés". Aucune de ces deux cartes de profondeur n'est intéressante en soi, leurs défauts étant trop prononcés. Pourtant, elles possèdent toutes les deux les qualités qui composent une carte de profondeur parfaite. Une carte de profondeur parfaite révèle la profondeur exacte de tous les éléments de la scène, c'est-à-dire qu'elle inclut tous les petits détails de l'image, mais sans être "bruitée". Toute l'information (ou presque) pour fabriquer une carte de profondeur parfaite est contenue dans nos deux cartes de profondeur. Il s'agit seulement de soutirer l'information intéressante de chacune d'entre elles afin d'obtenir le meilleur de ces deux cartes. Ceci n'est pas une tâche facile et c'est ce que nous avons tenté de faire avec notre technique "consistance de profondeur". Nous n'arrivons pas à obtenir des cartes de profondeur parfaites, mais elles se comparent très bien aux cartes de profondeur obtenues d'un lissage "idéal" (qui elles ne sont pas parfaites non plus!). Le lissage "idéal" doit être choisi par un utilisateur après avoir fait de multiples tests à différents lissages alors que la technique "consistance de profondeur" est complètement automatique. Cette technique nécessite de comparer des cartes de profondeur de plusieurs images adjacentes entre elles. Il est vrai que cela allonge considérablement le traitement si nous ne voulons qu'une seule carte de profondeur. Cependant, quand il est question d'ajouter un objet virtuel dans la scène (comme dans notre cas), il faut obtenir une carte de profondeur pour toutes les images dans lesquelles l'objet virtuel apparaît. Dans ces conditions, comparer des cartes de profondeur entre elles ne nécessitent pas d'efforts supplémentaires, puisque les cartes de profondeur doivent être fabriquées de toutes façons. La seule condition

à respecter est de ne pas travailler sur une seule carte de profondeur du début du processus jusqu'à la fin, mais plutôt travailler sur toutes les cartes de profondeur en parallèle.

- Etapes pour la technique "consistance de profondeur" :

    1. Fabriquer une carte de profondeur avec un lissage minimal pour chacune des images de la séquence.
    2. Pour chacune des cartes de profondeur obtenues à l'étape 1, valider la profondeur de chaque pixel en évaluant sa consistance sur les cartes de profondeur adjacentes.
    3. Fabriquer une carte de profondeur avec un lissage maximal pour chacune des images de la séquence, en "verrouillant" la profondeur des pixels qui auront été validés à l'étape 2.

Nous avons fixé le lissage minimal à 1. Théoriquement, ce lissage minimal devrait être égal à 0. En pratique cependant, une recherche directe (lissage de 0) crée des cartes de profondeur tellement "bruitées" que presque tous les pixels des cartes de profondeur (si ce n'est pas tous) sont considérés invalides à l'étape 2. Pour que la technique fonctionne bien, il faut qu'il y ait un minimum de pixels valides dans la carte de profondeur. Ce minimum varie avec le nombre de détails dans l'image. En ce qui concerne les séquences que nous avons essayées, un lissage de 1 nous a toujours permis d'obtenir ce minimum. Nous avons fixé le lissage maximal à 1 000 000. Théoriquement, le lissage maximal devrait être égal à l'infini. En pratique, une valeur de 1 000 000 convient parfaitement.

Remarque : Pour donner un ordre de grandeur en ce qui concerne le niveau de lissage, disons que le lissage "idéal", même s'il peut varier beaucoup d'une image à l'autre, se maintient souvent entre 25 et 1 500.

## 6.1 *Comment valider la profondeur d'un pixel qui fait partie d'une carte de profondeur ?*

Pour faire cela, nous nous sommes inspirés d'une technique assez connue mais qui est surtout utilisée, par les autres chercheurs, pour détecter les demi-occlusions : le "right-left consistency check" (réf. [7], [11]). La théorie qui est derrière peut se résumer ainsi : si on prend un pixel de la caméra de référence et qu'on le dé-projette vers le monde selon sa profondeur trouvée sur la carte de profondeur, pour ensuite l'observer à partir d'une autre caméra, la profondeur trouvée pour ce point sur la carte de profondeur de cette autre caméra doit correspondre (en tenant compte de la position différente de cette caméra versus la caméra de référence). En clair, cela veut dire que l'on peut vérifier la validité des profondeurs trouvées sur une carte de profondeur en vérifiant si cette information est corroborée par les autres cartes de profondeur (des autres images). Donc, pour valider la profondeur d'un pixel de la carte de profondeur de la caméra de référence, il faut vérifier sa compatibilité avec la profondeur obtenue sur les cartes de profondeur des autres caméras. En plus, pour être plus rigoureux dans la validation, nous comparerons aussi la couleur du pixel de l'image de référence avec la couleur des pixels dans les autres images. Les détails de ce calcul se trouvent ci-dessous. A noter qu'il faut recommencer ce même processus pour tous les pixels de la carte de profondeur que l'on veut valider.

Tout d'abord, on prend note de la profondeur du pixel qui figure sur la carte de profondeur de la caméra de référence ($prof_{ref}$), ainsi que de la couleur du pixel correspondant dans l'image de référence ($coul_{ref}$). Ensuite, il faut dé-projeter ce pixel vers le monde 3D (figure 6.2).

$$P'_{ref} = \begin{bmatrix} X_{ref} \\ Y_{ref} \\ Z_{ref} \\ H_{ref} \end{bmatrix} = M_{ref}^{-1} \cdot \begin{bmatrix} x_{ref} \\ y_{ref} \\ 1 \\ -\frac{1}{prof_{ref}} \end{bmatrix} \tag{6.1}$$

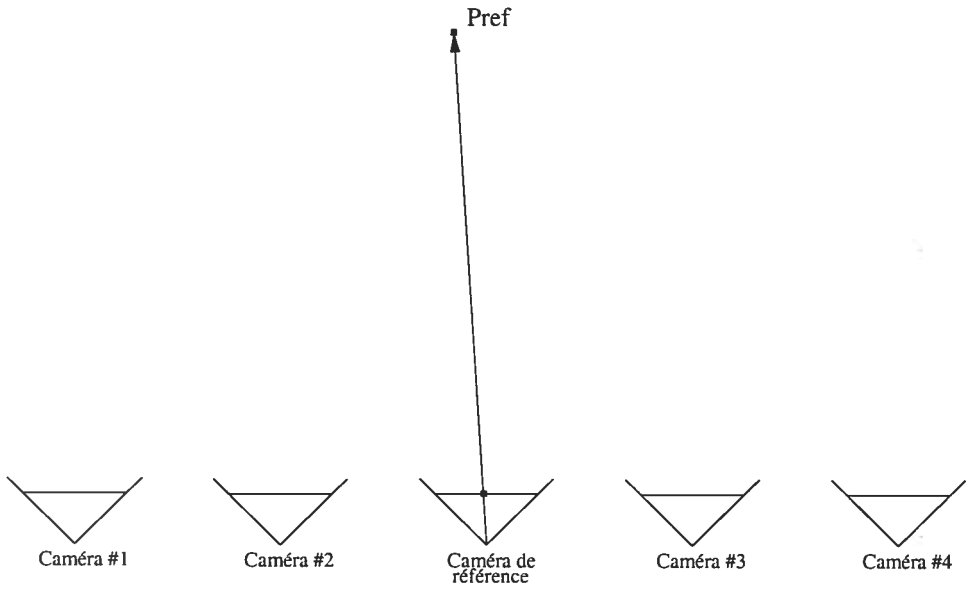



FIG. **6.2. Dé-projection du pixel de l'image de référence.**

où $P'_{ref}$ est le vecteur des coordonnées homogènes associées au point 3D $P_{ref}$ de la scène (du monde), $M_{ref}^{-1}$ est la matrice de dé-projection de la caméra de référence, $x_{ref}$ et $y_{ref}$ sont les coordonnées $x$ et $y$ du pixel de la carte de profondeur de la caméra de référence.

Ensuite il faut projeter ce point dans chacune des autres images (figure 6.3).

$$p'_n = \begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix} = M_n \cdot P'_{ref} \tag{6.2}$$

où $p'_n$ est le vecteur des coordonnées homogènes associées aux coordonnées pixel dans l'image $n$ et $M_n$ est la matrice de projection de la caméra $n$. $n$ représente le numéro de n'importe quelle des caméras que l'on utilise, à l'exception de la caméra de référence.

Pour obtenir les coordonnées pixel $x_n$ et $y_n$, il suffit de diviser les deux premiers

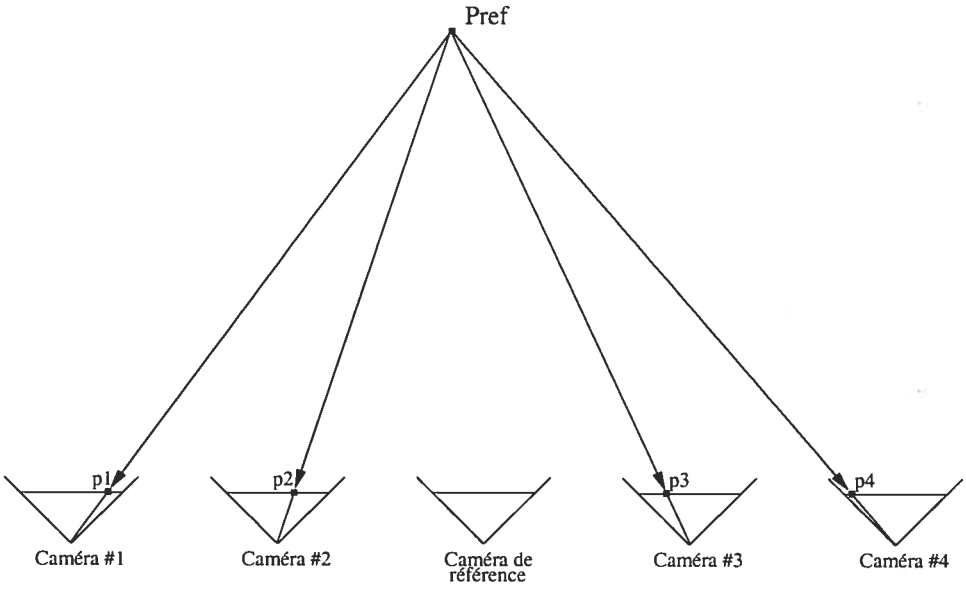



FIG. **6.3. Projection du point $P_{ref}$ dans les autres images.**

éléments du vecteur $p'_n$ par le troisième élément de ce même vecteur :

$$x_n = \frac{x}{z} \tag{6.3}$$

$$y_n = \frac{y}{z} \tag{6.4}$$

où $x_n$ et $y_n$ sont les coordonnées pixel $x$ et $y$ correspondant au point $P_{ref}$ projeté sur l'image $n$.

Ici on prend note de la couleur du pixel obtenue sur chacune des images ($coul_n$). Maintenant il faut dé-projeter chacun de ces pixels obtenus sur les images en utilisant la profondeur trouvée sur leur carte de profondeur respective (figure 6.4).

$$P'_n = \begin{bmatrix} X_n \\ Y_n \\ Z_n \\ H_n \end{bmatrix} = M_n^{-1} \cdot \begin{bmatrix} x_n \\ y_n \\ 1 \\ -\frac{1}{z_n} \end{bmatrix} \tag{6.5}$$

où $P'_n$ est le vecteur des coordonnées homogènes associées au point 3D $P_n$ de la scène (du monde), $M_n^{-1}$ est la matrice de dé-projection de la caméra $n$, $x_n$ et $y_n$ sont

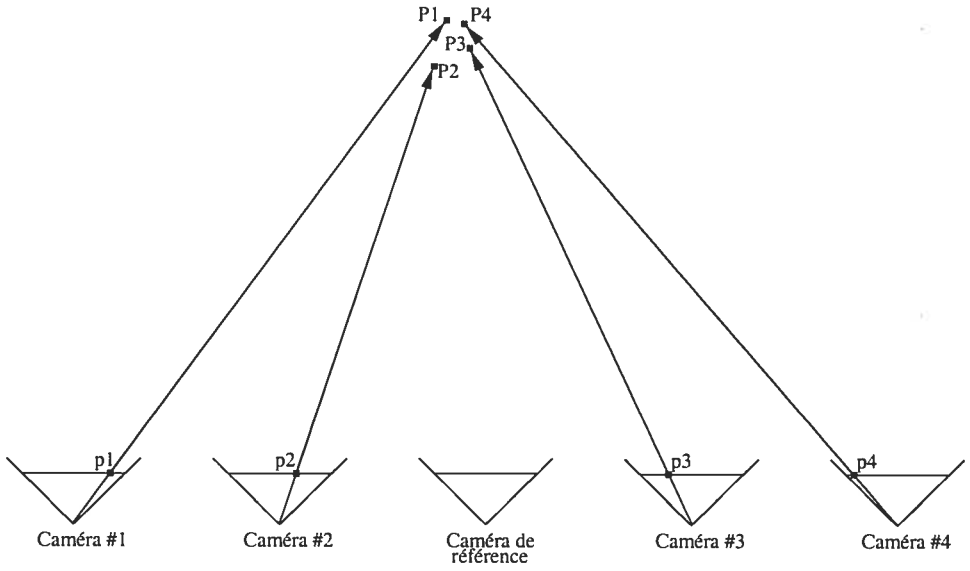



FIG. **6.4. Dé-projection des pixels des autres images.**

les coordonnées $x$ et $y$ du pixel de l'image $n$ et $z_n$ est la profondeur trouvée sur la carte de profondeur de la caméra $n$. $n$ représente le numéro de n'importe quelle des caméras que l'on utilise, à l'exception de la caméra de référence.

Finalement, il faut reprojeter tous les points $P_n$ dans la caméra de référence afin de pouvoir trouver la profondeur par rapport à la caméra de référence (figure 6.5). Si on prenait la profondeur directement sur la carte de profondeur de chacune des caméras, cette profondeur serait faussée puisque les caméras ne se trouvent pas toutes à la même distance des objets.

$$p'_{ref2} = \begin{bmatrix} x_{ref2} \\ y_{ref2} \\ z_{ref2} \\ h_{ref2} \end{bmatrix} = M_{ref} \cdot P'_n \tag{6.6}$$

où $p'_{ref2}$ est le vecteur des coordonnées homogènes associées aux coordonnées pixel dans l'image de référence, $P'_n$ est le vecteur des coordonnées homogènes obtenues de la dé-projection du pixel de la caméra $n$, et $M_{ref}$ est la matrice de projection de la

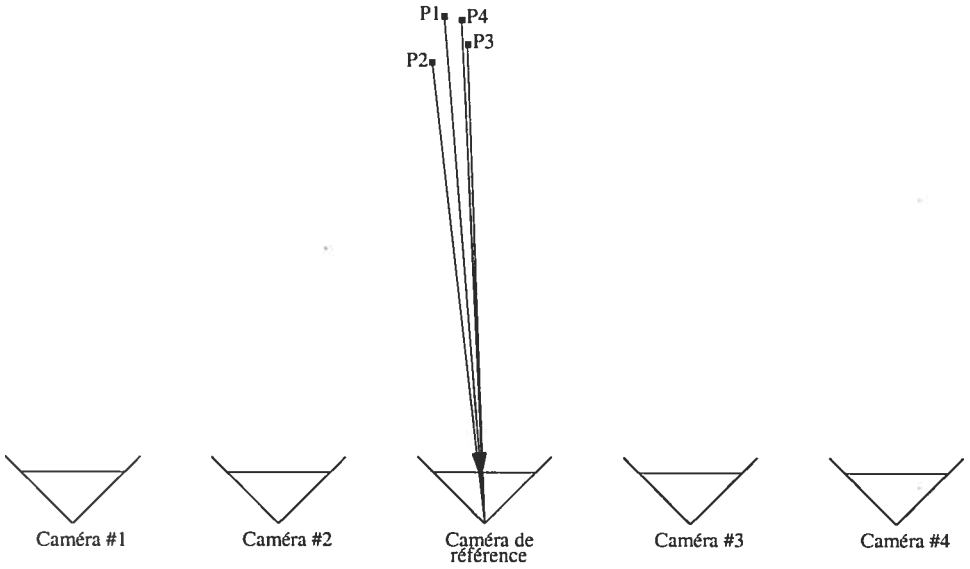



FIG. **6.5. Projection des points $P_n$ dans la caméra de référence.**

caméra de référence.

Enfin, la profondeur, par rapport à la caméra de référence, obtenue de chacune des cartes de profondeur des autres caméras est calculée ainsi :

$$prof_n = -\frac{z_{ref2}}{h_{ref2}} \tag{6.7}$$

où $prof_n$ est la profondeur de la carte de profondeur $n$ par rapport à la caméra de référence, $z_{ref2}$ et $h_{ref2}$ sont les deux derniers éléments du vecteur $p'_{ref2}$ obtenu à l'équation 6.6.

Maintenant nous avons tous les éléments. Mais avant de procéder à la comparaison proprement dite, il faut d'abord convertir les profondeurs en numéros de subdivisions, c'est-à-dire au numéro correspondant à la subdivision du volume de reconstruction à laquelle ils appartiennent (voir chapitre 4). Pour les convertir, il faut faire l'inverse de l'équation 4.7 pour chacune de ces profondeurs.

$$sub_n = \left\lfloor \sqrt{\frac{prof_n - z_{avant}}{z_{arriere} - z_{avant}}} * (nb\_sub - 1) + 0.5 \right\rfloor \tag{6.8}$$

où $sub_n$ est la subdivision correspondant à $prof_n$, $prof_n$ est la profondeur obtenue à

l'équation 6.7 pour la caméra $n$, $z_{avant}$ et $z_{arriere}$ sont respectivement les profondeurs correspondant à "l'avant" et "l'arrière" du volume de mise en correspondance, et $nb\_sub$ est le nombre de subdivisions totales du volume de mise en correspondance. $sub_{ref}$ correspondant à la subdivision pour $prof_{ref}$ peut être obtenue par la même équation.

Maintenant on peut procéder à la comparaison. Le résultat de chaque caméra est comparé individuellement avec le résultat de la caméra de référence. On commence par comparer les subdivisions (profondeurs).

$$pct\_diff\_sub = \frac{|sub_n - sub_{ref}|}{nb\_sub} \tag{6.9}$$

où $pct\_diff\_sub$ est la différence de subdivisions en pourcentage, $sub_n$ est la subdivision obtenue à l'équation 6.8 pour la caméra $n$, $sub_{ref}$ est la subdivision pour la caméra de référence, et $nb\_sub$ est le nombre total de subdivisions du volume de mise en correspondance.

Il faut maintenant comparer la couleur des pixels. Pour cela, il faut faire la somme des différences carrées des trois composantes RGB du pixel de l'autre image versus le pixel de l'image de référence.

$$pct\_diff\_coul = \frac{\sqrt{(R_n - R_{ref})^2 + (G_n - G_{ref})^2 + (B_n - B_{ref})^2}}{255 * \sqrt{3}} \tag{6.10}$$

où $pct\_diff\_coul$ est la différence de couleur en pourcentage, $R_n$, $G_n$ et $B_n$ sont les composantes RGB du pixel de l'image $n$, et $R_{ref}$, $G_{ref}$ et $B_{ref}$ sont les composantes RGB du pixel de l'image de référence. $n$ représente le numéro de n'importe quelle des caméras que l'on utilise, à l'exception de la caméra de référence. Pour avoir le pourcentage , il faut diviser $diff\_coul$ par $(255 * \sqrt{3})$ car il y a trois composantes couleur ($R$, $G$ et $B$), chacune ayant une intensité variant de 0 à 255. Si on travaille avec des images monochromes, il faut diviser $diff\_coul$ par 255 seulement.

Finalement, nous avons obtenu deux poucentages de validité : un pour la profondeur ($pct\_diff\_sub$) et un pour la couleur ($pct\_diff\_coul$). Il ne reste plus qu'à

comparer ces pourcentages selon un certain seuil pré-établi. Si ce seuil est trop bas, l'algorithme sera trop rigoureux et validera trop peu de pixels de la carte de profondeur de la caméra de référence. Par contre, ce qui est bien pire, si ce seuil est trop élevé alors l'algorithme ne sera pas assez rigoureux et plusieurs valeurs fausses sur la carte de profondeur de la caméra de référence seront considérées valides. Il s'avère moins pire, pour la suite de cette technique, d'invalider des bonnes profondeurs que de valider des profondeurs fausses. Il faut donc que ce seuil soit assez bas. En pratique, un seuil de 5% pour la profondeur et de 9% pour la couleur donne de bons résultats. Ces pourcentages sont fixés, c'est-à-dire qu'ils ne changent pas selon l'image. Donc, pour que la profondeur indiquée par le pixel de la carte de profondeur de la caméra de référence soit considérée valide par l'autre caméra avec laquelle on la compare, il faut que :

$$pct\_diff\_sub \leq 5\% \tag{6.11}$$

ET

$$pct\_diff\_coul \leq 9\%. \tag{6.12}$$

Si c'est le cas, on considère qu'une caméra a interprété ce pixel de la carte de profondeur de la caméra de référence comme étant valide. Dans le cas contraire on considère qu'une caméra l'a interprété invalide. On répète ce processus de comparaison (équations 6.9 à 6.12) pour toutes les autres caméras. A la fin, on compile le nombre de caméras considérées valides, et ensuite, le nombre de caméras considérées invalides. Le rapport entre ces deux nombres déterminera la validité finale. En pratique, deux fois plus de valides que d'invalides donne de bons résultats. Moins que ça (autant de valides que d'invalides par exemple) permet de valider des profondeurs fausses alors que plus que ça (trois fois plus de valides que d'invalides par exemple) ne permet pas de valider assez de bonnes profondeurs. Donc, en résumé, ce pixel de la carte de profondeur de la caméra de référence est considéré valide si :

$$nb\_valides \geq 2 * nb\_invalides \tag{6.13}$$

Sinon, il est considéré invalide.

Ainsi nous avons validé un seul pixel de la carte de profondeur de la caméra de référence. Donc, pour valider tous les pixels de cette carte de profondeur, il faudra répéter tout ce calcul (équations 6.1 à 6.13) pour chacun d'entre eux.

Il nous reste à souligner deux points importants.

Premièrement, contrairement à la fabrication de la carte de profondeur, il vaut mieux travailler sur un seul pixel à la fois au lieu d'un carré de 3x3 pixels (voir *Le coût de mise en correspondance*, chapitre 4). Nous avons fait cette constatation après de multiples tests, les résultats étant toujours meilleurs avec un carré d'un seul pixel. Nous n'avons pas fait une analyse approfondie sur le sujet et donc la raison de ce phénomène n'est pas encore très claire. L'intuition que nous avons est la suivante. Mettre en correspondance des carrés de 3x3 pixels a un effet de filtre passe-bas sur l'image puisque cela empêche des pixels bruités isolés de faire la correspondance dans les autres images. Un seul pixel erroné sur neuf n'a habituellement pas beaucoup d'impact sur le résultat et donc cela donne un effet de filtre passe-bas puisque les pixels aberrants ("outliers") n'ont pas d'impact sur la carte de profondeur trouvée. Or, si on prend des carrés de 3x3 pixels sur les cartes de profondeur pour fabriquer la carte de consistance, on se trouve à refaire un autre effet de filtre passe-bas. Ce que l'on constate alors, c'est que la carte de consistance considère invalide tous les pixels qui font partie des petits détails dans l'image, comme si on avait trop filtré. Il vaut donc mieux ne prendre qu'un seul pixel sur les cartes de profondeur pour fabriquer la carte de consistance.

Deuxièmement, si on utilise la technique "max-gauche-droite" (chapitre 5), il faut séparer la validation en deux : une pour les images de gauche et une pour les images de droite. En effet, s'il y a, pour un pixel de la carte de profondeur de la caméra de référence, un cas de demi-occlusion pour un des deux côtés, il est certain que ce pixel sera considéré comme étant invalide par toutes les caméras de ce côté puisqu'elles

ne "voient" pas cet élément de la scène. Or, la profondeur indiquée par ce pixel sur la carte de profondeur, laquelle a été évaluée par les caméras de l'autre côté (grâce à la technique "max-gauche-droite"), peut très bien être la bonne. Si les caméras du côté où il n'y a pas d'occlusion considèrent ce pixel comme étant valide, il devrait être considéré comme tel. Donc en résumé, il faut vérifier, pour chaque pixel, d'abord la validation avec les caméras à gauche seulement, et ensuite, la validation avec les caméras à droite seulement. La profondeur, indiquée par ce pixel de la carte de profondeur de la caméra de référence, doit être considérée valide si :

elle est considérée valide par les caméras de gauche
et que le pixel vient des caméras de gauche
OU
elle est considérée valide par les caméras de droite
et que le pixel vient des caméras de droite.

En pratique, puisque les caméras du côté où il y a une demi-occlusion vont probablement toutes considérer la profondeur de ce pixel invalide, on n'a pas besoin de savoir d'où provient le pixel. Si la profondeur est validée par les caméras de gauche ou les caméras de droite, alors elle est considérée valide.

## *6.2 Comment "verrouiller" la profondeur d'un pixel dans l'algorithme de flot maximum ?*

Précédemment, nous avons vu comment valider les pixels d'une carte de profondeur, à savoir quels sont, parmi ces pixels, ceux qui sont fiables (et donc probablement exacts) et ceux qui ne le sont pas. Le but de cette validation est de pouvoir conserver la profondeur que représente ces pixels fiables alors que l'on va ré-exécuter l'algorithme de flot maximum sur l'image avec un lissage fort. Sur la nouvelle carte de profondeur obtenue, la profondeur représentée par ces pixels fiables ne devra pas

avoir changée. Il faut donc "verrouiller" la profondeur des pixels fiables dans l'initialisation du problème de flot maximum. Petit rappel : la solution obtenue est une coupe minimum du "cube", ce qui correspond aux endroits où il y a eu saturation des arcs de disparité (voir *Le flot maximum*, chapitre 4). Il faut donc s'assurer, en exécutant de nouveau l'algorithme de flot maximum, que les arcs de disparité correspondant aux pixels fiables font partie de la coupe minimum. Pour cela, il suffit d'initialiser les arcs de disparité correspondant aux pixels fiables comme suit :

$$c_{disp} = \begin{cases} 0 & \text{si l'arc correspond à la profondeur validée de ce pixel} \\ \infty & \text{pour tous les autres arcs de disparité de ce pixel} \end{cases}$$

où $c_{disp}$ représente la capacité d'un arc de disparité d'un pixel fiable.

Tous les autres arcs du "cube" doivent être initialisés exactement selon la procédure habituelle (équations 4.9 et 4.11). Ainsi nous sommes assurés de conserver les pixels fiables à leur profondeur validée. Étant donné que ces profondeurs validées sont absolument sûres (aucune ambiguïté), il est probable qu'ils influenceront les pixels voisins par la contrainte de lissage.

La technique "consistance de profondeur" permet d'obtenir une carte de profondeur d'une qualité s'approchant du lissage "idéal" tout en ne fabriquant que deux cartes de profondeur au total : une à un lissage minimal et l'autre à un lissage maximal. Les figures 6.1 et 6.6 démontrent bien ce fait. On peut constater que la carte de profondeur finale obtenue par notre méthode se compare très bien avec la carte de profondeur obtenue d'un lissage "idéal". Ce lissage a été évalué en comparant le résultat de différents lissages et en conservant la carte de profondeur que nous avons jugée la meilleure. Cette façon de faire est extrêmement longue et exige l'intervention d'un utilisateur pour finalement donner un résultat semblable à celui obtenu par "consistance de profondeur". Davantage de résultats de cette méthode sont présentés au chapitre suivant.

La technique "consistance de profondeur" remplace donc un seuil (le choix du lissage) par trois autres seuils : le pourcentage de différence de subdivisions (équation 6.11), le pourcentage de différence de la couleur (équation 6.12) et le nombre de fois de plus de caméras valides que d'invalides que l'on doit obtenir (équation 6.13). A première vue, il semble que nous avons juste compliqué le problème d'avantage. Cependant, contrairement au niveau de lissage, les trois seuils de notre technique sont indépendants du contenu de l'image. Cela signifie que ces valeurs sont fixes et conviennent à toutes les images. Tous les résultats présentés dans ce mémoire l'ont été à partir des trois mêmes valeurs de seuils que nous avons établies. Cette technique est donc complètement automatique, et c'est ce qui la rend vraiment intéressante.

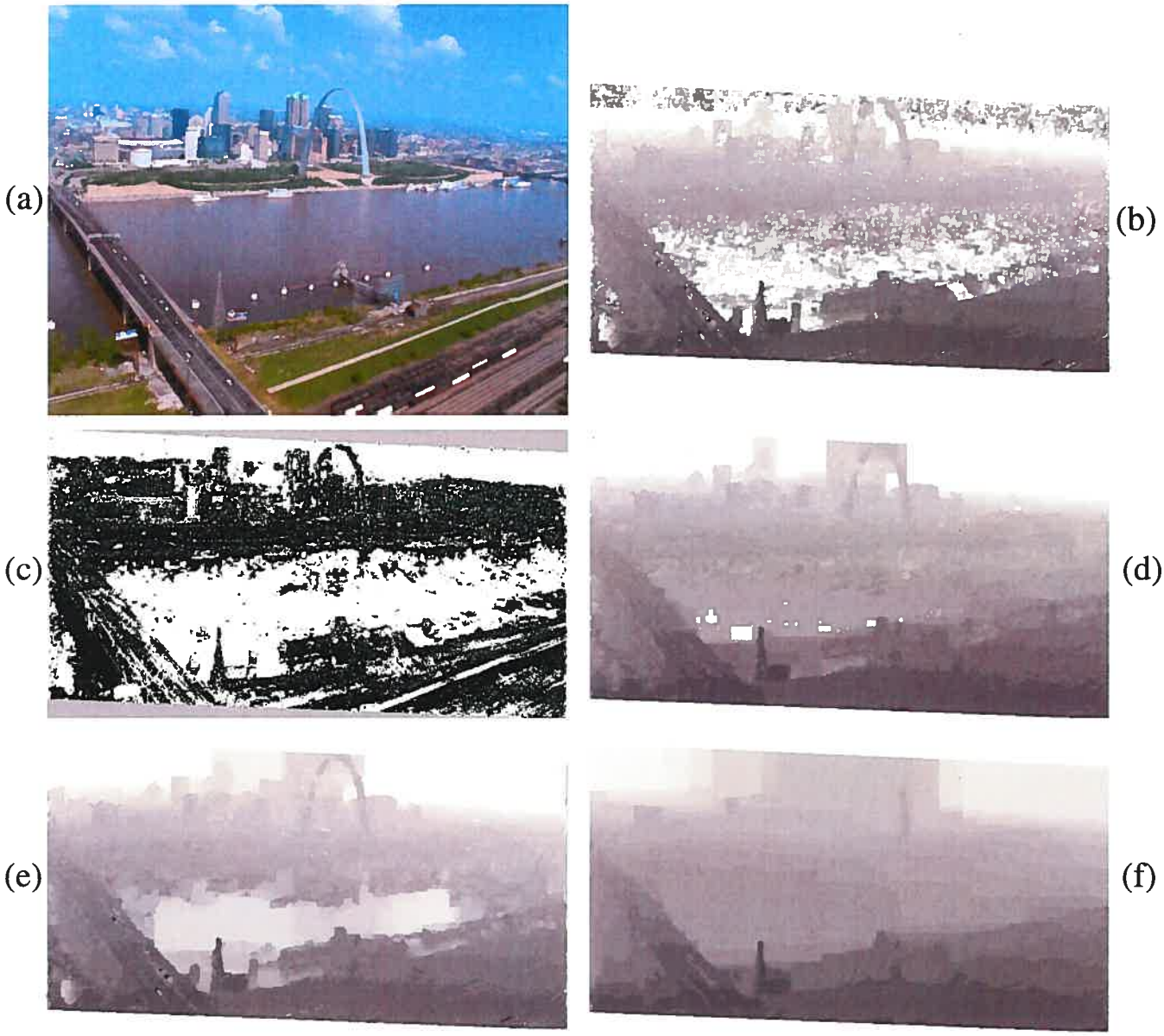



FIG. **6.6. La technique consistance de profondeur sur la séquence de l'arche de St-Louis. (a) image de l'arche de St-Louis, (b) carte de profondeur avec un lissage de 1, (c) carte de consistance associée à (b) (les pixels noirs sont valides), (d) carte de profondeur finale obtenue par la technique "consistance de profondeur" (lissage de 1 000 000 avec verrouillage de profondeur), (e) carte de profondeur obtenue d'un lissage "idéal" (30), (f) carte de profondeur obtenue d'un lissage trop fort (500). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

Chapitre 7

# TRAITEMENT ET RÉSULTATS

Le but ultime de notre méthode est d'ajouter un objet virtuel dans une séquence d'images. Ce chapitre fait la présentation des différentes étapes pour arriver à cette fin, avec les résultats intermédiaires obtenus à partir de l'un de nos tests. Le test en question visait à ajouter une poubelle virtuelle dans notre laboratoire que nous avons préalablement filmé. Plusieurs autres résultats sont aussi présentés à la fin de ce chapitre. Produire tous ces résultats a pris beaucoup de temps : dépendant de la séquence vidéo utilisée, le temps d'exécution de toute la méthode (avec un seul CPU 1.2 GHz) variait de un à dix jours. Mais cela aurait été encore bien plus long si nous n'avions pas bénéficié de deux techniques qui permettent d'accélérer beaucoup le traitement : le fenêtrage et le parallélisme.

## *7.1 Techniques pour accélérer le traitement*

### *7.1.1 Technique #1 : Le fenêtrage (clipping) autour de l'objet virtuel*

La résolution du flot maximum prend plus de 60% du temps de toute la méthode. Plus le problème du flot maximum est grand, plus l'algorithme met du temps à le résoudre. Si nous pouvions réduire le problème, le temps de calcul serait réduit d'autant. Pour la dernière étape du traitement (intégration de l'objet virtuel dans l'image), seuls les pixels qui touchent à l'objet virtuel sont importants ; les autres sont inutiles. Donc, pour accélérer le traitement, on peut se concentrer sur les pixels qui touchent à l'objet virtuel en fabriquant la carte de profondeur sur ceux-ci seulement. Cette carte de profondeur étant habituellement (tout dépend de la taille de l'objet virtuel) beaucoup plus petite que l'image entière, on accélère ainsi beaucoup le traitement. Par

contre, dans l'algorithme de flot maximum, comme dans tous les algorithmes utilisant une contrainte de lissage, les pixels influencent leur voisinage. Théoriquement, avec l'algorithme de flot maximum, s'il y avait énormément d'ambiguïtés dans l'image, un pixel en haut à gauche pourrait influencer la profondeur attribuée au pixel en bas à droite. Faire une carte de profondeur sur les seuls pixels qui touchent à l'objet virtuel pourrait limiter l'efficacité du lissage. En pratique cependant, il est rare qu'un pixel en influence un autre au-delà de 5 pixels de distance. Dans notre cas, pour être sûr de ne pas perdre trop d'information de l'image, nous avons fixé cette distance "tampon" à 10 pixels. Pour ce faire, il faut procéder à une dilatation (réf. [32], [31]) de cette distance tampon autour de l'objet virtuel, et transférer les coordonnées pixel obtenues sur l'image originale. Seuls les pixels de l'image contenus dans cette dilatation doivent faire partie du problème de flot maximum. Il y a deux volets à ce fenêtrage. Le premier volet consiste à découper un rectangle autour de la dilatation de l'objet virtuel. La raison est que les cartes de profondeur, tout comme les images d'ailleurs, doivent être rectangulaires pour être sauvegardées dans un fichier. De plus, la résolution du flot maximum, telle qu'elle a été présentée au chapitre 4, ne peut se faire que sur un "prisme rectangulaire". Donc, pour trouver les coordonnées de ce rectangle autour de l'objet dilaté, il suffit de prendre le minimum et le maximum en largeur et en hauteur. Faire cela permet de réduire la taille du problème à résoudre. Mais si l'objet n'est pas rectangulaire (un objet en forme de "T" par exemple), il y aura beaucoup de pixels dans ce rectangle qui ne font pas partie de la dilatation de l'objet virtuel. Ceci nous amène au deuxième volet de ce fenêtrage. La seule chose que l'on peut faire dans ces cas-là est d'initialiser le problème du flot maximum de sorte que l'algorithme ne perde pas du temps à tester différentes profondeurs pour ces pixels. Pour faire cela, il suffit simplement de mettre à zéro tous les arcs qui relient ces pixels. Ainsi, nous réduisons beaucoup le temps de calcul pour fabriquer une carte de profondeur par l'algorithme de flot maximum.

### *7.1.2 Technique #2 : Le traitement des images en parallèle*

Dans la prochaine section de ce chapitre, il sera question des différentes étapes de la méthode, de la prise du film jusqu'à l'intégration de l'objet virtuel dans la séquence. Passer à travers toutes les étapes peut s'avérer long, même en faisant du fenêtrage. En particulier, les étapes #4-A et #4-C (figure 7.1) peuvent prendre beaucoup de temps. Heureusement, les étapes #4-A à #5 sont indépendantes et peuvent donc se faire en parallèle. La seule exception est le passage de l'étape #4-A à #4-B. En effet, puisque la fabrication des cartes de consistance nécessite la comparaison des cartes de profondeur obtenues à l'étape précédente, il faut que l'étape #4-A soit terminée pour toutes les cartes de profondeur utilisées avant de passer à l'étape #4-B. L'utilisation du parallélisme permet de traiter plusieurs images dans un temps raisonnable. En fait, l'efficacité dépend alors du nombre d'ordinateurs à notre disposition. Supposons par exemple que nous voulons traiter une séquence de 100 images et que l'étape #4-C prend 15 minutes pour traiter une seule image. Si nous ne possédons qu'un seul ordinateur, l'étape #4-C prendra 1 500 minutes (25 heures) à s'exécuter. Par contre, si nous possédons 100 ordinateurs, 15 minutes suffiront pour traiter toutes les images. Chacune des 100 images prendra 15 minutes à s'exécuter, mais elles seront traitées en parallèle. Pour les grandes entreprises, acheter une centaine d'ordinateurs n'est pas quelque chose de prohibitif. Mais même si, comme dans notre laboratoire, nous ne possédons qu'un nombre restreint d'ordinateurs, le fait que le traitement peut se faire en parallèle permet de diviser le temps d'exécution d'autant de fois qu'il y a d'ordinateurs. Un dernier détail : s'il y a plus d'ordinateurs que d'images à traiter, les ordinateurs supplémentaires ne servent à rien. Dans l'exemple précédent, si nous aurions voulu traiter une seule image, même avec 100 ordinateurs, l'étape #4-C aurait quand même pris 15 minutes à s'exécuter.

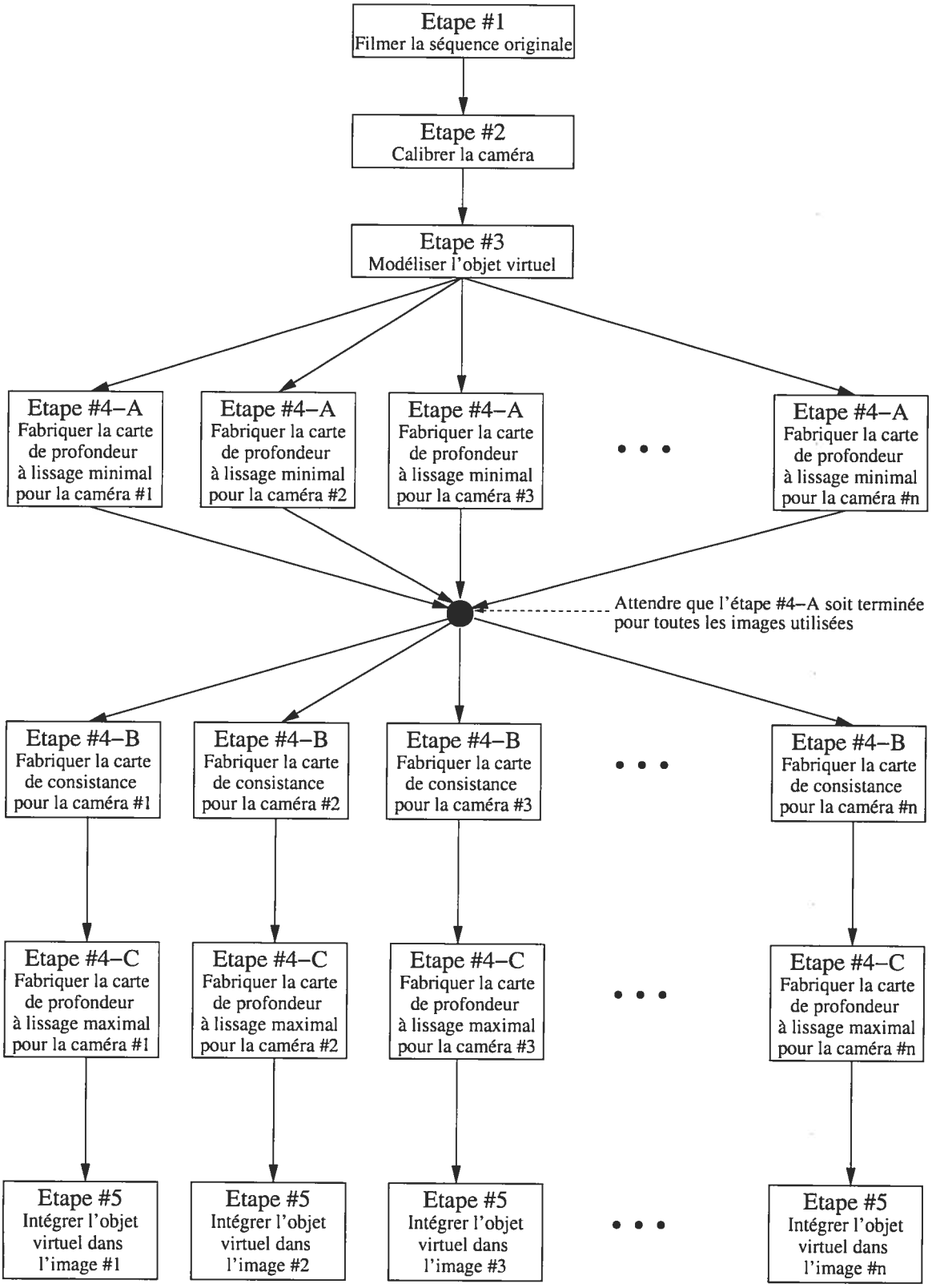



FIG. **7.1. Organigramme de notre méthode.**

## *7.2 Résumé des étapes de la méthode avec les résultats intermédiaires*

Dans cette section, les étapes de la méthode, de la première à la dernière, seront présentées avec les résultats intermédiaires qui ont été obtenus avec l'un de nos tests. Ce test consistait à intégrer une poubelle virtuelle dans notre laboratoire que nous avions filmé. Une simple remarque sur cette séquence vidéo : des objets réels (chaises, valises) ont été placés volontairement dans la scène afin de créer des situations d'occlusion ; ceci dans le but de pouvoir juger de l'efficacité de notre méthode. Vous remarquerez aussi que l'étape #4 a été divisée en trois sous-étapes (#4-A, #4-B et #4-C) afin de bien identifier tout le processus de la technique "consistance de profondeur".

### *7.2.1 Etape #1 : Filmer la séquence originale*

Cette étape consiste à filmer la scène avec les contraintes décrites au chapitre 2. Le film obtenu, dans notre cas, était sous la forme d'un fichier ".avi". Il a fallu, par la suite, séparer ce film en images. Ce processus a été effectué par le programme "MPlayer" (voir en annexe pour les détails complets sur la version du logiciel).

En entrée : rien.

En sortie : les images de la séquence originale.

### *7.2.2 Etape #2 : Calibrer la caméra de façon automatique*

D'abord, la calibration est effectuée par le logiciel "MatchMover" (voir en annexe pour les détails complets sur la version du logiciel). Ensuite il faut transformer les informations obtenues de ce logiciel (fichier ".xsi") en des matrices de calibration (fichier ".mat"). Ce calcul est expliqué plus en détail dans le chapitre 3.

En entrée : les images de la séquence originale.

En sortie : un fichier ".xsi" obtenu de "MatchMover" (utilisé dans l'étape #3 seulement), et un fichier ".mat" qui contient la matrice de projection et la matrice

FIG. **7.2. Etape #1 : La scène originale.**

de dé-projection de chacune des images de la séquence (ce fichier sera utilisé dans les étapes #4-A à #4-C).

### *7.2.3 Etape #3 : Modéliser l'objet virtuel*

Cette étape consiste à modéliser un objet virtuel en 3D. Dans notre cas, nous avons utilisé le logiciel "XSI" (SoftImage). La calibration est utilisée dans "XSI", ce qui permet de reproduire dans l'univers synthétique la trajectoire de la caméra dans la scène réelle. Ainsi les objets virtuels sont observés du même point de vue que lors du tournage de la séquence. Les détails de cette modélisation se trouvent au chapitre 2.

En entrée : le fichier ".xsi" obtenu de "MatchMover", et les images de la séquence originale. Les images ne sont utilisées que pour faciliter, pour l'utilisateur, le positionnement de l'objet virtuel dans la scène.

En sortie : les images de l'objet virtuel modélisé, et un fichier ".Zpic" pour chaque

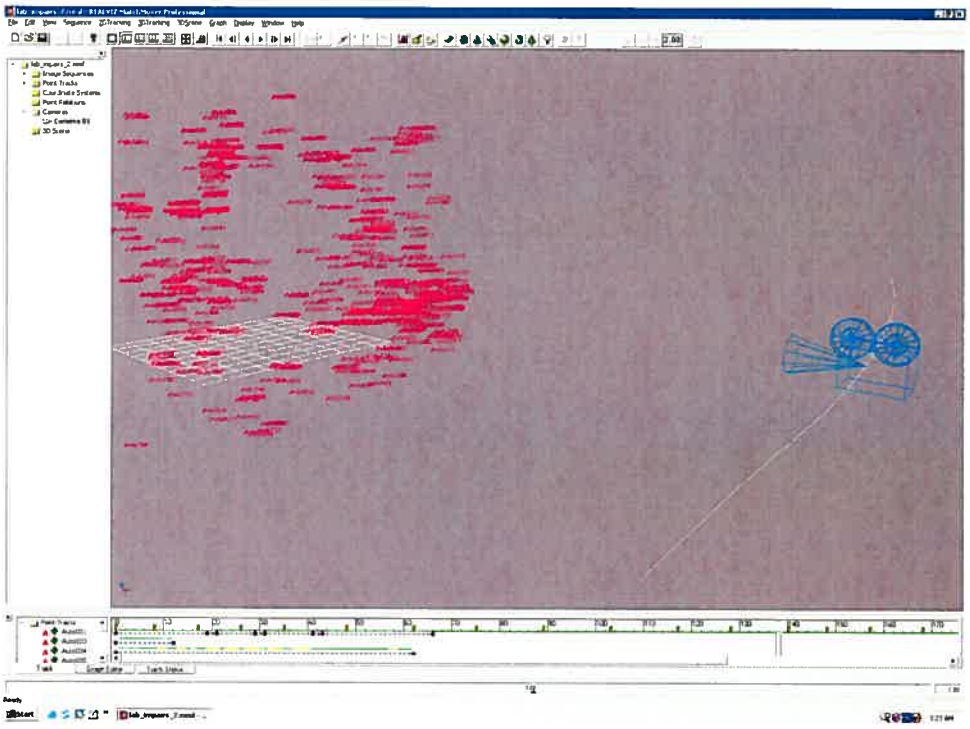


FIG. **7.3. Etape #2 : La calibration obtenue dans "MatchMover".**

image contenant la profondeur de tous les pixels de ces images.

### *7.2.4 Etape #4-A : Fabriquer les cartes de profondeur des différentes images de la séquence, en utilisant un lissage minimal*

Première étape de la technique "consistance de profondeur" (chapitre 6), elle consiste à fabriquer une carte de profondeur, par l'algorithme de flot maximum (chapitre 4), avec un lissage de 1. En fait, l'algorithme de flot maximum doit être exécuté deux fois : une fois avec les caméras à gauche et l'autre fois avec les caméras à droite. Ensuite, conformément à la technique "max-gauche-droite" (chapitre 5), on fabrique la carte de profondeur finale (pour cette étape) en conservant la plus grande profondeur entre ces deux cartes de profondeur. Il faut aussi noter que juste avant d'initialiser le problème du flot maximum, on procède au fenêtrage autour de l'objet

FIG. **7.4. Etape #3 : Le rendu de l'objet virtuel (obtenu de "XSI").**

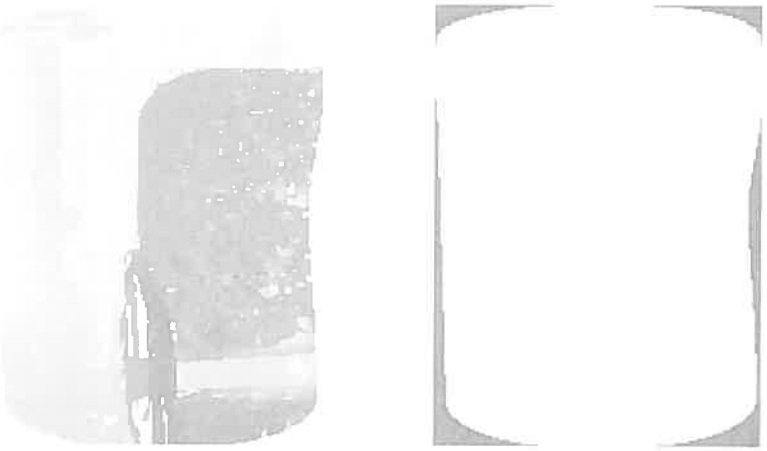
virtuel dans les images de la séquence originale (voir plus haut dans ce chapitre pour plus de détails).

En entrée : les images de la séquence originale, le fichier ".mat" contenant les matrices de calibration, et l'ensemble des fichiers ".Zpic" qui serviront pour le fenêtrage (aux endroits où il n'y a pas d'objet virtuel, la profondeur est maximale (habituellement 10 000 dans "XSI")).

En sortie : les cartes de profondeur correspondant aux images de la séquence, et les cartes de fenêtrage pour chacune d'entre elles. Ces cartes de fenêtrage sont en fait des images binaires qui définissent la zone occupée par la dilatation de l'objet virtuel.

### *7.2.5 Etape #4-B : Fabriquer les cartes de consistance correspondant aux cartes de profondeur*

C'est dans cette étape que l'on va valider, conformément à la technique "consistance de profondeur" (chapitre 6), les pixels des cartes de profondeur qui ont été



(a) (b)

FIG. **7.5. Etape #4-A : (a) La carte de profondeur obtenue du lissage minimal (lissage de 1), (b) La carte de fenêtrage. Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

fabriquées à l'étape précédente.

En entrée : les images de la séquence originale, le fichier ".mat" contenant les matrices de calibration, les cartes de profondeur et les cartes de fenêtrage.

En sortie : les cartes de consistance.

### *7.2.6 Etape #4-C : Fabriquer les cartes de profondeur des différentes images de la séquence, en utilisant un lissage maximal, tout en "verrouillant" la profondeur des pixels qui auront été validés à l'étape précédente*

C'est la dernière étape de la technique "consistance de profondeur" (chapitre 6). Il faut faire exactement comme dans l'étape #4 à l'exception de deux éléments :

1. Le lissage est de 1 000 000.
2. Les problèmes de flot maximum doivent être initialisés de sorte que les profondeurs représentées par les pixels validés (sur les cartes de consistance) sont verrouillées, conformément à la technique "consistance de profondeur".

FIG. **7.6. Etape #4-B : La carte de consistance (les pixels noirs sont valides).**

En entrée : les images de la séquence originale, le fichier ".mat" contenant les matrices de calibration, les cartes de profondeur obtenues du lissage minimal et les cartes de consistance.

En sortie : les cartes de profondeur finales.

### *7.2.7 Etape #5 : Intégrer l'objet virtuel dans les images*

Cette étape consiste à intégrer l'objet virtuel dans la scène avec la bonne occlusion, en comparant la profondeur de l'objet virtuel (fichier ".Zpic") avec la profondeur de la scène obtenue dans les cartes de profondeur. Ce processus est décrit plus en détail au chapitre 2.

En entrée : les images de la séquence originale, les images de l'objet virtuel, l'ensemble des fichiers ".Zpic" pour la profondeur de l'objet virtuel, et les cartes de profondeur finales (obtenues de l'étape #6) pour la profondeur de la scène.

En sortie : les images de la séquence avec l'objet virtuel intégré.

Note : La scène du laboratoire est relativement facile. Comme on peut le constater, la poubelle virtuelle a été très bien intégrée dans la scène.

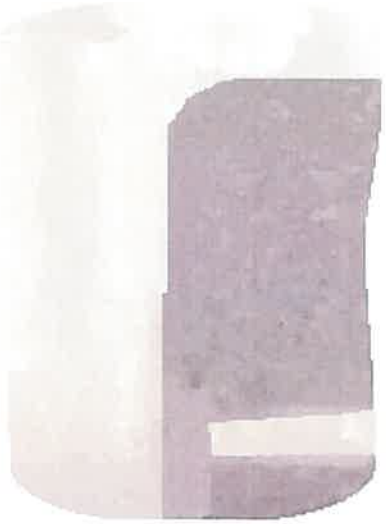


FIG. 7.7. **Etape #4-C : La carte de profondeur obtenue du lissage maximal (1 000 000) avec verrouillage de profondeur. Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

FIG. 7.8. **Etape #5 : L'image finale (réalité augmentée). Le vidéo complet est disponible à l'adresse : "http ://www.iro.umontreal.ca/∼vision/hollowap/".**

### *7.3 Autres résultats*

Dans cette section sont présentés les résultats que nous avons obtenus de diverses séquences vidéo. Ces séquences vidéo sont divisées en trois groupes :

1. Les scènes virtuelles.

Ces scènes ont été obtenues par le logiciel "VTK" (voir en annexe pour les détails complets sur la version du logiciel). Les résultats obtenus sont les meilleurs puisque les images ne contiennent pas de bruit, les surfaces dans la scène sont parfaitement lambertiennes, et surtout, la calibration est parfaite (elle est fournie par le logiciel VTK). Les étapes #1 à #3 sont remplacés par la modélisation, dans VTK, de la scène virtuelle et de l'objet virtuel à intégrer. Le grand intérêt de travailler avec ces images est que l'on peut comparer le résultat final obtenu par notre méthode, avec le résultat "parfait" en faisant directement un rendu de l'objet virtuel dans la scène virtuelle.

2. Les scènes réelles avec images rectifiées.

Ces scènes ont été captées par des caméras qui étaient toutes alignées à la même hauteur, et leurs axes de visée étaient parallèles entre eux et perpendiculaires à la scène. L'avantage des images rectifiées est que la recherche, pour faire la mise en correspondance d'un pixel, est restreinte à la même ligne dans l'image, dans toutes les images. De plus, on a habituellement pris soin de mesurer la distance qui sépare les différentes caméras, ce qui permet d'obtenir une meilleure calibration que par l'analyse de points saillants qui est utilisé par le logiciel "MatchMover". Nous obtenons donc de meilleurs résultats que sur les images non-rectifiées, mais les contraintes imposées sont d'une telle rigidité qu'aucune séquence vidéo "normale" ne peut s'y conformer. En d'autres mots, les images rectifiées ne sont captées que pour tester des algorithmes de vision par ordinateur. Il est possible, avec certaines méthodes, de rectifier des images qui ne le sont pas au départ (réf. [30], [23]). Cependant, cela introduit toutes sortes d'autres problèmes. Les images rectifiées que nous avons testées étaient toutes des "vraies"

images rectifiées au départ.

Note : Les séquences d'images rectifiées que nous avons utilisées nous ont été fournies avec les cartes de profondeur mesurées à la main et discrétisées sur 16 niveaux de profondeur. Ces cartes de profondeur peuvent être considérées comme étant "parfaites", ce qui nous permet de comparer notre méthode plus facilement.

3. Les scènes réelles captées dans une séquence vidéo "normale".

Les séquences vidéo "normales" sont celles qui ont été filmées dans des conditions réelles. Toutes les séquences vidéo que nous avons nous-mêmes filmées en font partie. Par exemple, la séquence du laboratoire, dans laquelle nous avons intégré la poubelle virtuelle, fait partie de ce groupe. Les seules contraintes que nous devions respecter sont celles décrites au chapitre 2 (*Etape #1 : Filmer la séquence originale*).

Remarque : il n'est pas facile d'évaluer la qualité d'une carte de profondeur. Dans le cas des scènes réelles captées dans une séquence vidéo "normale", nous n'avons pas le résultat "idéal" avec lequel on peut comparer le résultat obtenu par notre méthode. Dans les cas où il est possible d'obtenir ce résultat "idéal" (séquences virtuelles ou avec images rectifiées), un autre problème se pose : comment comparer les résultats entre eux? Certains ont proposé de faire une comparaison pixel à pixel (réf. [26]). Mais cette façon de faire n'est pas vraiment adéquate. La perception humaine est très sensible aux contrastes forts : dix pixels qui varient de dix tons de gris ne seront presque pas perceptibles alors qu'un seul pixel variant de cent tons de gris va immédiatement capter l'attention de la personne qui regarde l'image. Il n'existe pas vraiment de mesures d'erreurs qui tiennent vraiment compte de toute la complexité de la perception humaine. Et justement, le but de la méthode qui a été présentée est de produire les meilleures images possibles du point de vue de la perception humaine. Nous avons donc décidé de laisser le soin au lecteur d'évaluer par lui-même la qualité des résultats que nous avons obtenus.

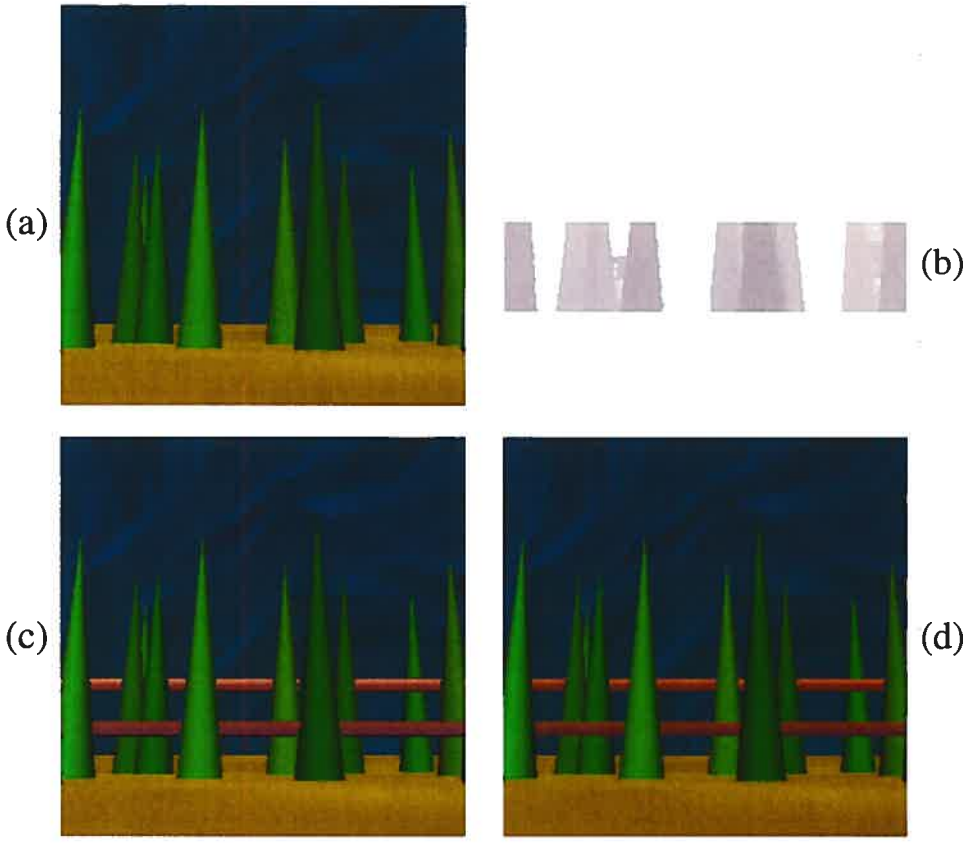



FIG. 7.9. **Scène virtuelle : La forêt virtuelle. Un cas très facile. (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée), (d) résultat "parfait" (rendu VTK). On peut remarquer de minimes différences entre les deux images ((c) et (d)). Nombre d'images utilisées : 11 (5 à droite, 5 à gauche et l'image de référence).**

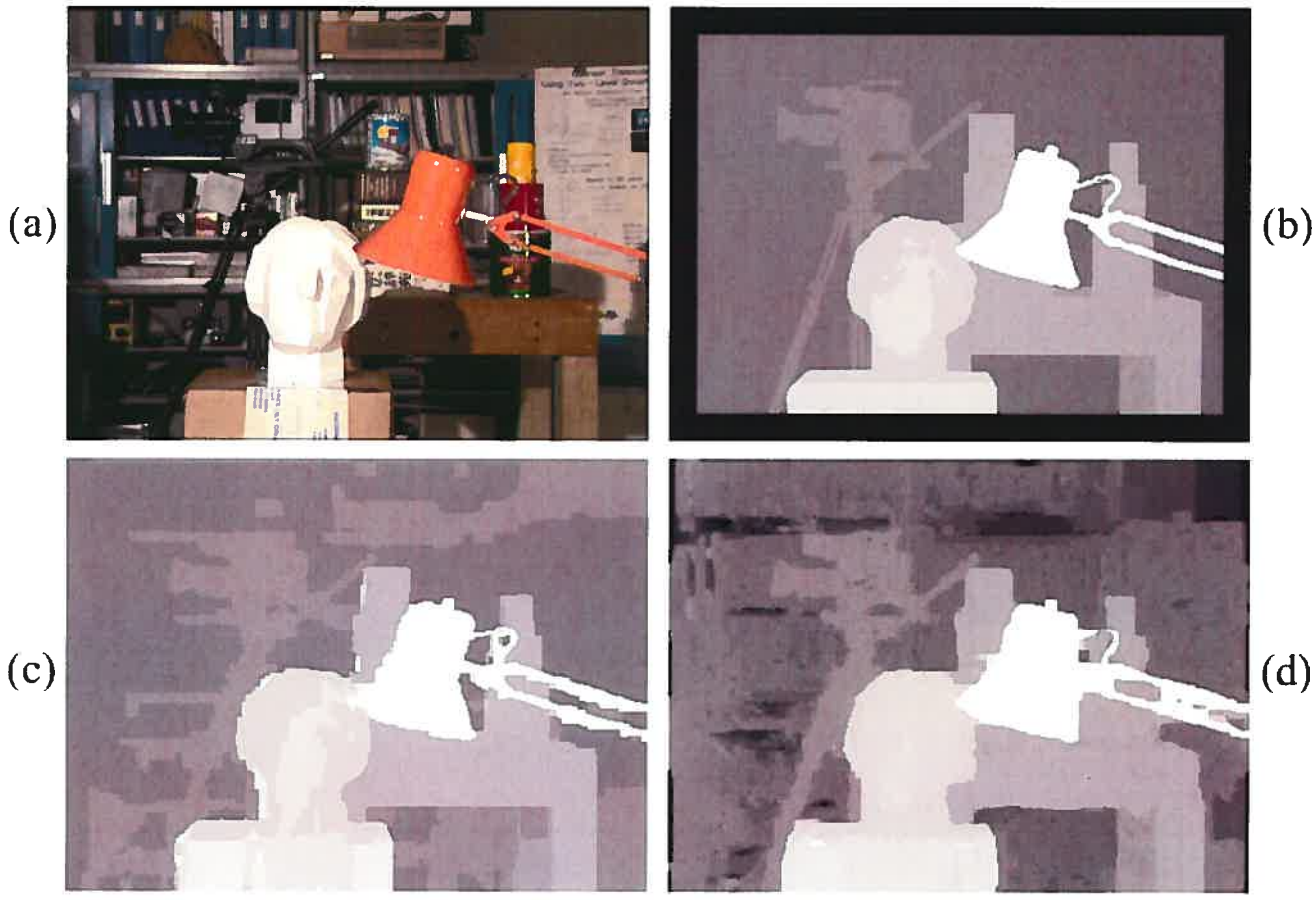



FIG. 7.10. **Scène réelle avec images rectifiées : Tsukuba. Un cas plutôt facile, sauf qu'il y a présence de quelques reflets (surfaces non-Lambertiennes) qui nuisent à la fabrication de la carte de profondeur. (a) image originale, (b) carte de profondeur mesurée à la main et discrétisée sur 16 niveaux de profondeur, (c) la meilleure carte de profondeur pour cette séquence selon Middlebury (réf. [26]), obtenue par l'algorithme "Belief Propagation" (réf. [27]), (d) carte de profondeur finale obtenue par notre méthode. Nombre d'images utilisées : 5 (2 à droite, 2 à gauche et l'image de référence).**

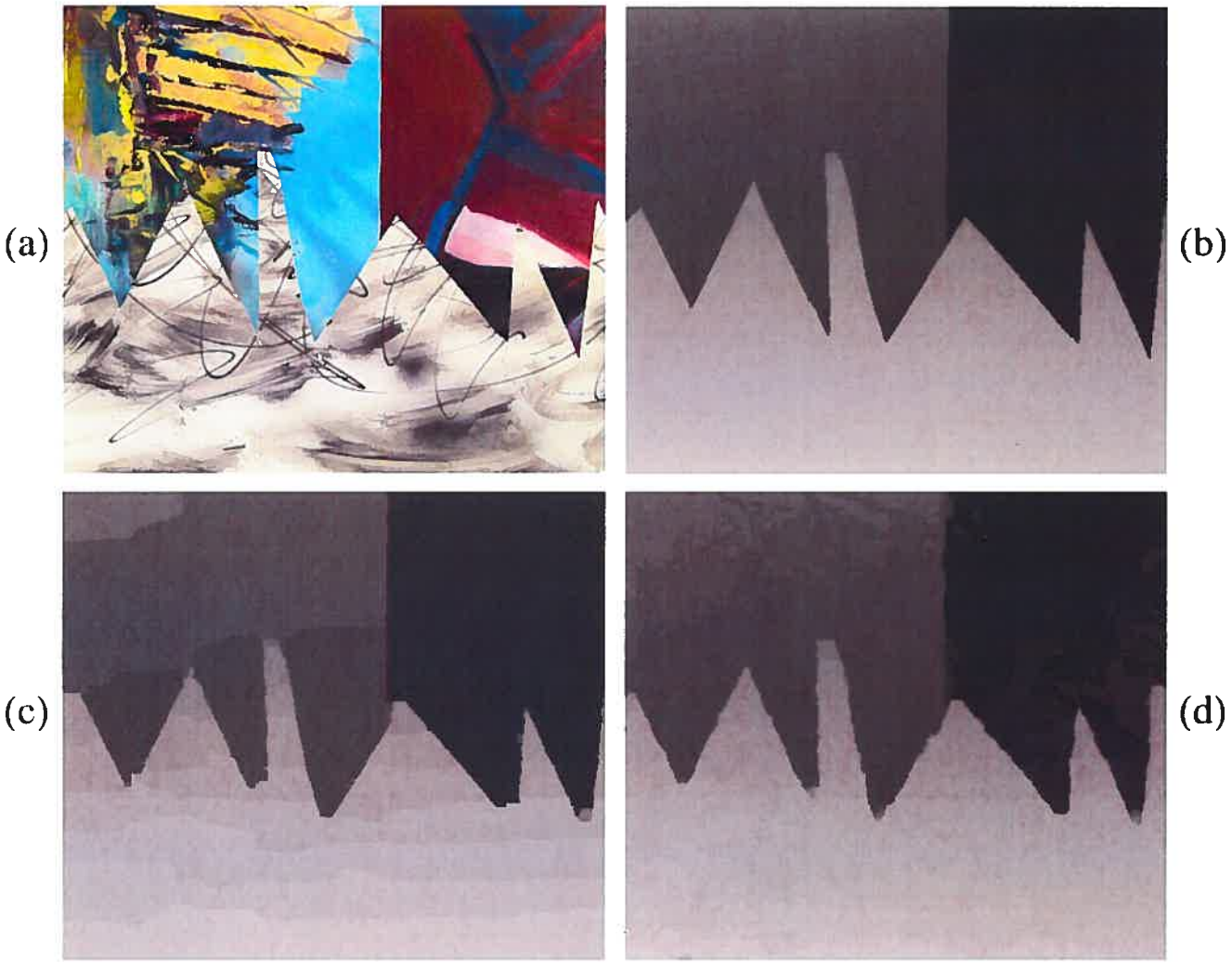



FIG. 7.11. **Scène réelle avec images rectifiées : Sawtooth. Un cas facile mais présentant des discontinuités difficiles. (a) image originale, (b) carte de profondeur mesurée à la main et discrétisée sur 16 niveaux de profondeur, (c) la meilleure carte de profondeur pour cette séquence selon Middlebury (réf. [26]), obtenue par l'algorithme "Layered Stereo" (réf. [17]), (d) carte de profondeur finale obtenue par notre méthode. Nombre d'images utilisées : 9 (4 à droite, 4 à gauche et l'image de référence).**

(a)

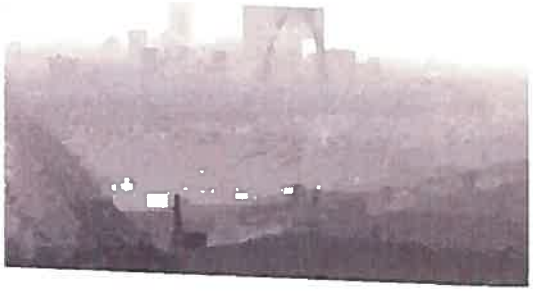
(b)

(c)

FIG. 7.12. **Scène réelle captée dans une séquence vidéo "normale" : Le laboratoire. Le "rideau" virtuel passe tout juste derrière la chaise grise et les valises. C'est un cas relativement facile malgré certains endroits difficiles. Note : C'est la même séquence originale que la poubelle virtuelle. Le fait d'intégrer un "rideau" très large dans la scène nous permet de mieux pouvoir évaluer le résultat. (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

(a)

(b)



(c)

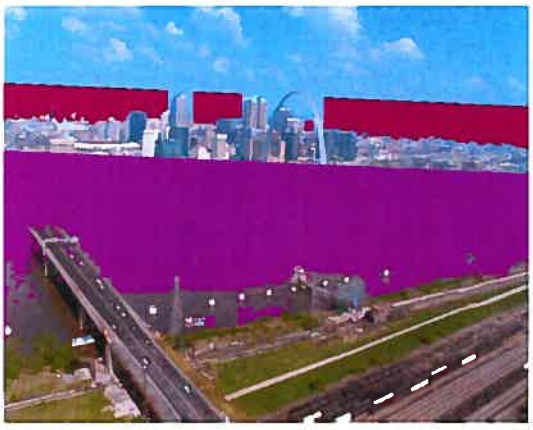


FIG. 7.13. **Scène réelle captée dans une séquence vidéo "normale" : L'arche de St-Louis. Le "rideau" virtuel du haut passe derrière la ville alors que le "rideau" virtuel du bas passe au milieu de la rivière. Un cas très difficile pour deux raisons. D'abord, le ciel est de couleur uniforme. Ensuite, et surtout, la rivière n'est ni statique (vagues qui bougent d'une image à l'autre), ni Lambertienne. (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

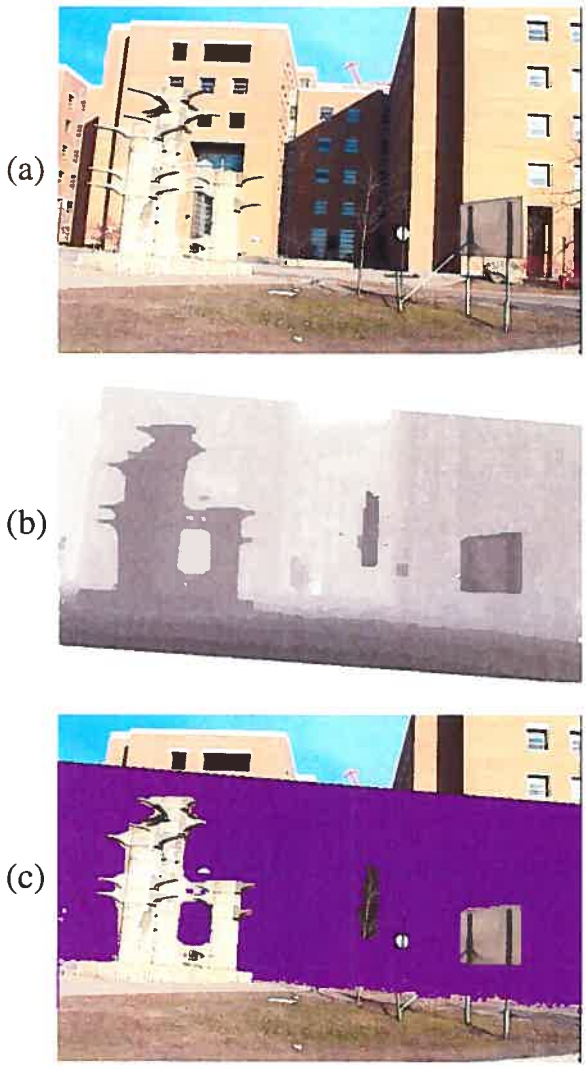



FIG. 7.14. **Scène réelle captée dans une séquence vidéo "normale" : L'Isolateur (avec le rideau). Le "rideau" virtuel passe sur la rue arrière. Un cas assez difficile dû à la présence d'objets très minces dans la scène (les poteaux des pancartes ainsi que les branches et le tronc du petit arbre.). Par contre, la sculpture (l'Isolateur) a été bien "découpée". (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

(a)

(b)

(c)

FIG. 7.15. **Scène réelle captée dans une séquence vidéo "normale" : L'Isolateur (avec les drapeaux). C'est la même scène qu'avec le "rideau" virtuel. Les drapeaux sont situés en bordure de la rue arrière, du côté du bâtiment. (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence). Le vidéo complet est disponible à l'adresse : "http ://www.iro.umontreal.ca/∼vision/hollowap/".**

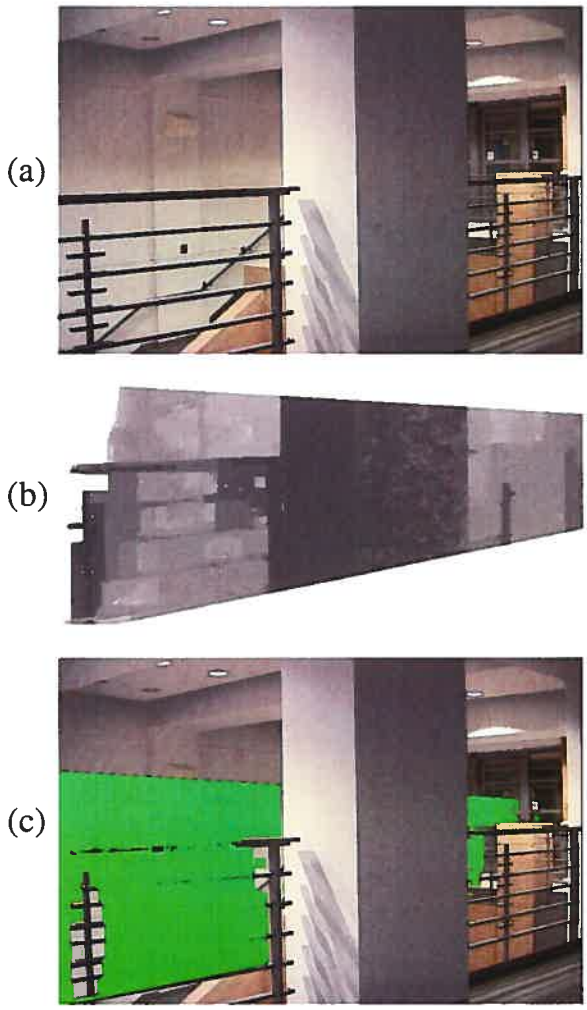



FIG. 7.16. **Scène réelle captée dans une séquence vidéo "normale" : Le corridor. Le "rideau" virtuel passe entre la rampe et le mur arrière. Un cas extrêmement difficile : des bareaux minces de couleur uniforme devant un mur de couleur uniforme. Cette séquence a été la moins bien réussie de l'ensemble de nos tests. (a) image originale, (b) carte de profondeur finale, (c) image finale (réalité augmentée). Nombre d'images utilisées : 51 (25 à droite, 25 à gauche et l'image de référence).**

Chapitre 8

# CONCLUSION

La réalité augmentée est un sujet qui reçoit actuellement beaucoup d'attention. On lui trouve de plus en plus d'applications pratiques. Ce n'est plus un simple sujet de recherche. C'est un domaine de plus en plus en demande dans l'industrie, que ce soit pour faire des études d'impact visuel en architecture ou pour créer des effets spéciaux pour l'industrie du cinéma. C'est en fait l'industrie du cinéma qui dépend le plus de ces produits. D'ailleurs, toutes les compagnies qui créent des logiciels ayant des applications en réalité augmentée (*RealViz, 2d3, SoftImage, Discreet,* etc.) spécialisent leurs produits en fonction de l'industrie du cinéma. Malheureusement, jusqu'à maintenant, toutes les séquences de réalité augmentée qui ont été créées dans un but commercial l'ont été par une méthode demandant une forte interactivité humaine. La plupart du temps, c'est la méthode du "masque" qui est utilisée. Il n'y a pas encore de méthode automatique qui est actuellement utilisée en industrie. Les méthodes automatiques permettraient pourtant d'économiser beaucoup de temps et d'argent dans la fabrication de ces séquences de réalité augmentée. Pourquoi donc ne sont-elles pas utilisées ? Parce que les résultats obtenus par les méthodes automatiques ne sont pas d'une assez bonne qualité.

La méthode qui a été présentée dans ce mémoire est un pas dans la direction de l'automatisation de la réalité augmentée. Nous ne prétendrons pas que notre méthode peut remplacer, dès maintenant, les autres méthodes non-automatiques actuellement utilisées en industrie. Les résultats obtenus sont très bons mais, comme on le constate, ils n'ont pas encore la qualité que l'on retrouve dans un film commercial. En fait, quand nous avons fait nos premiers tests avec la technique de stéréoscopie classique, les résultats étaient bien pires encore. C'est la qualité médiocre de ces pre-

miers résultats qui nous a amenés à développer deux nouvelles techniques : la technique "max-gauche-droite" et la technique "consistance de profondeur". Ces deux techniques, qui constituent la contribution majeure de ce mémoire, nous ont permis enfin d'obtenir des résultats respectables. La technique "max-gauche-droite" nous a permis de traiter les problèmes liés aux demi-occlusions. Quant à la technique "consistance de profondeur", elle offre une solution à l'un des plus grands problèmes de la stéréoscopie dense : le choix du lissage. Ces deux techniques ne sont donc pas limitées à notre méthode de réalité augmentée. Elles peuvent très bien être utilisées dans n'importe quelles autres applications qui impliquent la fabrication d'une carte de profondeur.

Mais évidemment, il reste encore quelques points à améliorer dans notre méthode avant d'obtenir le résultat idéal.

La calibration est quelque chose qu'il faudrait regarder de près dans les travaux futurs. Le fait que les résultats obtenus avec les séquences "normales" sont vraiment de moins bonne qualité que ceux obtenus avec les images rectifiées tend à désigner ce coupable. Si la calibration était parfaite, on obtiendrait d'aussi bons résultats avec les images "normales" qu'avec les images rectifiées. D'ailleurs, on peut voir, quand le logiciel MatchMover a terminé son suivi de points ("feature tracking"), qu'il a fait quelques erreurs. Il y a toujours quelques points saillants ("feature points") qu'il a suivis et qu'il juge bons, alors que concrètement on voit bien que ces points ne demeurent pas "accrochés" à quelque chose dans l'image. De là à dire que ces points, dont se sert MatchMover puisqu'il les a jugés bons, viennent fausser la calibration, il n'y a qu'un pas. MatchMover n'est pourtant pas un mauvais logiciel. Seulement, il n'a pas été conçu pour garantir le niveau de précision nécessaire à la fabrication d'une carte de profondeur en stéréo dense. Lorsque ce type de logiciel est utilisé en industrie, un opérateur "aide" à extraire les trajectoires. Dans nos expériences, on s'est limité aux trajectoires non-retouchées.

L'autre point majeur à améliorer est la vitesse d'exécution de l'algorithme de flot maximum. Le fait de faire du fenêtrage ("clipping") autour de l'objet virtuel et de fabriquer les différentes cartes de profondeur en parallèle, aide beaucoup à réduire le temps d'exécution à un niveau acceptable. Mais cela reste encore relativement lent. La résolution du flot maximum est vraiment ce qui prend le plus de temps dans toute la méthode, et si on veut accélérer le traitement de cette méthode, c'est par là qu'il faut commencer.

Je vais conclure ce mémoire en faisant le vœu que cette méthode automatique pour faire de la réalité augmentée, et surtout les deux techniques que nous avons développées, soit "max-gauche-droite" et "consistance de profondeur", puissent inspirer d'autres recherches en vision par ordinateur, qui feront un jour disparaître la frontière entre le réel et le virtuel.

# RÉFÉRENCES


[1] M. Armstrong, A. Zisserman, et R. I. Hartley. Self-calibration from image triplets. Dans *Proc. European Conference on Computer Vision*, pages 3–16, 1996.

[2] R. T. Azuma. A survey of augmented reality. *Presence : Teleoperators and Virtual Environments*, 6(4) :355–385, 1997.

[3] S. Baker, R. Szeliski, et P. Anandan. A layered approach to stereo reconstruction. Dans *Conference on Computer Vision and Pattern Recognition*, pages 434–441, Santa Barbara, Californie, 1998.

[4] S. Birchfield et C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. Dans *Proc. Int. Conference on Computer Vision*, pages 1073–1080, 1998.

[5] V. Chvátal. *Linear Programming.* W. H. Freeman and Company, New York, NY, 1983.

[6] T. H. Cormen, C. E. Leiserson, et R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill, New York, 1990.

[7] G. Egnal et R. Wildes. Detecting binocular half-occlusions : Empirical comparisons of four approaches. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 466–473, 2000.

[8] O. Faugeras. De la géométrie au calcul variationnel : théorie et applications de la vision tridimensionnelle. Dans *Actes du 11ème Congrès de Reconnaissance des Formes et Intelligence Artificielle (RFIA'98)*, pages 15–34, Clermont-Ferrand, 1998.

[9] G. Fielding et M. Kam. Weighted matchings for dense stereo correspondence. *Pattern Recognition*, 33(9) :1511–1524, 2000.

[10] James D. Foley, Andries van Dam, Steven K. Feiner, et John F. Hughes. *Com-*

*puter Graphics, Principles ans Pratice (2nd edition in C).* Addison Wesley, Reading, Massachusetts, 1996.

[11] P. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. *12th Int. Joint Conf. Artificial Intelligence*, 91 :1292–1298, 1991.

[12] A. Fusiello, V. Roberto, et E. Trucco. Efficient stereo with multiple windowing. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 858–863, Puerto Rico, 1997.

[13] Bernard Grob. *Télévision et vidéo.* McGraw Hill, Montréal, Québec, 1986.

[14] Y. Huang et H. Niemann. Layers-based image segmentation incorporating motion estimation with static segmentation. Dans *Irish Machine Vision and Image Processing'2000*, Belfast, Northern Ireland, 2000.

[15] Bill Jacob. *Linear Functions and Matrix Theory.* Springer-Verlag, New York, NY, 1995.

[16] Jae S. Lim. *Two dimensional signal and image processing.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[17] M. Lin et C. Tomasi. *Surfaces with Occlusions from Layered Stereo.* PhD thesis, Stanford University, 2002.

[18] K. Mühlmann, D. Maier, J. Hesser, et R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 30–36, Kauai Marriott, Hawaii, 2001.

[19] J. Mulligan et K. Daniilidis. Trinocular stereo for non-parallel configurations. Dans *Proc. of Int. Conf. on Pattern Recognition*, volume 1, pages 567–570, 2000.

[20] M. Pollefeys, R. Koch, et L. J. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. Dans *Proc. Int. Conference on Computer Vision*, pages 90–95, 1998.

[21] S. Reeves et R. Mersereau. Blur identification by the method of generalized cross-validation. *IEEE Trans. Image Processing*, 1 :301–311, 1992.

[22] S. Roy et I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. Dans *Proc. Int. Conference on Computer Vision*, pages 492–499, Bombay, Inde, 1998.

[23] S. Roy, J. Meunier, et I. J. Cox. Cylindrical rectification to minimize epipolar distortion. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 393–399, San Juan, Puerto Rico, 1997.

[24] Sébastien Roy. Notes de cours : Vision tridimensionelle (ift6145). *http ://www2.iro.umontreal.ca/~roys/ift6145/diapos/stereo1et2.html/*, 2003.

[25] John C. Russ. *The image processing handbook, 2nd edition.* CRC Press, Boca Raton, Floride, 1995.

[26] Daniel Scharstein et Richard Szeliski. Middlebury college stereo vision research page. *http ://www.middlebury.edu/stereo/.*

[27] J. Sun, H. I. Shum, et N. N. Zheng. Stereo matching using belief propagation. Dans *Proc. European Conference on Computer Vision*, volume 2, pages 510–524, 2002.

[28] N. M. Thalmann et D. Thalmann. Animating virtual actors in real environments. *ACCMS'97*, 5(2) :113–125, 1997.

[29] B. Triggs. Autocalibration and the absolute quadric. Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[30] Emanuele Trucco et Alessandro Verri. *Introductory techniques for 3-D computer vision.* Prentice Hall, Upper Saddle River, New Jersey, 1998.

[31] Alan Watt et Fabio Policarpo. *The Computer Image.* Addison Wesley, Harlow, England, 1998.

[32] Arthur R. Weeks Jr. *Fundamentals of electronic image processing.* SPIE and

IEEE Press, Bellingham, Washington (SPIE), Piscataway, New Jersey (IEEE Press), 1996.

[33] A. Zisserman, A. W. Fitzgibbon, et G. Cross. VHS to VRML : 3D graphical models from video sequences. Dans *ICMCS, Vol. 1*, pages 51–57, 1999.

[34] C. Zitnick et T. Kanade. Cooperative algorithm for stereo matching and occlusion detection. Rapport Technique CMU-RI-TR-99-35, Robotics Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, 1999.

Annexe A

# VERSIONS DES LOGICIELS UTILISÉS

---

Logiciel de calibration MatchMover

Nom du produit : MatchMover 2 Professional

Version : 2.2

Nom de la compagnie : RealViz

Logiciel de modélisation 3D XSI

Nom du produit : XSI

Version : 3.0.1

Nom de la compagnie : SoftImage (Avid Technology, Inc.)

Logiciel de visualisation 3D VTK

Nom du produit : The Visualization Tool Kit (VTK)

Version : 3.2

Nom de la compagnie : Kitware

Logiciel de lecture vidéo MPlayer

Nom du produit : MPlayer

Version : 0.90

Nom de la compagnie : Aucune - logiciel libre

Distribution : Selon les termes de la GNU General Public License Version 2.