Université de Montréal

Utilisation de miroirs dans un système de reconstruction interactif

par

Emric Epstein

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

Décembre 2004

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Utilisation de miroirs dans un système de reconstruction interactif

présenté par

Emric Epstein

a été évalué par un jury composé des personnes suivantes :

Président:	Sébastien Roy
Directeur de recherche:	Pierre Poulin
Membre:	Jean Meunier

Sommaire

Ce mémoire décrit notre intégration de miroirs planaires à l'intérieur d'un système de lumière structurée afin d'ajouter de nouveaux points de vue virtuels à la fois pour la caméra et pour le projecteur. Notre système de reconstruction interactif est flexible et efficace, et il permet de déplacer l'objet sur son piédestal, la caméra et les miroirs grâce à des outils automatiques de calibration et de poursuite. Le piédestal motorisé permet aussi une reconstruction entièrement automatique.

Nous développons de nouveaux motifs de lumière structurée robustes aux réflexions miroir. Nous décrivons comment un rendu OpenGL exploitant l'accélération matérielle des cartes graphiques permet d'éviter les conflits entre les motifs de lumière projetés et observés sur les miroirs, incluant les interréflexions multiples. Nous présentons un système d'optimisation par rendu OpenGL pour guider l'usager sur la disposition des composantes matérielles du système et illustrons le concept à l'aide d'une optimisation sur le placement contraint de deux miroirs. Nous concluons enfin en analysant nos résultats et en proposant plusieurs pistes d'améliorations futures.

Mots clefs :

Reconstruction 3D, reconstruction automatique et interactive, lumière structurée, réflexion miroir, code des frontières de bandes, calibration de projecteur, calibration de caméra, estimation de pose, masque, accélération graphique matérielle, représentation par points.

Abstract

This thesis describes our integration of planar mirrors within a structured light system, providing virtual view points for both the camera and the projector. Our interactive reconstruction system is flexible and efficient, and it allows the user to move the object on its stand, the camera, and the mirrors, thanks to automatic tools for calibration and tracking. An entirely automatic reconstruction is also provided with our motorized object stand.

We develop new structured light patterns that are robust to mirror reflections. We describe how hardware supported OpenGL-based rendering avoids conflicts between the projected and the captured light patterns, even under multiple mirror reflections. We present an optimization system, also using OpenGL-based rendering, to guide the user in configuring the reconstruction devices, and illustrate this concept by placing two constrained mirrors. We conclude by analyzing our results and by proposing several improvements.

Keywords :

3D reconstruction, automatic and interactive reconstruction, structured light, mirror reflection, stripe boundary code, projector and camera calibration, pose estimation, OpenGL masks, graphics hardware, point-based representation.

Table des matières

Remerciements **xv**

1 Introduction **1**

2 État de l'art **6**

2.1 Techniques de reconstruction 7

2.1.1 Techniques non-optiques 7

2.1.2 Techniques optiques passives 7

2.1.3 Techniques optiques actives 9

2.1.4 Techniques combinées 9

2.2 Miroirs 10

3 Lumière structurée **11**

3.1 Description 11

3.2 Encodage temporel 14

3.2.1 Encodage binaire 14

3.2.2 Encodage de Gray 15

3.2.3 Encodage des frontières 17

3.2.4 Notre implémentation 19

3.3 Synchronisation 23

3.4 Limitations 27

4 Miroirs en lumière structurée **30**

4.1 Calibration du projecteur 30

4.2 Calibration de la caméra 31

4.3 Pose d'un miroir 34

4.4 Boîte englobante 37
4.5 Miroir 37
4.6 Motifs de lumière structurée 40

5 Système de reconstruction interactif 44
5.1 Piédestal 44
5.1.1 Initialisation 44
5.1.2 Suivi temporel (*tracking*) 46
5.1.3 $LEGO^{TM}$ 49
5.2 Utilisation interactive de miroirs 51
5.2.1 Philosophie 51
5.2.2 Initialisation 52
5.2.3 Suivi temporel 3D 52
5.3 Intégration et post-traitement du modèle final 54
5.4 Affichage du modèle 57
5.5 Système d'optimisation 58

6 Résultats 62
6.1 Implémentation 62
6.2 Trois miroirs fixes et objet fixe 64
6.3 Trois miroirs fixes et objet sous multiples poses 65
6.4 Miroir déplacé interactivement et objet fixe 66
6.5 Piédestal automatique sans miroir 67
6.6 Piédestal automatique avec miroir 68

7 Conclusion 74

8 Glossaire 79

Bibliographie 81

Liste des tableaux

3.1 Tableau des codes de Gray pour les nombres de 3 bits. 17

6.1 Temps approximatifs des étapes d'initialisation du système. 63

6.2 Statistiques pour la statue de la tête. 65

Table des figures

1.1 Gauche : Système de reconstruction traditionnel par lumière structurée. Droite : Système de reconstruction proposé intégrant des miroirs ; les interactions de l'usager sont indiquées par un symbole de "personne". . 2

1.2 Notre système de reconstruction interactif avec miroirs. 4

3.1 (a) Schéma d'une caméra avec projection perspective : un point 3D de l'espace P se projette sur le plan image dans le pixel p_{cam} ; à ce pixel correspond un rayon R_{cam} partant du centre de projection de la caméra, passant par le pixel p_{cam} sur le plan image et se prolongeant dans l'espace en une ligne qui traverse le point P. (b) Schéma similaire pour le projecteur ; il est à noter que le plan image du projecteur n'est pas centré sur son axe optique. 12

3.2 (a) Reconstruction point par point à l'aide d'un émetteur laser ; (b) reconstruction ligne par ligne en utilisant une ligne laser. Cette solution est couramment utilisée dans les systèmes de reconstruction commerciaux. . 13

3.3 (a) Encodage binaire : chaque ligne horizontale représente une image de motifs projetée et chaque colonne est une séquence (code) d'illumination temporelle unique ; on peut observer la présence de la frontière du milieu (entre les codes 7 et 8) dans toutes les images du code binaire ; (b) encodage de Gray, il est à noter que chaque frontière n'apparaît temporellement qu'une seule fois. Aussi, la largeur des bandes est inversement proportionnelle au nombre de bandes. 15

3.4 Graphe tiré de [HHR01]. Les noeuds représentent les codes binaires des bandes de lumière structurée et les arêtes représentent les frontières valides entre ces bandes. 18

3.5 (a) Les bandes de lumière structurée qui sont projetées avec une largeur en pixels d_{proj}, sont réfléchies par l'objet avec une largeur inconnue D et la caméra capte ces bandes avec une largeur inconnue d_{cam} en pixels. (b) Nous inversons le processus en approximant l'objet par un plan Π_{obj} pour estimer la largeur des bandes dans le projecteur à partir de la largeur minimale des bandes d_{cam} que l'on veut observer dans la caméra. 21

3.6 (a) Carte de correspondances de la caméra après une itération de reconstruction ; (b) carte de correspondances complétée après 7 itérations de reconstruction. La résolution utilisée dans cet exemple est de 150 codes de frontières de bandes. On peut noter que la présence d'interréflexions dans notre environnement de reconstruction empêche l'identification des codes dans le coin de celui-ci. Les cartes de correspondances ont été traitées en noir et blanc pour apparaître plus clairement lors de l'impression. Les cartes originales sont en couleurs, et chaque composante de couleur (RGB) encode une information. L'espace en haut à gauche a été réservé pour le code de synchronisation (Section 3.3). 22

3.7 (a) Environnement de reconstruction avec le code de synchronisation déformé pour apparaître dans le coin supérieur gauche de la caméra calibrée ; le code a été agrandi en bas à droite de l'image et les zones du code ont été mises en évidence pour faciliter la visualisation ; (b) environnement photographié d'un autre point de vue, on peut y observer la déformation du code de synchronisation. Les motifs et le code de synchronisation projetés sont différents dans les deux images. 24

4.1 (a) Modèle 3D de l'environnement de reconstruction ; (b) calibration sous-pixel du projecteur ; (c) calibration manuelle de la caméra avec l'outil *fish-eye* ; (d) calibration automatique de la caméra à l'aide de points de couleurs projetés ; (e) environnement de reconstruction synthétique projeté sur l'environnement réel par le projecteur ; (f) environnement de reconstruction synthétique affiché par-dessus l'image de la caméra. Les deux dernières images (e, f) permettent de valider visuellement la précision des matrices de projection retrouvées respectivement pour le projecteur et pour la caméra. Dans les deux cas, la matrice de calibration retrouvée est utilisée pour afficher le modèle synthétique du bon point de vue. 33

4.2 (a) Sélection des marqueurs de couleur pour définir le contour du miroir ; (b) carte des correspondances avec identification des contours du miroir ; (c) points 3D reconstruits et géométrie du miroir identifiée en 3D (ligne rouge). Comme nous ne pouvons faire d'hypothèse sur l'emplacement du miroir dans l'espace avant de connaître sa pose, nous projetons les motifs de lumière structurée partout où le projecteur peut émettre. Les points reconstruits ailleurs que sur le contour du miroir ne sont présents dans la figure (c) que pour des fins d'illustration. 34

4.3 Illustration 2D de l'estimation d'un plan moyen à partir d'un nuage de points. (a) Pour un point donné P, les points voisins dans un rayon r sont utilisés pour calculer le PCA ; (b) en 2D, on obtient une seule orientation principale T_p pour ce groupe de points ; (c) on peut donc calculer le vecteur N perpendiculaire à T_p et finalement retrouver l'équation du plan tangent à la surface. 36

4.4 (a) Image observée par la caméra ; (b) carte des zones du point de vue de la caméra ; (c) carte des zones du point de vue du projecteur. Les zones de conflit sont illustrées par un motif d'échiquier. 39

4.5 (a) Projection uniquement sur l'objet ; (b) projection uniquement sur le miroir de gauche ; (c) projection uniquement sur le miroir du milieu ; (d) projection uniquement sur le miroir de droite. Les motifs peuvent être analysés par la caméra simultanément sur l'objet et dans ses réflexions dans les miroirs. 41

4.6 (a) Décodage le long des lignes de pixels horizontales ; (b) décodage le long des lignes de pixels verticales ; (c) décodage combiné le long des lignes de pixels horizontales et verticales. 43

5.1 Cube RGB selon deux points de vue différents avec les ellipsoïdes englobants qui représentent les couleurs des marqueurs du piédestal, telles que captées par la caméra. 46

5.2 Suivi temporel du piédestal. (a) La boîte englobante reposant sur la base du piédestal est illuminée ; (b) seulement les marqueurs de couleur suivis temporellement sont illuminés ; (c) autre position suivie du piédestal. . . 48

5.3 (a) Contrôleur RCX pour les *LEGO MINDSTORMS* (image tirée de kauts.mine.nu/rcx.html) ; (b) piédestal déplacé automatiquement et construit à partir des *LEGO MINDSTORMS*. 49

5.4 (a) Numériseur laser à balayage motorisé développé par *Cyberware* ; (b) tête du numériseur composée d'une source laser, d'une caméra vidéo pour la numérisation 3D, d'une source de lumière à fibre optique et d'une caméra couleur haute définition. 51

5.5 Suivi temporel 3D d'un miroir déplacé manuellement : (a) pose initiale du miroir lors de l'estimation de pose affichée par-dessus l'image caméra et (b) vue synthétique dans notre système ; (c) pose après déplacement manuel du miroir suivi automatiquement en 2D et en 3D par le système et (d) équivalent synthétique. 53

5.6 Traitement du modèle de l'ourson, les images du haut illustrent l'évolution du modèle après chaque étape de filtrage et les images du bas sont des gros plans correspondants. Chaque point est affiché avec un disque de couleur. (a),(b) Modèle après alignement global ; (c),(d) après élimination des points isolés ; (e),(f) après filtrage médian ; (g),(h) après filtrage des couleurs. 56

5.7 Affichage du modèle : (a) modèle affiché sous forme de points après traitement ; (b) modèle affiché en *splats* par *Qsplat* [RLat] ; (c) modèle affiché en *splats* par *PointShop3D* [ZPKG3d]. 58

5.8 (a) Image représentant la couverture des configurations testées pour le placement de deux miroirs carrés de 5×5 pouces en fonction de la couverture qu'ils apportent ; chacun des deux miroirs est déplacé le long du plan de l'environnement contre lequel il repose ; l'axe des abscisses représente le déplacement du miroir de droite et l'axe des ordonnées représente le déplacement du miroir de gauche ; (b) positionnement optimal indiqué par le système d'optimisation ; une couleur spéciale est assignée aux pixels de la boîte englobante observée directement ou dans un miroir. 61

6.1 Photographies de quelques objets que nous avons utilisés pour la reconstruction. On peut observer que le tigre, le globe terrestre et les yeux du chiot réfléchissent la lumière de manière spéculaire. 64

6.2 Reconstruction statique de la tête : tous les éléments du système de reconstruction restent fixes, seul le nombre de miroirs utilisés change. Gauche : aucun miroir. Centre : un miroir ; Droite : trois miroirs. 64

6.3 Reconstruction de la tête avec trois miroirs. L'objet a été déplacé manuellement avec le piédestal et reconstruit selon quatre poses différentes. 65

6.4 Ces images montrent les résultats de reconstruction d'un objet fixe avec un miroir déplacé interactivement. Temps total de reconstruction : 10 minutes ; temps total de post-traitement : 5 minutes ; définition : 56,000 points. 66

6.5 Reconstruction de l'ourson sur le piédestal automatique. Aucun miroir n'est utilisé, seul l'objet est tourné automatiquement. Temps total de reconstruction : 4 minutes ; temps total de post-traitement : 8 minutes ; définition : 265,000 points. L'image en haut à gauche est une photo de l'ourson, fournie à titre comparatif et prise avec un éclairage différent. . 68

6.6 Le tigre reconstruit à l'aide du piédestal automatique illustre la reconstruction en présence de texture colorée. Temps total de reconstruction : 8 minutes ; temps total de post-traitement : 2 minutes ; définition : 152,000 points. 69

6.7 Le globe terrestre, reconstruit à l'aide du piédestal automatique illustre la reconstruction en présence de texture fine complexe. On peut observer que la forme sphérique du globe est assez bien préservée. Temps total de reconstruction : 5 minutes ; temps total de post-traitement : 3 minutes ; définition : 228,000 points. 70

6.8 Image de la caméra illustrant les conditions de reconstruction lors de l'utilisation du miroir avec le piédestal automatique. On peut observer que la caméra a une vue en plongée sur l'objet pour permettre le suivi précis des marqueurs de couleur du piédestal. Le miroir est placé pour offrir une vue en contre-plongée. 71

6.9 Pomme en plastique reconstruite à l'aide du piédestal automatique et d'un miroir en un seul tour complet. Temps total de reconstruction : 5 minutes ; temps total de post-traitement : 2 minutes ; définition : 50,000 points. 71

6.10 Comparaison de la reconstruction du modèle du chiot à l'aide du piédestal automatique, avec et sans miroir. Les conditions de reconstruction sont exactement les mêmes pour les deux modèles, seul le miroir a été désactivé pour le modèle illustré en haut. Ligne du haut : aucun miroir ; temps total de reconstruction : 5 minutes ; temps total de post-traitement : 5 minutes ; définition : 80,000 points. Ligne du bas : un miroir ; temps total de reconstruction : 8 minutes ; temps total de post-traitement : 7 minutes ; définition : 142,000 points. 72

6.11 Modèle du chiot reconstruit avec le piédestal automatique et un miroir. On peut observer que la partie inférieure museau est reconstruite malgré que cette surface pointe vers le sol. Temps total de reconstruction : 12 minutes ; temps total de post-traitement : 10 minutes ; définition : 307,000 points. 73

Remerciements

Lors des 2 dernières années, plusieurs personnes formidables, que je tiens à remercier, ont contribué à me permettre d'avancer dans les études et dans la vie.

La personne que je veux remercier avant tous est Pierre Poulin, qui prouve que l'on peut être en même temps un excellent professeur/chercheur et une personne humaine d'une qualité inégalable. Je pense sincèrement qu'il aurait été impossible pour moi de réaliser ce travail sans son aide précieuse. Je pense aussi qu'il aurait été amusant de conserver une copie de ce mémoire avec toutes les fautes originales pour illustrer la patience de Pierre.

Je tiens aussi à remercier mes parents, Michèle et Denis, qui me soutiennent depuis toujours et qui ont sacrifié beaucoup pour que je puisse venir étudier au Canada dans ce domaine qui me passionne. Merci particulièrement à Denis pour m'avoir fait découvrir très jeune le monde fantastique des ordinateurs et de l'infographie, et à Michèle qui m'a toujours incité à avoir une vie sociale et à aller à la plage.

Un grand merci à Martin, mon meilleur ami et partenaire d'études depuis maintenant 5 ans et désormais partenaire de travail. Merci à Yanesis, qui est un soleil dans ma vie peu importe qu'il fasse beau, froid ou qu'il neige.

Merci également à tous les membres du LIGUM, avec qui j'ai partagé ces 3 dernières années, et qui m'ont apporté beaucoup. Je pense particulièrement à Mathieu Ouimet et François Duranleau pour leurs expertises en programmation ainsi qu'à tous les autres pour leur soutien et leur présence agréable.

J'aimerais remercier les membres du jury pour leurs critiques constructives sur ce mémoire. Merci finalement au FCAR équipe, au MITACS et au CRSNG pour leur appui financier.

Chapitre 1

Introduction

Au cours des 20 dernières années, la quête du réalisme a été une voie prépondérante de la recherche en infographie. Soutenus par la puissance de calcul sans cesse croissante des ordinateurs, la complexité des applications 3D et l'interactivité des jeux vidéo, les besoins d'objets synthétiques réalistes et complexes sont en constante augmentation. L'industrie du cinéma est un des chefs de file dans cette quête incessante du réalisme. Chaque année les nouveaux films d'animation 3D tels *Final Fantasy*, *Shrek* et *Finding Nemo* poussent la qualité des modèles 3D, de l'animation, de la simulation et du rendu vers de nouvelles limites. Parallèlement, les grandes productions qui intègrent des effets spéciaux réalistes se doivent de pousser le réalisme au point de cacher la présence de traitements informatiques. Des scènes calculées synthétiquement sont intégrées dans les films comme *Terminator 2*, *Lord of the Rings* et *The Day after Tomorrow*, où la plupart des spectateurs ne peuvent même pas identifier la majorité des scènes retraitées ou rendues complètement synthétiquement.

La manière traditionnelle en infographie de créer ces objets fait appel aux logiciels de modélisation 3D tels *Softimage|XSI* [Sof], *Maya* de *Alias|Wavefront* [Ali] et *3D Studio MAX* de *Autodesk* [Aut]. Malgré la puissance de ces logiciels, ils nécessitent tous un apport considérable des artistes professionnels pour créer des objets complexes.

Plutôt que de construire manuellement ces objets synthétiques avec des programmes de modélisation 3D, le domaine de la vision par ordinateur propose des techniques et des algorithmes de reconstruction 3D, qui sont commercialisés et utilisés par l'industrie du cinéma et des jeux vidéo. La numérisation 3D d'objets réels est donc un champ de recherche très actif depuis quelques dizaines d'années en vision par ordinateur [CM99,

SG99]. De plus depuis récemment, ce champ de recherche s'est étendu au domaine de l'infographie [BR00, LPC+00, RHHL02, BR02].

Parmi toutes les techniques de reconstruction 3D, la lumière structurée est une des plus populaires [RCM+01]. Une séquence temporelle de motifs de lumière (bandes d'illumination généralement verticales) est émise par un projecteur vidéo sur la surface d'un objet et les images résultantes sont acquises par une caméra. Chaque élément de surface illuminé par le projecteur et visible par la caméra est associé à une séquence d'états (illuminé/non-illuminé) qui identifie une bande unique d'illumination du projecteur. Cette séquence d'états est appelée "code d'illumination". La séparation entre deux bandes d'illumination dans le projecteur définit un plan 3D dans l'espace et ce plan est intersecté par un rayon issu du pixel de la caméra. De cette intersection résulte un point 3D. Les données spatiales (plan 3D et rayon) nécessitent un système de coordonnées commun entre le projecteur et la caméra, qui est obtenu à partir de la calibration au préalable de ces deux appareils. Ce processus est illustré à la Figure 1.1 (gauche).

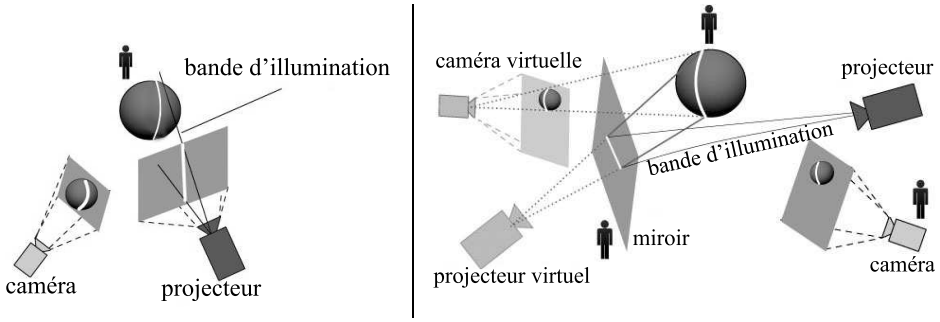


FIG. 1.1 – Gauche : Système de reconstruction traditionnel par lumière structurée. Droite : Système de reconstruction proposé intégrant des miroirs ; les interactions de l'usager sont indiquées par un symbole de "personne".

La reconstruction par lumière structurée a pour avantages la simplicité, l'efficacité, la précision et la flexibilité. Cependant, comme pour la plupart des techniques de reconstruction, le problème des occlusions se pose : seules les parties des objets qui sont à la fois illuminées par le projecteur et visibles par la caméra peuvent être reconstruites. Pour compléter la reconstruction, plusieurs configurations caméra/projecteur/objet sont donc nécessaires. Ce problème pourrait être réglé en utilisant simultanément plusieurs systèmes caméra/projecteur calibrés, mais le coût du système deviendrait rapidement prohibitif. Les solutions les plus communément utilisées consistent à déplacer manuel-

lement ou à l'aide d'un bras robotisé, soit l'objet, soit le système de reconstruction.

En associant l'utilisation de la cohérence temporelle et d'un algorithme d'alignement ICP (*Iterative Closest Point*), il est possible de déplacer interactivement un objet devant un système de lumière structurée et d'aligner ensemble en 3D les résultats des itérations successives. Ce concept a été démontré par Rusinkiewicz *et al.* [RHHL02] dans leur système de reconstruction en temps réel. Cette solution n'est toutefois pas applicable lorsque l'objet à reconstruire est trop lourd ou lorsque celui-ci est fixé dans un environnement contraint.

Une solution moins automatique mais très populaire, consiste à reconstruire l'objet selon plusieurs points de vue, puis à aligner manuellement les points reconstruits de manière suffisamment précise pour qu'un algorithme ICP puisse converger vers un alignement satisfaisant. Malheureusement, cette solution demande beaucoup d'interaction de la part de l'usager, car le placement manuel selon les six degrés de liberté (3 translations et 3 rotations) des groupes de points reconstruits peut être très fastidieux.

La solution appliquée dans le projet *The Digital Michelangelo* [LPC+00] consiste à fixer le système de reconstruction sur un bras robotisé très précis permettant de connaître le déplacement et l'orientation du bras à chaque itération de la reconstruction. L'alignement final utilise encore une fois un algorithme ICP. La solution du bras robotisé possède cependant plusieurs désavantages que nous détaillerons plus tard dans la Section 5.2.1.

Déplacer manuellement le point de vue d'une caméra et la calibrer automatiquement en utilisant le projecteur [GPEP04] permet d'obtenir plus de points 3D. Cependant, si le projecteur et l'objet restent respectivement fixes, les éléments de surface illuminés par le projecteur demeurent les mêmes. Seulement des éléments qui deviennent visibles par la caméra mobile peuvent être reconstruits.

Ainsi, déplacer le projecteur permettrait de reconstruire selon d'autres angles d'illumination. Mais malheureusement, les projecteurs sont habituellement plus encombrants que les caméras. De plus, il est plus difficile de calibrer automatiquement un projecteur car celui-ci ne peut généralement pas acquérir d'images. Fixer une caméra au projecteur [RBB+03] permet de résoudre partiellement ce problème, mais cette solution est analogue à déplacer l'objet devant le système de reconstruction.

L'utilisation de miroirs offre une alternative au déplacement de la caméra et du pro-

jecteur. Capté par une caméra, un miroir planaire produit une image telle qu'observée d'un point de vue virtuel. Cette position correspond au point de vue de la caméra réfléchi par le plan de support du miroir. Similairement, la lumière émise par un projecteur et réfléchie par un miroir correspond à la lumière qu'émettrait un projecteur dont le "point de vue" serait réfléchi par le miroir.

Dans le contexte de la reconstruction par lumière structurée, un miroir ajoute (Figure 1.1 (droite) et Figure 1.2) :

- un projecteur virtuel qui émet des motifs sur l'objet, captés directement par la caméra ;
- une caméra virtuelle qui capte les motifs émis directement par le projecteur ;
- une caméra virtuelle qui capte les motifs émis par un projecteur virtuel.

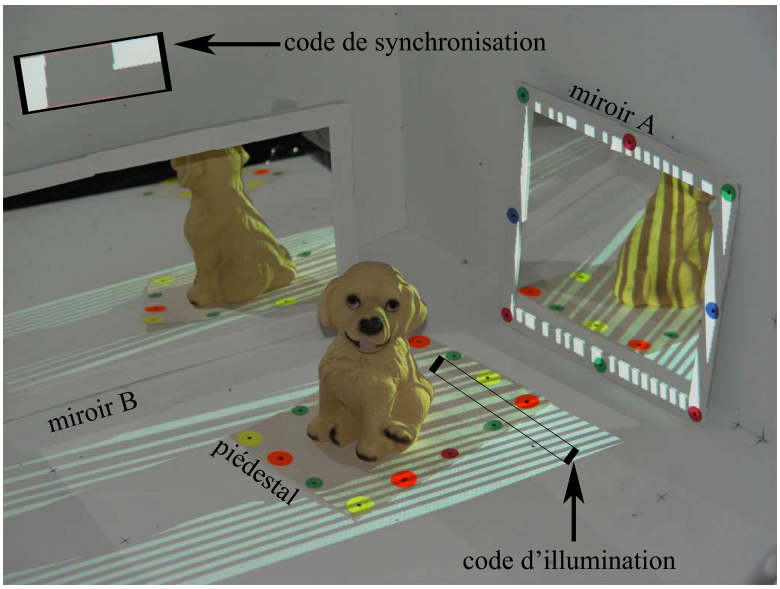


FIG. 1.2 – Notre système de reconstruction interactif avec miroirs.

Les miroirs offrent donc une alternative intéressante dans les situations où l'objet à reconstruire ne peut être déplacé (grande statue ou environnement architectural), et/ou lorsque la caméra et le projecteur doivent rester dans une zone limitée de l'espace. Ils permettent de ne calibrer précisement la caméra et le projecteur qu'une seule fois lors de l'étape d'initialisation. Ainsi, la seule contrainte pour obtenir un système de reconstruction robuste, réside dans l'estimation précise de la pose (géométrie, position et orientation) du miroir dans l'espace.

Nos contributions sur l'intégration des miroirs dans le processus de reconstruction

par lumière structurée ont fait l'objet d'une publication [EGPP04] et sont les suivantes :

- l'estimation précise de la pose de miroirs ;
- le suivi temporel du déplacement de miroirs ;
- le calcul efficace de masques en OpenGL pour identifier les régions valides (sans risque d'ambiguïté) où projeter et analyser l'information pour la reconstruction ;
- la modification du code de lumière structurée de Hall-Holt et Rusinkiewicz [HHR01] permettant l'analyse du code après transformation de réflexion ;
- un système d'optimisation permettant au système de guider l'usager dans le placement des miroirs.

Dans les prochains chapitres, nous présenterons comment exploiter avantageusement les miroirs dans un système de reconstruction interactif. Nous survolerons différentes techniques de reconstruction existantes ainsi que celles faisant état de l'utilisation de miroirs dans des systèmes caméra/projecteur (Chapitre 2). Nous décrirons ensuite les bases de la reconstruction par lumière structurée ainsi que l'implémentation de cette technique dans notre système (Chapitre 3). Par la suite, nous montrerons l'utilité de l'intégration de miroirs dans ce système (Chapitre 4). Enfin, nous décrirons les différents outils et algorithmes permettant l'interaction entre l'usager et le système lors de la reconstruction d'un objet (Chapitre 5). Nous montrerons et commenterons les modèles reconstruits par notre système en indiquant certaines statistiques (Chapitre 6). Finalement nous ferons le bilan des aspects positifs et négatifs de notre système de reconstruction, des contributions que nous avons apportées, ainsi que des perspectives d'améliorations possibles (Chapitre 7).

Chapitre 2

État de l'art

La reconstruction 3D est un champ de recherche très populaire en vision par ordinateur et depuis peu en infographie. Les techniques de reconstruction sont utilisées dans beaucoup de domaines dont, pour n'en nommer que quelques-uns, la médecine, la robotique, l'industrie militaire, l'industrie du cinéma et des jeux vidéo.

Les différentes techniques de reconstruction qui existent peuvent être catégorisées selon leurs particularités. Ainsi on peut différencier les techniques non-optiques qui n'utilisent pas de caméras ni de projecteurs, les techniques optiques passives qui acquièrent de l'information à l'aide de récepteurs optiques (caméras) sans interférer avec la scène, et les techniques optiques actives qui agissent sur la scène à l'aide d'un émetteur (émetteur laser ou projecteur vidéo) pour augmenter la robustesse de l'information acquise par le récepteur (caméra).

Les miroirs ont été étudiés et simulés depuis longtemps en infographie. De plus, certains systèmes caméra/projecteur ont préalablement utilisé les miroirs dans différents contextes, soit pour augmenter virtuellement le nombre d'émetteurs et de récepteurs optiques, ou encore pour contrôler leurs points de vue virtuels.

Dans ce chapitre nous présentons un rapide survol de l'état de l'art sur les techniques de reconstruction 3D et sur l'utilisation des miroirs en vision 3D et en infographie. Le but de ce chapitre n'est pas de faire une revue de littérature exhaustive, mais de familiariser le lecteur avec certaines techniques de reconstruction existantes et les principales recherches en rapport avec ce document.

2.1 Techniques de reconstruction

2.1.1 Techniques non-optiques

Avant l'avènement des domaines de l'infographie et de la vision 3D, beaucoup de techniques ont été développées pour permettre l'analyse ou la reconstruction de l'environnement. Plusieurs de ces techniques se basent sur la télémétrie ou "temps de vol" d'un signal. La télémétrie consiste à mesurer le temps que met un signal pour se propager de l'émetteur à l'environnement sur lequel il "rebondit", puis pour revenir au récepteur. C'est généralement ce principe qui est utilisé dans les sonars et les radars. Le plus souvent les signaux sont émis sous forme d'ondes acoustiques, d'ondes radio ou d'ondes magnétiques.

Une autre technique, utilisée pour la reconstruction d'objets, est le palpage. Généralement cette technique consiste à placer le senseur d'un système de localisation 3D sur la surface de l'objet à reconstruire pour obtenir la position de points de la surface de l'objet. Ensuite, il suffit de lier ces points 3D pour créer des segments de ligne et éventuellement des polygones. Cette technique demande beaucoup de participation de la part de l'usager et semble avoir perdu de son attractivité dans les dernières années.

Une solution traditionnelle en infographie pour créer ou reconstruire des objets et des environnements est de faire appel à des artistes spécialisés qui modélisent manuellement les objets à l'aide de logiciels 3D [Sof, Ali, Aut]. Les modèles sont souvent créés à partir de photos, de dessins ou de sculptures interprétées par l'oeil et le cerveau de l'artiste. Ces logiciels sont aussi utilisés pour traiter, nettoyer, et animer les modèles reconstruits par les numériseurs 3D.

2.1.2 Techniques optiques passives

Le but de la stéréovision est de déterminer la profondeur des pixels qui composent les images provenant de deux caméras ou plus (*n-view stereo*). Les profondeurs sont généralement retrouvées par triangulation en déterminant des correspondances de pixels entre les images [SSZ01]. Les correspondances peuvent être déterminées à l'aide des couleurs ou d'autres caractéristiques comme la présence de coins, de contours, de segmentations, *etc.*

Les techniques de reconstruction passives ne nécessitent pas d'interagir avec la scène

à reconstruire. Cependant plusieurs hypothèses sont nécessaires pour appliquer ces algorithmes et souvent ces mêmes hypothèses définissent les limites des techniques.

Dans certains cas d'ambiguïté (réflexion, réfraction), les algorithmes ne peuvent déterminer correctement les correspondances en pixels des éléments entre deux images. Contrairement au cerveau humain, ils n'ont généralement pas connaissance de la topologie ou du contexte présent dans les images. Dans de tels cas, un usager peut souvent mieux décider des éléments à mettre en correspondance entre les différentes images.

Ainsi, la modélisation interactive [DTM96, POF98, Rea] consiste à faire intervenir l'usager dans le processus de reconstruction en sélectionnant et en mettant en correspondance des éléments dans les images pour créer des points en 3D. Ensuite l'usager peut créer segments de lignes entre ces points et des polygones à partir de ces segments. Le système affiche le modèle reconstruit et l'usager peut décider de modifier interactivement les données pour améliorer le résultat.

Certaines techniques permettent de créer un modèle 3D en se basant sur la cohérence entre les différentes images. La sculpture d'espace (*space carving*) [KS00, GPEP04] consiste à enlever dans une grille 3D tous les éléments de la grille dont la reprojection semble incohérente dans les différentes images. Les éléments éliminés sont ceux dont les couleurs observées sont suffisamment différentes dans plusieurs images. À l'inverse, il est possible de générer un modèle en créant des points aléatoirement dans l'espace et en ne conservant que ceux qui sont cohérents dans les différentes images [PSD$^+$03] selon leurs points de vue respectifs.

L'infographie a aussi apporté plusieurs techniques qui favorisent la qualité de rendu des modèles reconstruits plutôt que la qualité du modèle géométrique sous-jacent. Les techniques comme le *lumigraph* [GGSC96] ou *QuickTime VR* [SCBB02] consistent à acquérir l'information visuelle autour d'un objet puis à la reproduire par interpolation pour de nouveaux points de vue sans nécessiter de géométrie. Le *image-based visual hull* [MBR$^+$00] crée une géométrie approximative basée sur les silhouettes qui permet l'acquisition et le rendu en temps réel. Cette technique a été intégrée et améliorée récemment dans un système d'acquisition, de transmission et d'affichage auto-stéréoscopique nommé 3DTV [MP04]. En consacrant plus de temps à l'acquisition de données et au rendu, il est possible de traiter des modèles complexes et partiellement transparents avec des techniques comme le *opacity hull* [MPN$^+$02].

2.1.3 Techniques optiques actives

Par opposition aux techniques optiques passives, les techniques actives consistent à émettre un signal optique sur l'objet à reconstruire pour faciliter la mise en correspondance de points. Généralement, un émetteur laser ou un projecteur vidéo émet sur la surface de l'objet un motif qui est capté par un récepteur optique (caméra). Ces techniques de reconstruction sont plus précises et les ambiguïtés lors de l'analyse des correspondances sont diminuées de beaucoup. La plupart des systèmes commerciaux de reconstruction sont d'ailleurs basés sur les solutions optiques actives.

Les systèmes de reconstruction basés sur les émetteurs laser [Cyb, Pol, DC01] sont généralement les plus précis mais aussi les plus couteux. Ils sont souvent utilisés par l'industrie du cinéma, de l'aérospatiale [SCBB02] ou dans les projets où une grande précision est nécessaire [LPC+00]. L'avantage des émetteurs laser est la puissance lumineuse qu'ils peuvent projeter sur un objet. Aussi les lasers ont été utilisés pour analyser l'interaction de la lumière à l'intérieur d'objets translucides [GLL+04].

Une solution plus abordable est d'utiliser un projecteur vidéo comme émetteur. Les systèmes dits de lumière structurée [RCM+01] consistent à projeter une ligne illuminée ou un motif encodé temporellement qui est analysé par une caméra pour établir les correspondances. Certains systèmes commerciaux sont ainsi basés sur la lumière structurée [Ins]. Hall-Holt et Rusinkiewicz ont développé un système basé sur la lumière structurée permettant la reconstruction interactive d'un objet en temps réel [HHR01, RHHL02]. L'interaction avec l'usager permet d'aider le processus de reconstruction car le système peut afficher l'état actuel de la reconstruction et ainsi guider l'usager pour placer l'objet dans une configuration où de nouvelles informations seront disponibles.

2.1.4 Techniques combinées

Plusieurs travaux de recherche ont tenté d'intégrer ensemble les techniques actives et passives pour profiter des avantages des deux. Ainsi, des systèmes combinant deux caméras et un projecteur ont été développés pour créer des résultats de référence précis de stéréovision et ainsi permettre de comparer les performances des techniques optiques passives [SS03]. Certains ont étudié comment combiner efficacement les techniques actives et passives [ZCS03, DRR03] pour obtenir des reconstructions plus rapides ou plus

précises.

2.2 Miroirs

Les miroirs synthétiques sont étudiés et utilisés depuis longtemps en infographie. Plusieurs techniques ont été développées pour permettre le rendu efficace et de qualité d'objets réflectifs et de miroirs (planaires ou non) dans les techniques de rendu par lancer de rayons (*ray tracing*) [MH92, JC98] et par *Z-Buffer* [Ope, RTJ94]. Certaines techniques de rendu de surfaces miroirs par tracé de lignes sont intégrées dans les jeux vidéo depuis plusieurs années (*Duke Nukem 3D*, *Need for Speed*, *etc.*).

En vision par ordinateur, les miroirs ont été étudiés dans le cadre de la stéréovision [BN99, GN01]. Ainsi il est possible de n'utiliser qu'une seule caméra augmentée de deux miroirs pour obtenir deux points de vue pour la stéréovision passive. Cela permet de réduire les problèmes de mises en correspondance induits par les différences de réglages entre deux caméras.

Les systèmes caméra/projecteur ont été associés aux miroirs planaires dans les recherches d'IBM [Pin01] pour créer un projecteur "mobile" permettant de projeter sur différentes surfaces planaires. À notre connaissance ce système permet de projeter sur différentes surfaces planaires et la correction de la déformation de projection est calibrée manuellement par l'usager pour chaque surface de projection.

Récemment un mur de 12 miroirs a été utilisé en conjonction avec une caméra et un projecteur pour le prototypage d'un mur de caméras et de projecteurs [LCV+04]. Ce prototype a permis d'étudier plusieurs applications telles l'imagerie homofocale et la projection homofocale avant de réaliser un véritable mur de caméras et de projecteurs pour poursuivre leurs recherches.

Chapitre 3

Lumière structurée

3.1 Description

La reconstruction par lumière structurée est, comme mentionné plus tôt, une technique de reconstruction active utilisant à la fois un projecteur et une caméra. Avant tout, il faut procéder à la calibration de ces deux périphériques (paramètres intrinsèques et paramètres extrinsèques) pour pouvoir les situer dans un même monde de référence. La technique de calibration utilisée dans notre système sera décrite plus en détail dans la Section 4.1.

Cette calibration permet de calculer une matrice de projection. Nous retrouvons donc une matrice $\mathbf{M}_{cam}$ pour la caméra et une matrice $\mathbf{M}_{proj}$ pour le projecteur. Ces deux matrices permettent d'obtenir, pour tout point P de l'espace 3D, le pixel correspondant (respectivement p_{cam} dans la caméra et p_{proj} dans le projecteur) par simple multiplication matricielle (Figure 3.1 (a)).

$$p_{cam} = \mathbf{M}_{cam} P$$
$$p_{proj} = \mathbf{M}_{proj} P.$$

Il est possible de calculer les matrices inverses de projection $\mathbf{M}_{cam}^{-1}$ et $\mathbf{M}_{proj}^{-1}$. Ces matrices permettent de retrouver pour chaque pixel de la caméra (ou du projecteur) la demi-droite dans l'espace qui lui correspond, dont l'origine est le centre de projection et la direction passe par le centre du pixel.

L'avantage d'utiliser un projecteur dans un système de reconstruction est de pouvoir modifier l'illumination de la scène grâce à celui-ci. Il faut donc au cours de la reconstruction pouvoir différencier une surface illuminée par le projecteur d'une surface qui ne

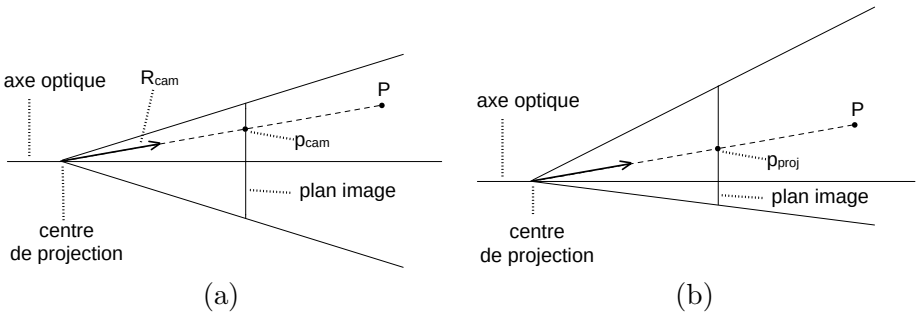
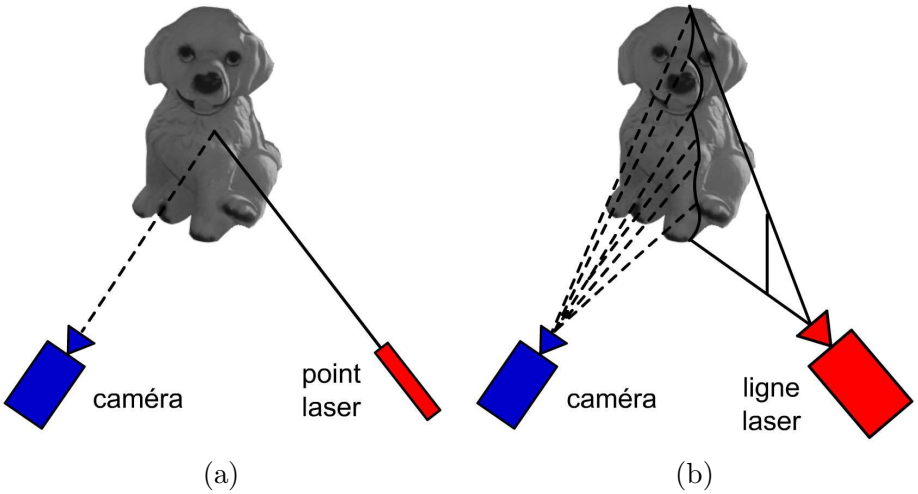


FIG. 3.1 – (a) Schéma d'une caméra avec projection perspective : un point 3D de l'espace P se projette sur le plan image dans le pixel p_{cam} ; à ce pixel correspond un rayon R_{cam} partant du centre de projection de la caméra, passant par le pixel p_{cam} sur le plan image et se prolongeant dans l'espace en une ligne qui traverse le point P. (b) Schéma similaire pour le projecteur ; il est à noter que le plan image du projecteur n'est pas centré sur son axe optique.

l'est pas. Ainsi, il faut poser comme hypothèse que la surface de l'objet à reconstruire doit réfléchir la lumière de manière lambertienne. Cela signifie que la lumière reçue en un point d'une surface doit être réfléchie dans toutes les directions avec une intensité équivalente. De plus la surface doit réfléchir suffisamment de lumière pour pouvoir être reconstruite. Des surfaces noires (*i.e.* absorbant la lumière) ou des surfaces non lambertiennes (*e.g.* boule miroir) ne pourront donc pas être reconstruites par lumière structurée.

L'implémentation la plus simple d'un système de lumière structurée consiste à allumer chaque pixel du projecteur l'un après l'autre et à acquérir l'image correspondante avec la caméra. Ainsi pour chaque image de la caméra il faut déterminer le ou les pixel(s) p_{cam} correspondant au point de la surface illuminé par le pixel du projecteur p_{proj}. Comme illustré aux Figures 3.1 et 3.2, ces deux pixels correspondent à deux demi-droites dans l'espace. Il suffit de calculer l'intersection entre ces deux demi-droites pour obtenir la position dans l'espace du point correspondant sur la surface. Bien sûr, il est presque impossible que ces deux demi-droites s'intersectent exactement à cause des erreurs de précision dans les calibrations et des erreurs de précision numérique. En pratique c'est le point 3D le plus proche de ces deux demi-droites qui est reconstruit.

Évidemment, cette technique est très inefficace. La résolution standard des projecteurs étant de 1024×768 pixels, cela correspond à projeter et acquérir 786,432 images



(a) (b)

FIG. 3.2 – (a) Reconstruction point par point à l'aide d'un émetteur laser ; (b) reconstruction ligne par ligne en utilisant une ligne laser. Cette solution est couramment utilisée dans les systèmes de reconstruction commerciaux.

pour chaque configuration caméra/projecteur/objet. Cette technique est cependant utilisée pour reconstruire des objets translucides [GLL+04] parce que la lumière projetée sur leur surface est diffusée à l'intérieur de l'objet et peut donc ressortir en n'importe quel endroit de sa surface.

Pour les surfaces opaques lambertiennes le problème est plus simple car un élément de surface illuminé réfléchit la lumière dans toutes les directions (à l'extérieur de l'objet) avec la même intensité. Il est donc possible d'optimiser le processus en projetant des lignes de pixels plutôt que des pixels individuels. Le projecteur projette donc indépendamment chaque ligne de pixels et la caméra fait l'acquisition de ces images.

Le principe reste ensuite le même. Tout pixel p_{xy} de la caméra identifié comme étant illuminé correspond à une demi-droite L_{xy} qui intersecte un plan dans l'espace 3D Π_i lui-même défini par une ligne de pixels l_i dans le projecteur. Il suffit ainsi de calculer l'intersection entre chaque ligne L_{xy} et le plan Π_i correspondant pour reconstruire des points sur la surface. L'orientation des lignes de pixels projetées est déterminée par la géométrie épipolaire. Ces plans (Π_i) doivent être perpendiculaires au plan défini par les centres du projecteur, de la caméra et de l'objet.

Cette technique, illustrée à la Figure 3.2 (b), est beaucoup plus efficace que la précédente car il suffit maintenant de projeter et d'acquérir 1024 images. C'est cette technique qui est généralement utilisée dans les numériseurs laser 3D commerciaux de

haute qualité [Cyb].

Pour augmenter l'efficacité en diminuant le nombre d'images à projeter, il est possible d'utiliser des techniques pour encoder le signal (image) projeté par le projecteur et pour le décoder dans la caméra.

3.2 Encodage temporel

3.2.1 Encodage binaire

L'algorithme d'encodage de lumière structurée le plus simple à implémenter est l'encodage binaire. Au lieu de projeter chaque ligne du projecteur indépendamment, il est possible de projeter une série d'images composées de bandes (*stripes*) verticales illuminées et non-illuminées. Simultanément la caméra acquiert ces images et les décode pour retrouver pour chaque pixel la bande du projecteur correspondante. L'encodage binaire consiste à donner à une bande de pixels du projecteur un code, qui est habituellement sa position unidimensionnelle. Chaque bit b_i de ce code détermine l'état de cette ligne de pixels (illuminée ou non-illuminée) pour l'image i. Ainsi pour encoder une résolution de n bandes de pixels, il faut projeter et acquérir $\log_2(n)$ images (Figure 3.3 (a)).

La caméra acquiert ces images et il faut ensuite procéder au décodage. L'état de chaque pixel de la caméra est analysé en fonction du temps (images successives), en déterminant si celui-ci est illuminé ou non. Un pixel illuminé est considéré comme un 1 lors du décodage et un pixel non-illuminé est considéré comme un 0. Lorsque l'état de chaque pixel est connu pour toutes les images, il est possible de reconstruire le code initial (en assemblant les 0 et les 1) et donc identifier la bande du projecteur correspondante. Cependant, il est parfois difficile de déterminer si un pixel observé par la caméra est illuminé ou non à cause des interréflexions entre l'objet à reconstruire et son environnement. Pour augmenter la robustesse du décodage il est préférable selon Scharstein et Szeliski [SS03], de projeter l'image de motifs et son inverse afin de déterminer si un pixel est illuminé ou non. Nous avons éprouvé les mêmes difficultés avec les interréflexions et nous projetons donc successivement chaque motif binaire (image) suivi de son inverse lorsque nous utilisons cet encodage. Cette contrainte demande donc de projeter et d'acquérir $2\log_2(n)$ images de motifs pour obtenir une résolution de n codes.

En pratique, on ne peut utiliser la résolution totale du projecteur, car il faudrait pouvoir distinguer les 1024 lignes verticales dans la caméra. Notre caméra vidéo (*Panasonic PVGS-70*) a une résolution de 720×480, ce qui limite la résolution maximale utilisable. De plus certains problèmes de mise au point du projecteur et de compression de signal dans la caméra augmentent la difficulté à distinguer les signaux de haute fréquence dans la caméra (*i.e.* si l'on projette les 1024 lignes en alternant les lignes noires et blanches, on observera une image grise dans la caméra). De plus, comme expliqué dans la section suivante, l'encodage binaire souffre de l'accumulation d'erreurs aux frontières des bandes de lumière. Pour résoudre ce problème, la technique utilisée pour perturber l'encodage est le code de Gray.

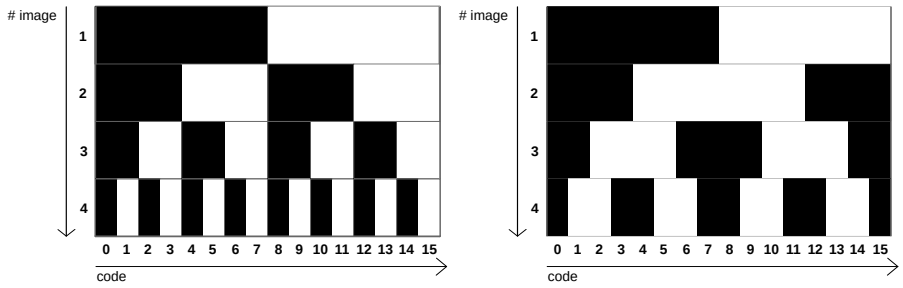


FIG. 3.3 – (a) Encodage binaire : chaque ligne horizontale représente une image de motifs projetée et chaque colonne est une séquence (code) d'illumination temporelle unique ; on peut observer la présence de la frontière du milieu (entre les codes 7 et 8) dans toutes les images du code binaire ; (b) encodage de Gray, il est à noter que chaque frontière n'apparaît temporellement qu'une seule fois. Aussi, la largeur des bandes est inversement proportionnelle au nombre de bandes.

3.2.2 Encodage de Gray

Démontré par Baudot en 1878 et breveté par Gray en 1953 dans les laboratoires *Bell*, cet encodage binaire garantit que la représentation encodée de deux nombres successifs ne diffère que d'un seul bit. À l'origine, ce code a été créé pour réduire le nombre de bits qui changent d'états dans les compteurs mécaniques ou dans les anciens circuits à relais électromagnétiques. Dans ces compteurs, changer l'état d'un bit créait un bruit sonore désagréable et l'encodage binaire a la particularité de changer l'état de tous les bits lorsque la représentation d'un nombre nécessite un bit supplémentaire (*e.g.* tous les

bits changent d'état du nombre 7 (0111) au nombre 8 (1000)). Ainsi l'encodage de Gray permet de réduire le bruit généré par les compteurs en garantissant que d'un nombre au prochain, un seul bit change d'état.

La manière la plus simple d'encoder un nombre binaire n en code de Gray est de calculer le "ou exclusif" (XOR) entre n et $n/2$ (division entière). Le décodage se fait ensuite inversement à l'aide d'une série d'opérations "ou exclusif" et de divisions entières. Voici deux fonctions en C [Enc] pour transformer un nombre entier (représenté sur 32 bits) en représentation binaire vers l'encodage de Gray, et inversement :

```
unsigned int binary_to_gray( unsigned int n )
{
  return n ^ (n >> 1) ;
}


unsigned int gray_to_binary( unsigned int n )
{
  n ^= n >> 1 ;
  n ^= n >> 2 ;
  n ^= n >> 4 ;
  n ^= n >> 8 ;
  return n ^ (n >> 16) ;
}
```

Plusieurs autres algorithmes permettent de procéder à l'encodage et au décodage de Gray, mais ces deux-là sont probablement les plus simples. Le résultat de l'encodage des nombres de trois bits est illustré dans le Tableau suivant :

Dans le contexte de la lumière structurée, l'encodage de Gray réduit l'accumulation d'erreurs aux frontières. En effet, on peut remarquer (Figure 3.3) que chaque frontière entre deux bandes n'apparaît qu'une seule fois avec le code de Gray. Cette particularité réduit les erreurs d'analyse des codes de lumière pour les pixels correspondant aux frontières. Tout comme pour le code binaire, pour obtenir une résolution de 256 bandes, il faut projeter et acquérir $2\log_2(256) = 16$ images. Comme nous voulions un système de reconstruction rapide, nous avons opté pour un algorithme d'encodage plus efficace qui se base sur la cohérence spatiale pour encoder les frontières entre deux bandes

TAB. 3.1 – Tableau des codes de Gray pour les nombres de 3 bits.

nombre	binaire	binaire/2	code binaire de Gray XOR(binaire,(binaire/2))	code de Gray
0	000	000	000	0
1	001	000	001	1
2	010	001	011	3
3	011	001	010	2
4	100	010	110	6
5	101	010	111	7
6	110	011	101	5
7	111	011	100	4

successives.

3.2.3 Encodage des frontières

Hall-Holt et Rusinkiewicz [HHR01] ont développé un système de reconstruction temps réel où l'usager déplace manuellement l'objet à reconstruire devant la caméra et le projecteur. Leur encodage particulier est beaucoup plus efficace que le code binaire ou le code de Gray. Les codes binaires de deux bandes consécutives définissent un code unique pour la frontière qui sépare ces deux bandes (*stripe boundary*). Cette modification permet d'élever au carré le nombre de codes possibles pour un nombre d'images donné. En effet, pour n images projetées, en utilisant l'encodage binaire, il existe 2^n codes possibles pour chaque bande. En utilisant le code de Hall-Holt et Rusinkiewicz, il est possible d'obtenir approximativement 2^{2n-2} codes uniques. En pratique certains codes binaires ne sont pas utilisés pour faciliter l'identification lors du décodage. Par exemple, le code binaire 0 qui correspond à "jamais illuminé" et le code binaire $2^n - 1$ qui correspond à "toujours illuminé", sont trop difficiles à discerner car leur état d'illumination ne change pas dans le temps. Aussi, pour permettre la reconstruction en temps réel d'un objet en mouvement, leur système doit suivre temporellement les frontières entre les bandes dans la caméra d'une image à la suivante. Pour faciliter le suivi dans la caméra, une autre règle est nécessaire : "chaque frontière entre deux bandes doit être au moins visible dans une image de motifs sur deux". Pour créer ce code de lumière structurée, les auteurs [HHR01] expriment le problème sous forme d'un parcours de graphe (Figure 3.4), où chaque noeud représente le code binaire d'une bande de lumière et chaque arc représente une frontière entre deux bandes de lumière. Le problème se

résout par identification du chemin le plus long dans le graphe qui passe sur chaque arc (orienté) au plus une fois. Ce chemin est trouvé par une approche probabiliste (*brute force*), et il correspond au code des frontières de bandes de lumière structurée. Leur implémentation fonctionne avec 4 images de motifs de lumière structurée, ce qui permet d'obtenir une résolution de 111 codes, qui est suffisante pour leurs besoins.

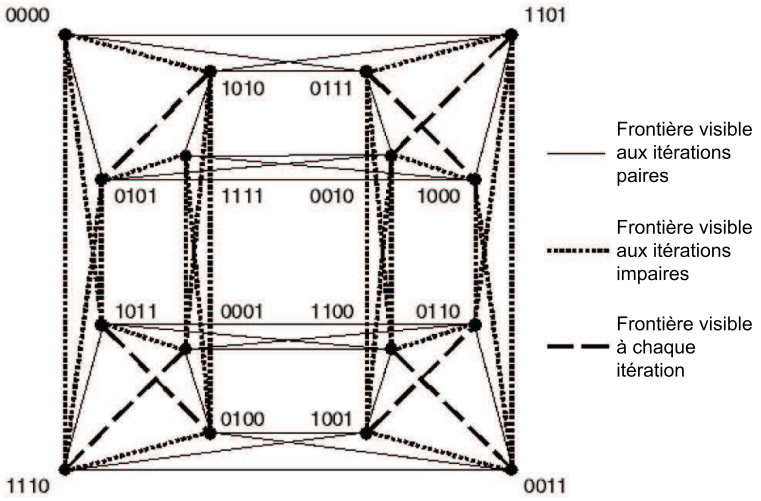


FIG. 3.4 – Graphe tiré de [HHR01]. Les noeuds représentent les codes binaires des bandes de lumière structurée et les arêtes représentent les frontières valides entre ces bandes.

Lors du décodage, la caméra acquiert les premières images et décode pour chaque pixel le code binaire auquel il correspond. Ensuite chaque ligne (*scanline*) de chaque image est parcourue en faisant une recherche des maxima et minima locaux pour identifier les frontières entre les bandes de lumière structurée. Le code d'une frontière est déterminé à l'aide des codes binaires des deux bandes adjacentes à cette frontière. Ensuite, chaque image subséquente est analysée en utilisant les images précédentes et un algorithme fait correspondre les frontières retrouvées dans cette nouvelle image aux frontières précédemment identifiées. Ainsi après avoir projeté les premiers motifs de lumière structurée, chaque nouvelle image donne une nouvelle itération de reconstruction et donc de nouveaux points. Ce système projette les motifs de lumière structurée en continu, à une fréquence de 60 Hz, et le décodage se fait lui aussi en temps réel à l'aide d'un algorithme de décodage simple et optimisé.

Toutefois, le décodage des frontières ne peut se faire correctement que si la surface

de l'objet à reconstruire est continue. Dans le cas d'une surface non continue, la frontière entre deux bandes peut ne pas être visible ou pire, une discontinuité peut être identifiée comme étant une frontière entre deux bandes. Pour éliminer les discontinuités avec l'arrière-plan sur les silhouettes externes de l'objet, les auteurs utilisent un tissu noir comme arrière-plan (qui ne réfléchit pas la lumière). Cependant, il est impossible de prévenir leur technique d'identifier de fausses frontières sur les discontinuités créées par les silhouettes internes de l'objet. Les auteurs spécifient qu'ils ont opté pour une élimination agressive des points erronés (*outliers*), mais ils ne donnent que très peu de détails sur ce sujet.

Ce système a été amélioré par les auteurs dans une publication subséquente [RHHL02]. Leur implémentation permet à l'usager de bouger manuellement l'objet devant le système lors de la reconstruction. L'ordinateur offre un retour visuel (*visual feedback*) sur l'état actuel de la reconstruction en affichant le modèle reconstruit. Ainsi l'usager peut décider de placer l'objet dans une certaine position pour remplir les trous observés dans le modèle actuel. Chaque nouvelle itération de reconstruction crée de nouveaux points et ceux-ci sont alignés avec le résultat des itérations précédentes grâce à un algorithme d'alignement ICP (*Iterative Closest Point*) que nous décrirons dans la Section 5.3. Leur système de reconstruction fonctionne en deux étapes. La première phase consiste à reconstruire l'objet avec un algorithme de décodage optimisé pour atteindre la vitesse de 60 Hz, et à enregistrer parallèlement toutes les images acquises sur un disque dur. Lors de la seconde phase, toutes les données sont récupérées dans un processus beaucoup plus long (*offline*) qui consiste à analyser précisément chaque image pour la reconstruction et à procéder à un alignement plus précis mais plus lent et ainsi obtenir après une ou deux heures un modèle de grande qualité.

3.2.4 Notre implémentation

Notre implémentation est basée sur l'encodage des frontières de lumière structurée avec certaines modifications pour l'adapter à nos besoins. L'algorithme original de décodage d'image nécessite l'analyse horizontale de lignes de pixels avec une recherche des maxima et minima locaux pour chaque image isolée. Cette approche, bien que très efficace, limite énormément la diversité des objets qui peuvent être reconstruits. En effet, si un objet comporte une texture, celle-ci risque de créer des erreurs dans l'ana-

lyse des maxima et minima locaux. Les auteurs ont souligné cette limitation et ils se sont concentrés sur la reconstruction d'objets en plâtre ne comportant pas, ou très peu de textures. Notre approche consiste à analyser en deux phases les images acquises. D'abord, l'intensité maximale et l'intensité minimale sont trouvées pour chaque pixel en analysant toutes les images. Ensuite, le décodage se fait en ré-analysant chaque pixel dans chaque image et en lui associant l'état "illuminé" ou "non-illuminé" selon si son intensité est plus proche de l'intensité maximale ou minimale identifiées pour ce pixel. Ainsi le code binaire de chaque pixel est retrouvé même si l'objet possède une texture. Cette modification permet de reconstruire des objets colorés et texturés comme illustré à la Figure 6.6.

Contrairement au système de Rusinkiewicz *et al.* [RHHL02], notre implémentation utilise 5 images de motifs de lumière structurée. Cela nous permet d'obtenir une résolution d'approximativement 400 codes de lumière structurée. Toutefois, d'après nos expérimentations, il est impossible d'utiliser une telle résolution pour les motifs projetés à cause de la résolution limitée de notre caméra, du crénelage du signal et de la compression MPEG des images transmises par notre caméra.

Nous avons développé une technique automatique qui permet de déterminer la résolution du code à utiliser en fonction de l'emplacement de la caméra, du projecteur et de l'objet à reconstruire. Comme notre système est calibré dans un même monde de référence, nous connaissons les matrices de projection de la caméra et du projecteur. De plus, nous connaissons approximativement la position de l'objet à reconstruire car celui-ci est posé sur un piédestal (Section 5.1) dont la position est connue après l'initialisation du système.

Le projecteur émet des motifs de lumière structurée à une certaine résolution res_{proj} (la largeur en pixel d'une bande est d_{proj}). Ceux-ci sont réfléchis par la surface de l'objet vers la caméra qui acquiert les images. Comme on peut l'observer à la Figure 3.5 (a), la largeur d_{cam} des bandes réfléchies dans l'espace image de la caméra dépend des orientations et positions relatives du projecteur, de la caméra et de la surface de l'objet à reconstruire. Toutes les données sont connues excepté pour l'orientation et la position de la surface de l'objet.

Nous inversons le processus pour trouver la largeur minimale des bandes d_{proj} que l'on peut projeter tout en garantissant que dans la caméra cette bande couvrira au

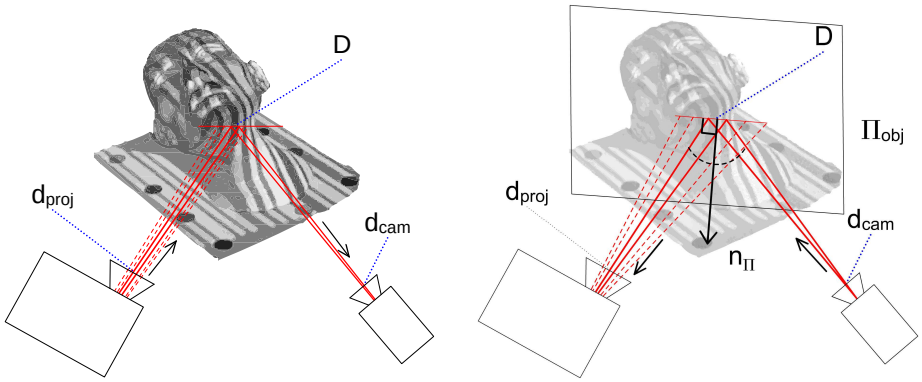


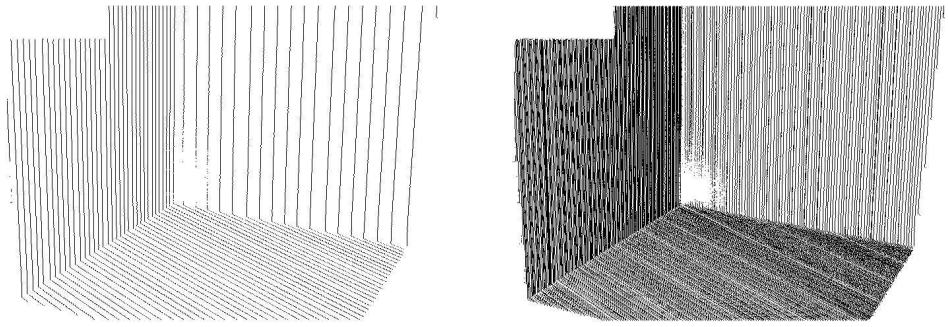
FIG. 3.5 – (a) Les bandes de lumière structurée qui sont projetées avec une largeur en pixels d_{proj}, sont réfléchies par l'objet avec une largeur inconnue D et la caméra capte ces bandes avec une largeur inconnue d_{cam} en pixels. (b) Nous inversons le processus en approximant l'objet par un plan Π_{obj} pour estimer la largeur des bandes dans le projecteur à partir de la largeur minimale des bandes d_{cam} que l'on veut observer dans la caméra.

moins d_{cam} pixels de large. Pour ce faire, nous approximons la surface de l'objet par un plan Π_{obj} normal à la bissectrice entre les vecteurs de vue de la caméra et du projecteur (Figure 3.5 (b)). Ce plan passe par le centre du piédestal dont la position est connue. L'usager spécifie la distance d_{cam} minimale désirée dans la caméra (en pixels). Le système rétro-projette, à l'aide de la matrice inverse de projection de la caméra, deux points au centre du piédestal distants de d_{cam} pixels pour obtenir deux vecteurs de l'espace, puis intersecte le plan Π_{obj} pour retrouver les deux points correspondants dans l'espace 3D. Ces deux points sont ensuite reprojetés dans le projecteur et la distance en pixels d_{proj} entre les deux points reprojetés permet de calculer la résolution maximale à utiliser dans le projecteur ($res_{proj} = \frac{1024}{d_{proj}}$ pour des motifs verticaux). Nous avons déterminé expérimentalement que les bandes de lumière structurée étaient correctement décodées lorsqu'elles occupaient au moins 4 pixels dans l'image de la caméra. Nous pouvons donc parler d'un ratio de 1 bande dans le projecteur pour 4 pixels de la caméra. Généralement, la résolution de bandes utilisée pour obtenir ce ratio varie entre 200 et 300 codes selon la disposition des éléments.

Nous appelons "correspondance" chaque pixel de la caméra décodé et identifié comme faisant partie d'une frontière entre deux bandes de lumière structurée. Cette no-

menclature découle du fait que le décodage du pixel permet de retrouver à quelle bande du projecteur celui-ci correspond. Les correspondances sont ajoutées à une "carte de correspondances", de mêmes dimensions que les images de la caméra (Figure 3.6 (a)).

Pour augmenter la densité des pixels retrouvés dans la carte de correspondances, notre implémentation permet de faire glisser d'un pixel les bandes de lumière structurée après chaque itération de reconstruction. Cette modification permet de reconstruire plus de points sur une surface sans pour autant avoir à déplacer manuellement cet objet. Par exemple, pour une résolution donnée de 200 codes des frontières de bandes, chaque bande occupe approximativement 5 pixels de large dans le projecteur. Donc si nous faisons glisser les motifs projetés de un pixel à chaque nouvelle itération, après 5 itérations de reconstruction chaque pixel du projecteur aura, à son tour, représenté une frontière de bande de lumière structurée et on aura ainsi utilisé les 1024 pixels du projecteur. Une carte de correspondances densifiée par de multiples itérations de reconstruction et "glissement" des motifs est illustrée à la Figure 3.6 (b).



(a) (b)

FIG. 3.6 – (a) Carte de correspondances de la caméra après une itération de reconstruction ; (b) carte de correspondances complétée après 7 itérations de reconstruction. La résolution utilisée dans cet exemple est de 150 codes de frontières de bandes. On peut noter que la présence d'interréflexions dans notre environnement de reconstruction empêche l'identification des codes dans le coin de celui-ci. Les cartes de correspondances ont été traitées en noir et blanc pour apparaître plus clairement lors de l'impression. Les cartes originales sont en couleurs, et chaque composante de couleur (RGB) encode une information. L'espace en haut à gauche a été réservé pour le code de synchronisation (Section 3.3).

Une manière triviale d'augmenter la densité des correspondances consiste à projeter

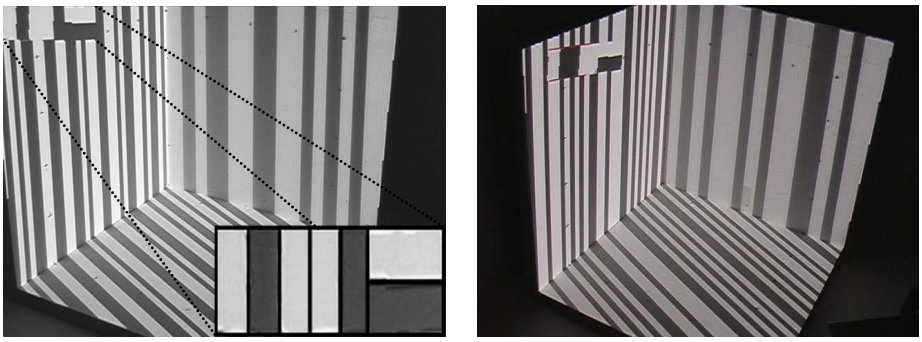
des motifs de lumière structurée horizontaux après les motifs verticaux. Cela permet d'exploiter la résolution verticale du projecteur et d'augmenter le nombre correspondances décodées dans la caméra, qui seront ajoutées à la carte de correspondances. Cependant, comme nous l'expliquerons dans la Section 3.4, pour reconstruire correctement un point 3D à partir d'une correspondance, l'angle formé entre le rayon de la caméra (pour ce pixel) et le plan du projecteur (bande correspondante) doit être supérieur à 20 degrés. Cette contrainte force donc la caméra et le projecteur à être suffisamment distants le long de l'axe horizontal du projecteur (dans l'espace) mais aussi le long de l'axe vertical si l'on veut projeter des motifs horizontaux. Malheureusement, cette contrainte pose deux problèmes : elle complique le placement de la caméra et du projecteur dans un environnement de reconstruction réel, et la quantité de surface (de l'objet à reconstruire) visible à la fois par la caméra et par le projecteur est diminuée (*i.e.* cela augmente les occlusions).

Finalement, pour extraire les couleurs des points reconstruits, notre système projette une image supplémentaire, composée d'une intensité de gris fixée par l'usager. Cette image, acquise par la caméra, est appelée "image de référence". Bien entendu, si l'objet est reconstruit sous plusieurs points de vue ou si l'illumination est modifiée d'une itération de reconstruction à une autre, les couleurs de l'objet seront bruitées. Pour éliminer ce bruit, nous filtrons spatialement les couleurs du modèle lors de l'étape de post-traitement (Section 5.3).

3.3 Synchronisation

Un système de lumière structurée comprend des périphériques matériels tels qu'un ordinateur, une caméra et un projecteur. Le projecteur projette des motifs de lumière structurée générés par l'ordinateur et la caméra en fait l'acquisition et transmet cette information à l'ordinateur sous forme d'images. Dans notre cas, un délai approximatif de 0.3 seconde est induit lors du transfert des images de la caméra vers l'ordinateur. Pour projeter et acquérir les 6 images requises pour une itération de reconstruction, cela prend donc 1.8 secondes si le système fonctionne de manière synchrone. Les projecteurs ont habituellement un taux de rafraîchissement de 60 images par seconde et les caméras NTSC reçoivent et transmettent 30 images par seconde. Le système de Rusinkiewicz *et al.* [RHHL02] maximise l'exploitation de ces caractéristiques grâce à un système

de synchronisation matériel et une carte d'acquisition vidéo de haute qualité. Cela leur permet d'acquérir et de traiter 30 images par seconde. Comme nous n'avions pas ce type de matériel à notre disponibilité, nous avons pallié à cette limitation en développant un code de synchronisation visuel utilisant le projecteur. Ce code visuel (Figure 3.7) est composé de trois éléments.



(a) (b)

FIG. 3.7 – (a) Environnement de reconstruction avec le code de synchronisation déformé pour apparaître dans le coin supérieur gauche de la caméra calibrée ; le code a été agrandi en bas à droite de l'image et les zones du code ont été mises en évidence pour faciliter la visualisation ; (b) environnement photographié d'un autre point de vue, on peut y observer la déformation du code de synchronisation. Les motifs et le code de synchronisation projetés sont différents dans les deux images.

Les quatre premières bandes verticales encodent en binaire le numéro d'identification de l'image projetée. Comme dans le cas du code des frontières, nous n'utilisons pas les codes binaires (0000) et (1111) qui correspondent respectivement à "aucune bande illuminée" et "toutes les bandes illuminées", ce qui garantit qu'au moins une des bandes sera toujours dans un état différent des autres.

La cinquième bande verticale est un booléen qui change d'état après chaque itération de reconstruction correctement terminée. Cela permet de différencier les images provenant de l'itération de reconstruction actuelle des images provenant d'une itération précédente.

Les deux dernières bandes horizontales résolvent un problème créé par les projecteurs DLP (*Digital Light Processing*). Ces projecteurs fonctionnent avec une roulette de couleurs (*color wheel*) qui tourne à plus de 180 tours par seconde. Cette roulette permet au projecteur de projeter alternativement les composantes rouge, vert et bleu de

l'image à projeter. Cette technique permet de construire des projecteurs plus puissants et plus petits que les projecteurs LCD (*Liquid Crystal Display*). Malheureusement ces projecteurs créent des artefacts visuels appelés effets arcs-en-ciel (*rainbow effect*) qui sont imperceptibles pour la majorité de la population humaine mais qui peuvent être dérangeants pour certaines personnes. Les caméras subissent aussi les effets de ces artefacts et ainsi, une image acquise peut être composée par exemple des composantes bleu de l'image i et rouge et vert de l'image $i+1$. Les deux dernières bandes horizontales de notre code de synchronisation résolvent ce problème en changeant d'état pour chaque image projetée successivement. Pour une image i projetée, la bande supérieure sera illuminée et la bande inférieure ne le sera pas. À l'image suivante, ce sera l'inverse : la bande supérieure ne sera pas illuminée et la bande inférieure le sera. Une image acquise par la caméra n'est donc valide que si une des deux bandes est illuminée (en blanc) et que l'autre ne l'est pas.

On procède à une analyse de couleurs dans un processus d'initialisation. Nous utilisons ici une analyse de couleurs développée par Granger-Piché [GP05] qui consiste à construire des ellipsoïdes englobants orientés dans l'espace RGB à partir des pixels fournis à l'initialisation. Nous créons deux ellipsoïdes lors de l'initialisation en projetant notre code de synchronisation et en analysant les pixels qui le composent : un ellipsoïde est créé pour les pixels provenant de la zone illuminée et un autre pour ceux de la zone non illuminée (illuminée indirectement par les interréflexions de la scène réelle).

Comme les images transférées par la caméra sont entrelacées, nous traitons séparément les lignes paires des lignes impaires pour l'analyse du code de synchronisation. Ainsi nous pouvons déterminer pour les lignes paires d'une image leur numéro d'identification et faire de même pour les lignes impaires. Les demi-images sont ensuite recomposées pour récupérer correctement et dans le bon ordre les images contenant les motifs projetés.

Pour analyser ce code dans l'image de la caméra, il faut qu'il soit projeté dans une zone de l'image toujours visible. Nous avons décidé de le projeter sur le plan $z = 0$ de l'environnement de reconstruction de manière à ce qu'il soit visible dans le coin supérieur gauche de l'image de la caméra avec une dimension de 150×50 pixels (Figure 3.7). Cette contrainte oblige la caméra à être placée de manière à ce que le plan $z = 0$ soit toujours visible dans le coin supérieur gauche lorsque l'on veut utiliser l'outil de synchronisation visuelle. Cependant, cela facilite beaucoup l'analyse de ce code car il suffit d'analyser

les lignes de pixels (paires ou impaires) dans ce coin de la caméra. Nous savons que les 20 premiers pixels de chaque ligne horizontale correspondent à la première bande verticale, les 20 suivants à la deuxième et ainsi de suite jusqu'à la cinquième bande. Les deux bandes horizontales sont visibles de la position 100 à 150 sur cette même ligne. La redondance des 25 lignes (50/2 à cause de l'entrelacement) permet ainsi de corroborer le décodage et d'assurer qu'une image invalide ne sera pas considérée.

Le code de synchronisation est déformé à la projection pour être visible selon la forme voulue et à la bonne position dans la caméra. Pour ce faire, nous avons utilisé la connaissance 3D de notre environnement de reconstruction, les matrices de calibration de la caméra et du projecteur, et les capacités de OpenGL. Le rectangle désiré dans l'espace image de la caméra correspond à un quadrilatère dans l'espace image du projecteur (Figure 3.7 (b)). Voici comment trouver les positions en pixels des 4 coins de ce quadrilatère dans le projecteur :
Pour chaque coin c_{xy} du rectangle dans la caméra :

1. rétro-projeter c_{xy} par multiplication avec la matrice inverse de la caméra ;
2. intersecter le rayon obtenu avec le plan de l'environnement de reconstruction dont l'équation est connue ($z = 0$) ;
3. reprojeter le point 3D obtenu avec la matrice de projection du projecteur.

Le code de synchronisation est créé sous forme de texture sur la carte graphique en utilisant OpenGL, puis le quadrilatère est dessiné dans le projecteur en utilisant la primitive GL_QUADS.

Grâce à cette technique de synchronisation, il est possible d'acquérir et d'analyser les 30 images transmises par la caméra à chaque seconde. Cependant, selon nos expériences, l'effet arc-en-ciel est très présent si notre projecteur DLP émet les motifs lui aussi à une fréquence de 30 images de seconde. Cela implique que presque la moitié des images acquises sont invalides. Ce nombre est de beaucoup diminué lorsque le projecteur n'émet que 20 images de motifs par seconde. Cette fréquence est toutefois bien plus grande que la fréquence maximum de 3 images par seconde à laquelle il faut se limiter si la boucle caméra/projecteur fonctionne de manière synchrone sans utiliser le code ni de synchronisation matérielle.

3.4 Limitations

Lorsque l'encodage des frontières est utilisé comme code de lumière structurée, un problème se pose aux silhouettes de l'objet et certains points erronés (*outliers*) sont reconstruits. La discontinuité entre un code binaire identifié sur la surface de l'objet et un code binaire identifié sur l'arrière-plan est considérée comme une frontière entre deux bandes de lumière structurée. Pour prévenir ces erreurs, un traitement spécial peut être effectué en détectant les silhouettes de l'objet et en ignorant les codes identifiés à proximité. Malheureusement ce problème peut aussi apparaître sur les silhouettes internes de l'objet (silhouettes créées par les concavités). Nous avons opté pour une solution d'élimination de ces points erronés lors du post-traitement du modèle, ce traitement est expliqué dans la Section 5.3.

Pour obtenir une reconstruction robuste et précise, il est important de considérer, pour un pixel p_{cam} identifié, l'angle entre le rayon correspondant et le plan Π_i. Plus cet angle est rasant, plus une erreur en pixels sur la localisation de la frontière entre deux bandes peut générer une grande erreur lors du calcul de l'intersection. Ainsi pour limiter les erreurs de précision, nous ne reconstruisons que les points pour lesquels l'angle entre le rayon et le plan est supérieur à 20 degrés. Pour déterminer cette valeur, nous avons placé le projecteur assez près de la caméra de façon à produire beaucoup d'erreurs de triangulation, puis nous avons essayé manuellement différentes valeurs lors de la reconstruction d'un plan. Nous avons ainsi déterminé que la contrainte de 20 degrés élimine de manière conservatrice les points où l'erreur de triangulation était visible.

Bien que la lumière structurée soit une technique très efficace et précise pour la reconstruction 3D, nous ne prétendons pas qu'elle est la meilleure et, comme toutes les autres techniques, elle possède aussi certaines limitations intrinsèques. La surface de l'objet à reconstruire doit respecter la contrainte de réflexion lambertienne. Ainsi il sera difficile, voire même impossible de reconstruire des objets très spéculaires, transparents, réfractifs ou translucides avec un système de reconstruction par lumière structurée. Aussi, la surface de l'objet à reconstruire doit réfléchir suffisamment la lumière pour permettre de distinguer dans la caméra, un élément de surface lorsqu'il est illuminé et lorsqu'il ne l'est pas.

Un autre problème que l'on peut observer à la Figure 3.6 est celui de l'illumination globale. Par exemple, dans le coin interne de notre environnement de reconstruction,

beaucoup d'interréflexions se produisent et il est difficile de distinguer si un élément de surface est illuminé ou non. Nous procédons donc à un seuillage lors du décodage et nous ignorons les pixels pour lesquels l'effet de l'illumination varie de moins de 30 niveaux d'intensités (sur 255). Ainsi on peut observer à la Figure 3.6 que les correspondances ne sont pas identifiées dans le coin interne de l'environnement de reconstruction.

L'utilisation d'un projecteur pour émettre des signaux de haute fréquence est aussi problématique. Les systèmes optiques des projecteurs conventionnels possèdent des limitations quant à la profondeur de champ (zone à l'intérieur de laquelle le signal projeté n'est pas flou). Théoriquement, plus un projecteur est loin de la surface sur laquelle il projette, plus sa profondeur de champ sera grande. Inversement, plus le projecteur est proche de la surface de projection, plus la profondeur de champ sera mince. Cette profondeur de champ dépend de la qualité de la lentille utilisée dans le projecteur et aussi de l'ouverture de l'iris. Sur certains projecteurs récents, il est possible de contrôler la taille de l'iris pour augmenter ou réduire la profondeur de champ, ce qui a respectivement pour effet de réduire ou d'augmenter la luminosité de l'image projetée. Étant donnés les projecteurs que nous avions à notre disposition, nous avons dû vivre avec ces limitations, et en pratique, nous plaçons le projecteur le plus loin que nous le pouvons de l'environnement de reconstruction en fonction des contraintes de dimensions de notre laboratoire (entre 3 et 4 mètres). La distance relative entre le projecteur et l'environnement est observable à la Figure 4.4 (c) dans laquelle on illustre un rendu synthétique du point de vue du projecteur. Cependant, certains projecteurs de haute qualité [Pro] sont construits à partir de lentilles de très bonne qualité permettant d'obtenir une grande profondeur de champ tout en nécessitant très peu de recul pour le projecteur (au moins 2 mètres).

Les caméras possèdent aussi des limitations sur leurs systèmes optiques. Généralement une distorsion radiale est présente sur la plupart des caméras de la gamme consommateur. Plusieurs techniques permettent de compenser cette distorsion, souvent à l'aide d'une table de déformation (*warping*) précalculée. Bien que notre caméra possède une distorsion radiale observable (surtout dans les coins de l'image), nous n'avons pas développé de technique particulière pour pallier à ces distorsions. Nous essayons de disposer les éléments du système pour que ceux qui nécessitent beaucoup de précision (objet, miroirs) soient le plus proche du centre de l'image.

La couverture est un des principaux problèmes de la reconstruction par lumière structurée, et présent dans pratiquement tous les algorithmes de stéréovision. Les caméras et projecteurs ont généralement une visibilité limitée de la surface des objets. Il n'est possible de reconstruire que la surface des objets visibles à la fois par les deux dispositifs (caméra/caméra en stéréovision passive ou caméra/projecteur en stéréovision active). Ainsi la plupart des systèmes de reconstruction nécessitent de déplacer soit l'objet [RHHL02], soit tout le système autour de l'objet [LPC$^+$00] pour compléter la reconstruction. Cette dernière solution nécessite un système robotisé précis et coûteux, ou encore une recalibration des périphériques pour chaque déplacement du système. Mais cette dernière solution reste encore peu praticable étant donné le poids actuel des projecteurs conventionnels (1 à 5 kg) comparé au poids des caméras (0.1 à 1 kg). Notre projecteur pèse 2.8 kg, il faut donc le fixer solidement à l'aide d'une *super-clamp* [Man] pour s'assurer qu'il ne changera pas d'orientation (sous le poids) lors de son utilisation. Comme nous le verrons dans le chapitre suivant, la recalibration manuelle nécessite beaucoup de temps de participation à l'usager. C'est ce problème de couverture qui a en grande partie motivé nos intérêts de recherche sur l'intégration des miroirs dans la reconstruction 3D.

Chapitre 4

Miroirs en lumière structurée

Comme nous l'avons indiqué plus tôt, reconstruire des objets dans un système de lumière structurée exige d'exprimer dans un même monde de référence les différentes composantes du système. Il faut donc procéder à une étape de calibration de la caméra et du projecteur. L'intégration de miroirs dans la reconstruction implique une étape d'initialisation pour estimer la pose de chaque miroir.

4.1 Calibration du projecteur

La calibration consiste à calculer une matrice de projection représentant à la fois les paramètres intrinsèques (distance focale, centre de l'image, *aspect ratio*) et les paramètres extrinsèques (position et orientation) d'un système optique (caméra ou projecteur). Comme expliqué au chapitre précédent, ces matrices permettent de passer de l'espace image du système optique à l'espace monde, et inversement. La calibration est un champ de recherche à part entière et plusieurs techniques existent pour retrouver ces matrices selon les contraintes disponibles. La technique utilisée dans notre système est expliquée en détail dans le livre de Faugeras [Fau93]. Elle permet de calculer une seule matrice affine (4×4) qui exprime autant les paramètres intrinsèques que les paramètres extrinsèques. Ce type de matrice, contrairement aux matrices de projection par perspective, permet de représenter des systèmes optiques non standards comme les projecteurs (Figure 3.1 (b)) qui ont leur axe optique décentré par rapport au plan image. La technique de calibration de Faugeras consiste à résoudre un système de contraintes, présenté sous la forme d'un système d'équations homogènes, en utilisant la décomposition en va-

leurs singulières (SVD). Les données nécessaires sont : les coordonnées 3D de points dans l'espace monde et leurs correspondances 2D dans l'espace image. En pratique, 8 à 20 correspondances (point 3D/pixel 2D) sont nécessaires pour résoudre ce système d'équations et obtenir la matrice de projection affine.

Inspirée par Raskar *et al.* [RWLB01], notre calibration du projecteur se fait en sélectionnant manuellement des correspondances entre des points 3D d'un objet de calibration connu et des points 2D dans l'image projetée par le projecteur. Dans notre cas, l'objet de calibration utilisé est notre environnement de reconstruction lui-même (Figure 4.1) dont nous connaissons les dimensions exactes et que nous avons modélisé dans *Softimage|XSI* [Sof]. Pour calibrer le système, l'usager doit sélectionner des points 3D sur le modèle de calibration et sélectionner dans l'image du projecteur le pixel qui illumine ce point de l'environnement. Pour augmenter la précision de la sélection dans l'espace image du projecteur, nous affichons une croix dessinée sous forme d'une texture OpenGL. Notre interface usager permet de sélectionner avec une précision sous-pixel la position 2D. La croix est affichée à la position désirée (Figure 4.1 (b)) avec une précision au quart de pixel induite par le filtrage de texture de la carte graphique.

Pour afficher le point de vue du projecteur et de la caméra avec OpenGL, nous décomposons la matrice de projection affine en deux matrices de projection perspective grâce à un algorithme donné en annexe dans [GP05].

4.2 Calibration de la caméra

La calibration manuelle de la caméra suit un procédé similaire. L'image acquise par la caméra est affichée par le système et l'usager doit sélectionner dans l'image le pixel 2D qui correspond au point 3D recherché. Comme dans la calibration du projecteur, pour augmenter la précision de la calibration, nous pouvons sélectionner la position 2D avec une précision sous-pixel. Pour ce faire, un outil d'agrandissement avec effet oeil-de-poisson (*fish-eye*) permet de visualiser plus précisément les détails de l'image à l'endroit où l'usager clique (Figure 4.1 (c)). La formule de cette déformation non-linéaire pour chaque pixel p_{ij} est :

$$p_{ij} = p_{ctr} + (p_{ij} - p_{ctr}) * \left(\frac{zoom\_factor}{zoom\_radius} * distance(p_{ctr}, p_{ij}) \right)^2 * zoom\_radius \quad (4.1)$$

où p_{ctr} est la position du centre d'intérêt (en pixels) de l'agrandissement, $zoom\_factor$ est le facteur d'agrandissement ($1.0 < zoom < 2.0$) et $zoom\_radius$ est le rayon de l'effet d'agrandissement (ou la moitié de la taille de la fenêtre). Nous utilisons une fenêtre d'agrandissement de 80×80 pixels avec un rayon ($zoom\_radius$) de 40 pixels et un facteur d'agrandissement ($zoom\_factor$) de 1.6. Parfois, la luminosité est trop faible ou trop forte pour que le point réel (noir) de calibration sur l'environnement soit visible dans la caméra. Pour régler ce problème, une égalisation de l'histogramme des intensités est appliquée à l'intérieur de la fenêtre d'agrandissement. Une intensité maximale et une intensité minimale sont trouvées (sur le rouge, le vert ou le bleu) dans la fenêtre d'agrandissement, puis les composantes RGB de chaque pixel sont redistribuées linéairement entre 0 et 255. La même transformation est appliquée à chaque composante de couleur de chaque pixel.

Pour faciliter le processus de calibration, il est possible de calibrer la caméra automatiquement grâce au projecteur. Le projecteur étant déjà calibré, il est possible de calculer pour tout point 3D situé sur l'environnement de reconstruction la position correspondante dans l'image du projecteur. Il suffit d'illuminer ce pixel dans le projecteur et de retrouver sa position dans l'image de la caméra pour obtenir une correspondance 2D/3D. En faisant ainsi pour 8 points ou plus, il est possible de calibrer automatiquement la caméra. Un algorithme basé sur l'analyse de couleurs a donc permis à Granger-Piché [GP05] de développer une technique de calibration automatique, robuste et efficace pour calibrer la caméra très rapidement à chaque nouvelle image acquise (Figure 4.1 (d)). Malheureusement, cette technique, bien que très pratique, ne permet pas d'obtenir une calibration aussi précise que la calibration manuelle.

Après calibration, l'erreur de calibration est estimée en calculant la moyenne des distances entre les points 3D reprojetés par la matrice de calibration et les pixels correspondants dans l'image. Avec la calibration automatique de la caméra, l'erreur moyenne de reprojection obtenue est généralement comprise entre 0.8 et 2.0 pixels d'erreur. Lorsque la calibration manuelle est effectuée avec l'aide de l'outil d'agrandissement, l'erreur moyenne de reprojection est comprise entre 0.5 et 0.9 pixel d'erreur.

Ainsi la tâche de recalibrer le système (caméra/projecteur) nécessite une participation active de l'usager, qui requiert généralement de 2 à 4 minutes. L'idée de déplacer le projecteur et la caméra autour de l'objet et de les recalibrer manuellement semble

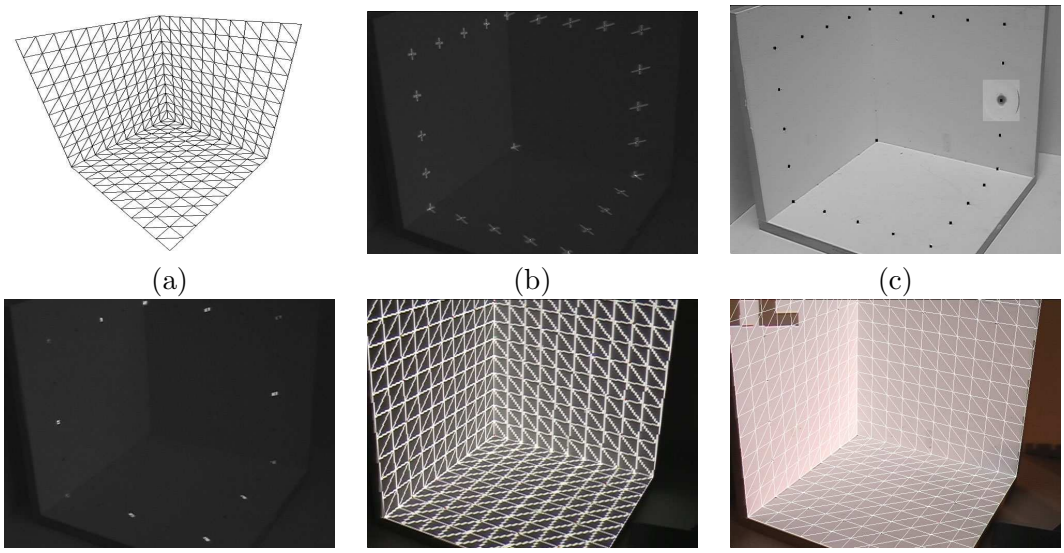


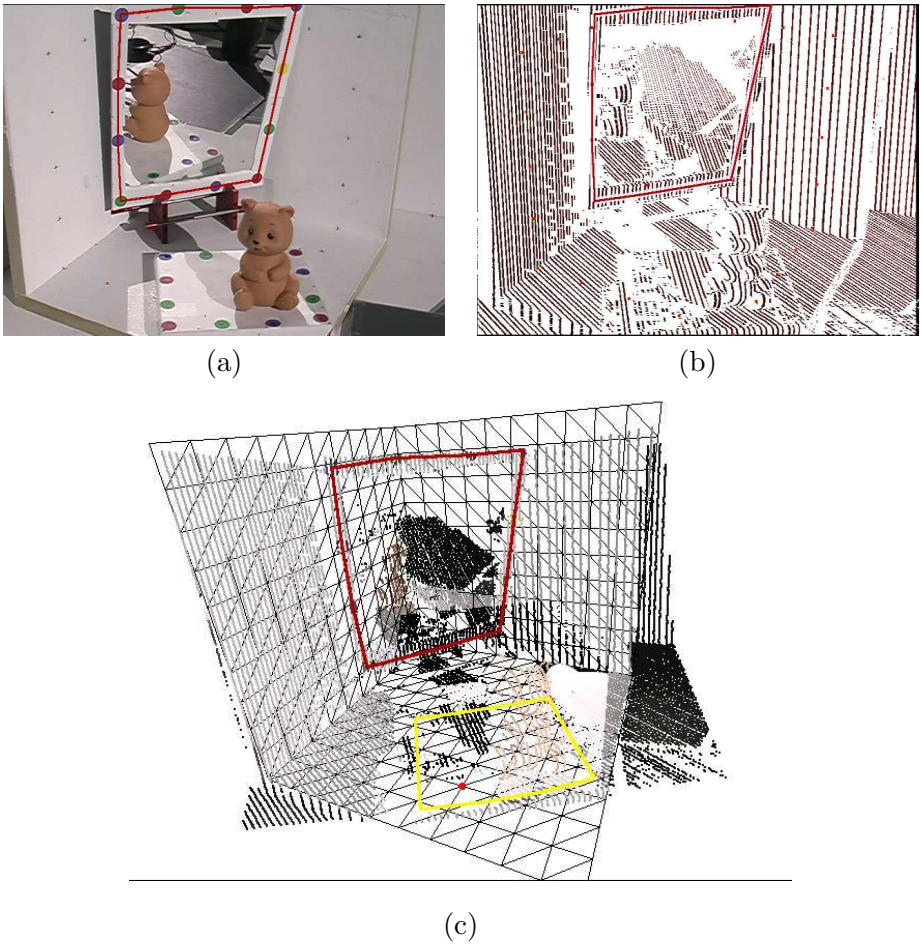
FIG. 4.1 – (a) Modèle 3D de l'environnement de reconstruction ; (b) calibration sous-pixel du projecteur ; (c) calibration manuelle de la caméra avec l'outil *fish-eye* ; (d) calibration automatique de la caméra à l'aide de points de couleurs projetés ; (e) environnement de reconstruction synthétique projeté sur l'environnement réel par le projecteur ; (f) environnement de reconstruction synthétique affiché par-dessus l'image de la caméra. Les deux dernières images (e, f) permettent de valider visuellement la précision des matrices de projection retrouvées respectivement pour le projecteur et pour la caméra. Dans les deux cas, la matrice de calibration retrouvée est utilisée pour afficher le modèle synthétique du bon point de vue.

donc prohibitive. D'autres solutions sont possibles comme d'unir le système de reconstruction à un bras robotisé de haute qualité tel que l'ont fait Levoy *et al.* [LPC+00] (Section 5.2). Mais le résultat est un système assez coûteux et plutôt encombrant. Il est aussi possible de déplacer l'objet devant le système de reconstruction (ce qui est analogue à déplacer la caméra et le projecteur autour de l'objet), tel que l'ont fait Rusinkiewicz *et al.* [RHHL02], mais cette solution n'est pas toujours applicable si l'objet est lourd ou s'il est fixé (*e.g.* statue ou environnement architectural).

Alternativement, un miroir offre la possibilité d'obtenir un nouveau point de vue à la fois pour la caméra et pour le projecteur. De plus, un miroir est léger et peut donc être facilement déplacé par l'usager. Enfin, comme nous le verrons dans la section suivante, il est possible d'estimer rapidement et automatiquement l'équation d'un miroir planaire et ainsi la transformation rigide associée (matrice de réflexion).

4.3 Pose d'un miroir

Pour déterminer l'emplacement d'un miroir planaire, nous commençons par estimer son plan de support. Nous avons disposé sur le contour de la surface du miroir des bandes adhésives de papier blanc. Pour aider au suivi temporel 3D du déplacement manuel du miroir (Section 5.2), nous avons ajouté des marqueurs de diverses couleurs (pastilles adhésives) disposés régulièrement sur les bandes blanches (Figure 4.2 (a)).



(a) (b)

(c)

FIG. 4.2 – (a) Sélection des marqueurs de couleur pour définir le contour du miroir ; (b) carte des correspondances avec identification des contours du miroir ; (c) points 3D reconstruits et géométrie du miroir identifiée en 3D (ligne rouge). Comme nous ne pouvons faire d'hypothèse sur l'emplacement du miroir dans l'espace avant de connaître sa pose, nous projetons les motifs de lumière structurée partout où le projecteur peut émettre. Les points reconstruits ailleurs que sur le contour du miroir ne sont présents dans la figure (c) que pour des fins d'illustration.

Une itération de lumière structurée est utilisée pour reconstruire les points 3D reposant sur les bandes adhésives blanches. Pour savoir où rechercher l'information dans la caméra, l'usager d'abord sélectionne manuellement les marqueurs de couleur pour en définir le contour (Figure 4.2 (a)) dans l'image de la caméra lors de l'étape d'initialisation. Un algorithme, décrit à la Section 5.1, re-centre automatiquement la sélection de l'usager sur le centre du marqueur dans l'image. Le projecteur émet ensuite les motifs de lumière structurée et le système reconstruit des points 3D pour toutes les frontières de bandes de lumière structurée identifiées. Les points 3D reconstruits sur la zone de contour du miroir (Figure 4.2 (b)), soit 10 pixels de part et d'autre des lignes de contour sélectionnées par l'usager dans l'image de la caméra, sont considérés comme des échantillons 3D du plan de support du miroir.

Un algorithme basé sur l'analyse des composantes principales (PCA), illustré en 2D à la Figure 4.3, est utilisé pour estimer le plan moyen passant par ces points 3D. Le PCA permet de compresser la dimensionnalité d'un groupe de données multidimensionnelles. Le but est de retrouver les orientations principales d'un groupe de points (ou paramètres). En 3D, le PCA peut servir à retrouver deux orientations principales et indépendantes dans un nuage de points. Si le nuage de points représente une surface reconstruite, les deux orientations correspondent alors à deux tangentes de cette surface, et on peut retrouver la normale du plan par un simple produit vectoriel. La composante D du plan est finalement trouvée en résolvant l'équation du plan (Equation 4.2) pour un point donné ou pour le centre du groupe de points. Le même algorithme est utilisé pour estimer une normale à partir d'un nuage de points.

$$\Pi_{mir} : Ax + By + Cz + D = 0. \tag{4.2}$$

Une fois l'équation du miroir trouvée, nous pouvons calculer les positions 3D des marqueurs de couleur sur le contour du miroir. Nous connaissons les positions 2D de chacun de ces marqueurs dans l'image de la caméra et nous connaissons sa matrice de calibration affine. Ainsi, en multipliant chaque position 2D par la matrice inverse de projection $\mathbf{M}_{cam}^{-1}$ nous obtenons les vecteurs 3D correspondants dans l'espace monde. Il est possible ensuite, par simple calcul d'intersection rayon/plan, de retrouver les positions 3D de ces marqueurs de couleur. Nous obtenons ainsi la géométrie du miroir dans l'espace monde (Figure 4.2 (c)), et donc, en utilisant la matrice de calibration du

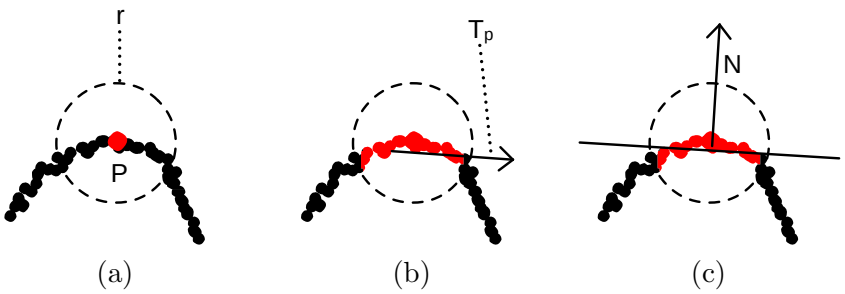


FIG. 4.3 – Illustration 2D de l'estimation d'un plan moyen à partir d'un nuage de points. (a) Pour un point donné P, les points voisins dans un rayon r sont utilisés pour calculer le PCA ; (b) en 2D, on obtient une seule orientation principale T_p pour ce groupe de points ; (c) on peut donc calculer le vecteur N perpendiculaire à T_p et finalement retrouver l'équation du plan tangent à la surface.

projecteur, la géométrie correspondante dans l'image du projecteur.

De plus, l'équation du miroir permet de calculer la réflexion de n'importe quel point de l'espace en utilisant l'équation suivante :

$$P_r = P + N_{mir} \cdot (-2(N_{mir} \cdot P) + D) \tag{4.3}$$

où N_{mir} est la normale du plan Π_{mir} (*i.e.* le vecteur (A, B, C) normalisé), P est le point à réfléchir et P_r est le point réfléchi résultant. Cette équation est utilisée depuis très longtemps en infographie, particulièrement dans les moteurs de rendu par lancer de rayons. Toute transformation linéaire peut être exprimée par une matrice et de plus, la combinaison de ces transformations s'effectue simplement en multipliant les matrices de transformations. La décomposition suivante montre comment obtenir la matrice de réflexion d'un miroir :

$$\mathbf{M}_{mir} = \mathbf{T}(0,0,D)\mathbf{A}_{mir}^{-1}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\mathbf{A}_{mir}\mathbf{T}(0,0,-D) \tag{4.4}$$

où D est la distance minimale entre le plan de support du miroir et l'origine, $\mathbf{T}$ est une translation et $\mathbf{A}_{mir}$ est la matrice de rotation qui aligne le système de coordonnées de l'espace monde avec le système de coordonnées local du miroir (reposant sur $Z = 0$ avec pour normale $Z = 1$).

Ainsi, pour calculer la réflexion d'un point par le miroir à l'aide de la matrice de

réflexion, il suffit de les multiplier ensemble :

$$P_r = \mathbf{M}_{mir} P. \tag{4.5}$$

4.4 Boîte englobante

Pour utiliser un miroir, nous devons connaître approximativement où se situe l'objet dans l'environnement de reconstruction. En effet, l'objet peut bloquer une partie du miroir selon le point de vue de la caméra ou du projecteur. Nous approximons le volume contenant l'objet à reconstruire par un parallélépipède rectangle ou boîte englobante (*bounding box*). Cette boîte englobante est créée en sélectionnant dans la caméra des marqueurs de couleur sur le piédestal, décrit au chapitre suivant, dont la face supérieure correspond à un plan parallèle au plan du sol de l'environnement de reconstruction. Les positions 3D de ces points sont retrouvées en calculant l'intersection du plan du piédestal et des rayons correspondant aux positions 2D dans la caméra, de la même manière que pour retrouver les positions 3D du miroir. La hauteur de la boîte englobante est choisie manuellement par l'usager. En affichant cette boîte dans le projecteur, l'usager peut approximativement estimer si la hauteur est suffisante en observant si l'objet est entièrement illuminé par la projection de cette boîte. Il est possible aussi d'afficher la boîte selon le point de vue de la caméra et de s'assurer que la boîte couvre entièrement l'image de l'objet.

Dans la section suivante nous verrons comment utiliser ces étapes de calibration afin de nous permettre d'utiliser les miroirs dans notre système de reconstruction 3D.

4.5 Miroir

Grâce à la matrice de réflexion du miroir, il est possible de retrouver la matrice de projection de la caméra réfléchie par le plan de support de ce miroir. Il suffit de multiplier la matrice de réflexion du miroir par la matrice de projection de la caméra :

$$\mathbf{M}_{cam\_r} = \mathbf{M}_{cam} \mathbf{M}_{mir}. \tag{4.6}$$

De la même manière nous pouvons calculer la matrice représentant le projecteur tel que réfléchi par le plan de support du miroir. La géométrie 3D du miroir délimite la

zone couverte par le miroir dans la caméra et dans le projecteur.

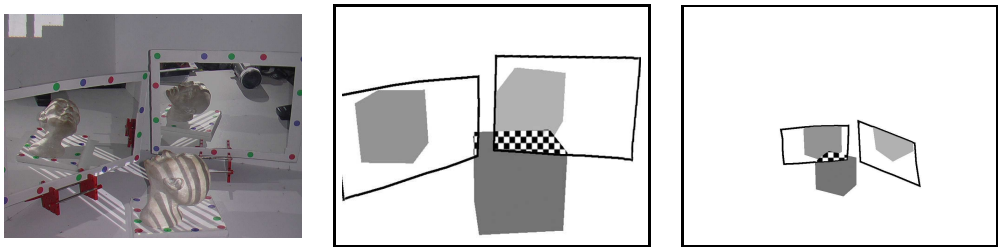
Pour utiliser un miroir dans notre système, nous devons connaître pour chaque pixel de l'image de la caméra (et du projecteur), s'il correspond à un point directement visible sur l'objet ou un point réfléchi par un miroir, afin de déterminer quelle matrice de projection utiliser (M_{cam} ou M_{cam_r}). Pour la caméra comme pour le projecteur, un pixel peut appartenir à l'une de ces quatre catégories suivantes :

1. directement à la boîte englobante de l'objet ;
2. à la boîte englobante de l'objet réfléchie dans le miroir ;
3. aux deux premières catégories à la fois (considérée alors comme zone de conflit) ;
4. à aucune des deux premières catégories (considérée sans information).

Si un pixel appartient à l'une des deux premières catégories (dans la caméra ou dans le projecteur), il peut être utilisé pour la reconstruction. S'il appartient à l'une des deux dernières catégories, il n'est pas traité. Pour la troisième catégorie, comme il est impossible de distinguer si c'est l'objet ou le miroir qui est visible, nous ne pouvons pas en tirer d'information sûre. Il serait possible d'imaginer une structure hiérarchique de cubes (*octree*) qui se raffinerait au long de la reconstruction et qui permettrait de réduire les zones de conflit en réduisant le volume englobant. Nous laissons cependant cette partie pour nos travaux futurs. Si le pixel provient de la dernière catégorie, il est évidemment inutile de projeter un motif dans ce pixel (projecteur) ou d'analyser l'information provenant de ce pixel (caméra).

Nous devons donc calculer une "carte des zones" qui contient pour chaque pixel l'information sur son état (valide, invalide ou inutile) et un numéro d'identification s'il s'agit d'un miroir.

Depuis longtemps, les miroirs ont été étudiés en infographie. Beaucoup de techniques ont été développées pour permettre le rendu efficace de miroirs par lancer de rayons (*ray tracing*) ou par *Z-Buffer* avec les cartes d'accélération graphiques. Pour intégrer plusieurs miroirs dans notre système, nous avons implémenté une technique standard de rendu de miroirs en OpenGL développée pour les jeux vidéo [Ope].



(a) (b) (c)

FIG. 4.4 – (a) Image observée par la caméra ; (b) carte des zones du point de vue de la caméra ; (c) carte des zones du point de vue du projecteur. Les zones de conflit sont illustrées par un motif d'échiquier.

La technique originale fonctionne comme suit :

1. Faire le rendu de la scène ;
2. Pour chaque miroir :
   - faire le rendu du miroir dans le STENCIL_BUFFER avec un STENCIL_ID unique ;
   - réfléchir le point de vue de la caméra à l'aide de la matrice de réflexion du miroir ;
   - créer un plan de découpage (*clipping plane*) selon le plan de support du miroir (pour éviter de dessiner les objets situés en arrière du miroir) ;
   - sélectionner le STENCIL_ID du miroir pour ne dessiner que sur les pixels couverts par le miroir ;
   - refaire le rendu de la scène.
   - désactiver le plan de découpage ;

Ainsi, nous utilisons le processeur graphique à l'aide de la librairie OpenGL pour calculer les cartes des zones. Nous affichons un polygone pour chaque miroir et un parallélépipède rectangle pour la boîte englobante. Comme il semble impossible jusqu'à présent de récupérer les informations contenues dans le STENCIL_BUFFER, nous faisons le rendu des informations simultanément dans le STENCIL_BUFFER et dans le COLOR_BUFFER. Le STENCIL_BUFFER permet de sélectionner sur quelle zone émettre les motifs depuis le projecteur. Le COLOR_BUFFER est utilisé pour récupérer la carte des zones calculée par la carte graphique et savoir où analyser l'information dans les images acquises (Figure 4.4 (b), (c)).

Nous avons étendu la technique de rendu de miroirs pour permettre le rendu des

interréflexions entre les miroirs par itérations multiples (récursivement). Cette modification nous permet de calculer les cartes des zones nécessaires pour l'utilisation de plusieurs miroirs tout en évitant les conflits. L'algorithme modifié consiste en les étapes suivantes :

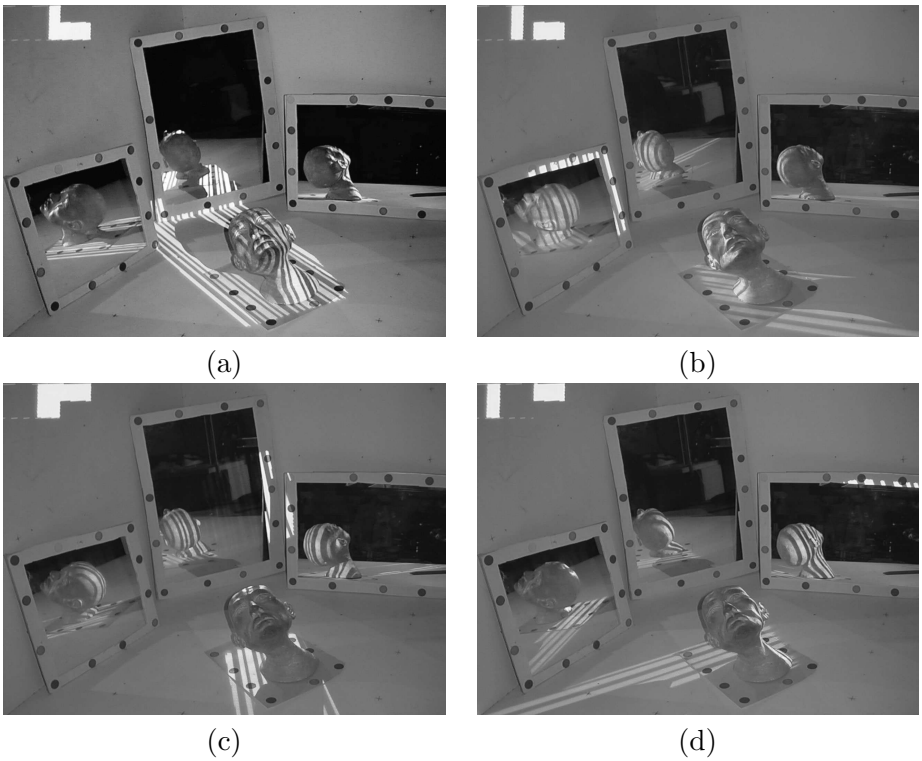
1. Faire le rendu de la boîte englobante ;
2. Pour chaque miroir :
   - faire le rendu du miroir dans le STENCIL_BUFFER et dans le COLOR_BUFFER avec un STENCIL_ID unique ;
   - réfléchir le point de vue de la caméra à l'aide de la matrice de réflexion du miroir ;
   - créer un plan de découpage selon le plan de support du miroir ;
   - sélectionner le STENCIL_ID du miroir pour ne dessiner que sur les pixels couverts par le miroir ;
   - désactiver le plan de découpage ;
   - exécuter récursivement depuis l'étape 1 tant que le niveau maximum de récursion n'est pas atteint.

Le niveau maximum de récursion peut être spécifié par l'usager, mais par défaut il est réglé à trois niveaux de récursion. En général, très peu de points seraient reconstruits après trois réflexions.

Pour reconstruire avec ces miroirs, le projecteur ne peut émettre de motifs simultanément sur l'objet et dans un miroir car les motifs de lumière se mélangeraient sur les zones de recouvrement et il serait alors impossible de distinguer les différents motifs. Pour éviter ce problème, notre système considère chaque configuration indépendamment. Ainsi, le projecteur émet successivement les motifs sur l'objet pour une première itération de reconstruction, puis uniquement dans un miroir, puis dans un autre miroir, puis ainsi de suite jusqu'à recommencer le cycle (Figure 4.5). La caméra quant à elle peut sans créer de conflits, analyser l'information de toutes les configurations (*i.e.* de tous les points de vue) en même temps.

4.6 Motifs de lumière structurée

Lorsque le projecteur émet les motifs de lumière structurée dans un miroir, ceux-ci sont réfléchis sur l'objet. Selon l'orientation de la caméra, du projecteur, de l'objet,



(a) (b)

(c) (d)

FIG. 4.5 – (a) Projection uniquement sur l'objet ; (b) projection uniquement sur le miroir de gauche ; (c) projection uniquement sur le miroir du milieu ; (d) projection uniquement sur le miroir de droite. Les motifs peuvent être analysés par la caméra simultanément sur l'objet et dans ses réflexions dans les miroirs.

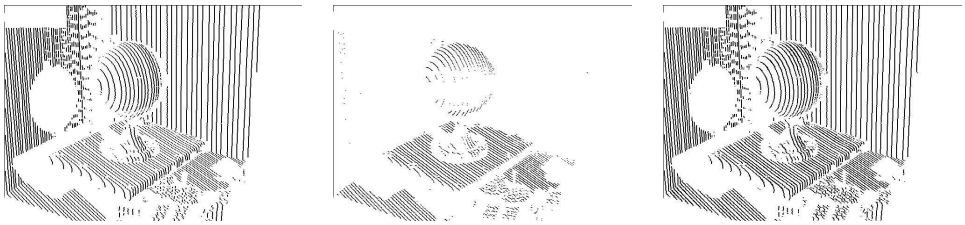
et du miroir, les motifs observés par la caméra sont rarement orientés verticalement tels que projetés à l'origine. Pour trouver analytiquement la transformation qui s'y applique, il faudrait connaître l'orientation de la surface de l'objet à reconstruire, ce qui est évidemment impossible car c'est cette même surface que l'on tente de reconstruire. Nous avons apporté deux modifications au code de Rusinkiewicz *et al.* [RHHL02] pour n'avoir à faire aucune hypothèse sur l'orientation de la surface.

La première modification permet de décoder les motifs de frontières de bandes même si ceux-ci sont inversés après réflexion dans un miroir. Pour ce faire, nous avons ajouté une contrainte lors du parcours de graphe représentant les codes de lumière structurée. Nous interdisons l'inclusion d'un code de frontière entre deux codes binaires $A \rightarrow B$ si celui-ci existe déjà dans l'autre sens ($B \rightarrow A$). Ainsi, lors du décodage, nous sommes assurés qu'il n'existe qu'un seul code de frontière qui lie deux codes binaires adjacents A et B. Cette contrainte a pour effet de diviser par deux la longueur du chemin le plus

long dans le graphe, car chaque arête ne peut maintenant être traversée qu'une seule fois. De plus, nous avons enlevé une contrainte sur le code original des frontières, inutile dans notre système de reconstruction. Lorsque Hall-Holt et Rusinkiewicz [HHR01] ont développé leur code de frontières de bandes, leur but était de développer un motif qui peut être suivi temporellement dans la caméra si l'objet à reconstruire est déplacé. Pour faciliter le suivi des frontières de bandes dans la caméra, ils ont ajouté la contrainte suivante : "chaque frontière doit être visible au minimum dans une image de motifs sur deux". Comme nous n'avons pas implémenté le suivi des frontières de bandes, nous ne requérons pas cette contrainte sur la fréquence de présence des frontières. Cela nous force donc à garder l'objet fixe à l'intérieur d'une itération de reconstruction. Sans cette contrainte, la longueur du chemin le plus long dans le graphe est deux fois plus grande et donc la résolution maximale du code de frontières est aussi deux fois plus grande. Ainsi, le fait d'enlever cette contrainte compense effectivement pour la contrainte que nous ajoutons sur l'unicité des codes de frontières.

Notre dernière modification concerne l'algorithme de décodage des images de motifs. L'algorithme original consiste à déterminer les codes de frontières de bandes en parcourant les lignes de pixels horizontales dans la caméra. En appliquant directement cet algorithme, peu de pixels appartenant aux frontières sont décodés lorsque les bandes de lumière structurée ne sont pas orientées verticalement (de -45 à 45 degrés) selon le point de vue de la caméra (Figure 4.6 (a)). Notre implémentation intègre une deuxième étape de décodage qui consiste à parcourir les lignes de pixels verticales pour décoder les frontières de bandes qui ont été ignorées lors du décodage horizontal (Figure 4.6 (b)). Cette deuxième étape permet de décoder généralement beaucoup plus de motifs et ce, peu importe leur orientation dans l'image de la caméra. Ce dernier point s'est révélé fort utile dans l'utilisation des miroirs.

Grâce à ces deux modifications, le système peut décoder les motifs projetés, peu importe leur orientation, et même si ceux-ci sont inversés à cause d'une réflexion miroir.



(a) (b) (c)

FIG. 4.6 – (a) Décodage le long des lignes de pixels horizontales ; (b) décodage le long des lignes de pixels verticales ; (c) décodage combiné le long des lignes de pixels horizontales et verticales.

Chapitre 5

Système de reconstruction interactif

En reconstruction 3D, il est traditionnel de développer des outils automatiques qui n'obligent pas l'usager à intervenir en tout temps. Mais il est parfois indispensable de lui donner un contrôle manuel lorsque les outils automatiques ne fonctionnent pas parfaitement. Nous avons développé plusieurs outils dans notre système interactif où le système peut influencer les décisions de l'usager en lui fournissant des informations sur la reconstruction actuelle ou, par exemple, en lui indiquant un meilleur agencement potentiel des miroirs. L'usager peut ainsi modifier la position de l'objet ou d'un miroir pour permettre la reconstruction de points dans les zones moins accessibles de l'objet.

5.1 Piédestal

Pour modifier la position de l'objet à reconstruire, nous avons introduit un piédestal que l'usager peut déplacer manuellement tandis que le système procède à sa poursuite (*tracking*) temporelle.

5.1.1 Initialisation

Le piédestal consiste en une boîte mince recouverte d'une feuille de papier et de quelques marqueurs de couleur identiques à ceux disposés sur le miroir. Ce piédestal est contraint à reposer sur le plan horizontal de l'environnement de reconstruction et ne peut donc être que glissé ou tourné sur ce plan.

Idéalement, nous voudrions avoir la liberté de déplacer l'objet librement dans l'espace 3D. Plusieurs raisons nous ont incités à contraindre le mouvement du piédestal au plan horizontal. Pour déplacer l'objet librement et en faire le suivi temporel, la solution d'un piédestal est plus ou moins applicable car il faudrait alors "coller" l'objet au piédestal, ce qui n'est pas toujours possible (selon le volume de l'objet et son poids). De plus il faudrait procéder à l'estimation de pose en 3D du piédestal qui, selon nos expérimentations [QL98], est moins précise que l'estimation de pose en 2D. Il faudrait aussi que l'objet et son piédestal demeurent parfaitement statiques pour les 6 images consécutives, ce qui est plus difficile dans l'espace 3D. Finalement, coller l'objet sur un piédestal empêche de reconstruire la partie collée de l'objet, qui n'est alors plus visible.

La technique utilisée par Rusinkiewicz *et al.* [RHHL02], que nous avons décrite plus tôt, consiste à déplacer librement l'objet à bout de bras et à aligner les points résultant de chaque nouvelle itération de reconstruction avec ceux issus des itérations précédentes. Cette technique, bien que beaucoup plus flexible, possède certaines limitations comme l'accumulation des erreurs à chaque nouvel alignement (*drift*). Par exemple, si l'objet à reconstruire est un vase, après un tour complet de l'objet (manuellement devant le système), les points reconstruits d'un côté et de l'autre du vase risquent de ne pas s'aligner correctement à cause des erreurs d'alignement accumulées à chaque itération de reconstruction.

La contrainte du plan permet donc de calibrer facilement le piédestal, et le suivi temporel 2D permet de retrouver les positions 3D de chaque marqueur et par la suite à calculer précisément la transformation 2D sur le plan engendrée par le déplacement du piédestal.

Pour l'initialisation, l'usager sélectionne manuellement les marqueurs de couleur sur le contour du piédestal. La couleur de chaque pixel est représentée par un point dans l'espace RGB. Ainsi pour chaque marqueur, les couleurs des pixels sont accumulées dans l'image de la caméra, dans un voisinage de 5×5 pixels autour du pixel sélectionné. Des ellipsoïdes englobants sont créés dans l'espace des couleurs RGB (Figure 5.1) à partir de ces couleurs accumulées. Par la suite, un pixel sera considéré comme faisant partie d'un marqueur si sa représentation dans l'espace RGB se trouve à l'intérieur de l'ellipsoïde correspondant à ce dernier. Les positions des marqueurs sont re-centrées à l'aide des ellipsoïdes englobants. Pour chaque marqueur, un voisinage de 30×30 pixels est

considéré autour du point initial sélectionné. La position 2D du marqueur est re-centrée en calculant la position moyenne des pixels de ce voisinage appartenant à l'ellipsoïde englobant.

Connaissant la hauteur d en millimètres du piédestal, il suffit de calculer l'intersection du plan horizontal d'équation $(Y = d)$ avec les rayons issus des positions 2D dans la caméra pour obtenir les positions 3D des marqueurs. C'est à partir de cette information que le volume englobant 3D de l'objet est créé.

Après l'initialisation le système connaît pour chaque marqueur : sa position 3D, sa position en pixels dans la caméra et dans le projecteur, et l'ellipsoïde englobant RGB contenant les pixels de couleurs telles que captées par la caméra.

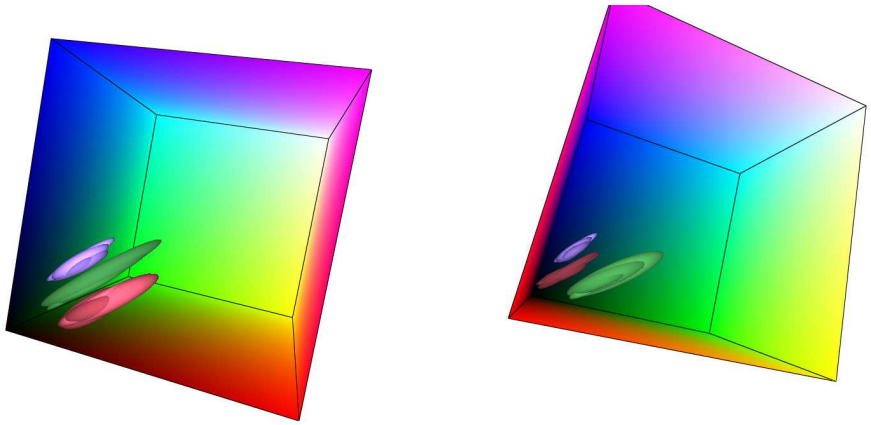


FIG. 5.1 – Cube RGB selon deux points de vue différents avec les ellipsoïdes englobants qui représentent les couleurs des marqueurs du piédestal, telles que captées par la caméra.

5.1.2 Suivi temporel (*tracking*)

L'utilité principale du piédestal est de permettre le déplacement manuel de l'objet à reconstruire. Pour reconstruire correctement l'objet selon plusieurs poses, le système doit retrouver les matrices de transformations (rigides) permettant de passer de la pose actuelle à la pose de référence de l'objet.

La poursuite temporelle est assurée en 2D par la caméra. À chaque nouvelle image, chaque marqueur de couleur est recherché dans l'image de la caméra dans un voisinage de 30×30 pixels autour de sa dernière position retrouvée. Cette recherche est faite à l'aide des ellipsoïdes englobants.

De même que pour l'initialisation, la connaissance des positions 2D dans la caméra permet de retrouver les positions 3D correspondantes par intersection avec le plan du piédestal. À partir des positions 3D initiales et des positions actuelles, il est possible de retrouver la transformation rigide par résolution d'un système de contraintes. Cette transformation rigide sur le plan $Y = d$ est représentée par la matrice 4×4 suivante :

$$\mathbf{M}_{piédestal} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & T_x \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

où θ est l'angle de rotation autour de l'axe des Y, T_x représente la translation selon l'axe des X et T_z la translation selon l'axe des Z. Il est donc possible de représenter ce problème sous forme d'un système d'équations ayant pour inconnues $\cos\theta$, $\sin\theta$, T_x et T_z. Chaque paire de points (P_{src}, P_{dst}) permet de créer les deux équations suivantes :

$$\begin{aligned} P_{dst}(x) &= P_{src}(x) * \cos\theta + P_{src}(z) * \sin\theta + T_x \\ P_{dst}(z) &= P_{src}(x) * -\sin\theta + P_{src}(z) * \cos\theta + T_z. \end{aligned}$$

Au moins 2 points sont nécessaires pour obtenir 4 équations et ainsi trouver une solution. Cependant un plus grand nombre de points mène à un système sur-contraint et donc à une solution plus robuste. Ce système d'équations est non homogène, il est donc possible de le résoudre par la méthode des "moindres carrés" [PTVF92].

Deux mesures d'erreur quantifient la précision de l'estimation de la pose. La première mesure

$$\frac{1}{n}\sum_{i=1}^{n} |distance(P_{dst}, \mathbf{M}_{piédestal} P_{src})| \tag{5.2}$$

est la moyenne des distances euclidiennes entre les n points 3D retrouvés et les points 3D initiaux après transformation par la matrice du piédestal retrouvée. Ce qui correspond à vérifier que la matrice transforme bien les positions initiales des marqueurs vers les positions des marqueurs retrouvées. La deuxième mesure

$$\left|\cos^2\theta + \sin^2\theta - 1\right| \tag{5.3}$$

estime la "rigidité" de la transformation. En effet, la transformation résultant du déplacement de l'objet étant rigide, la composante de rotation de la matrice retrouvée se doit

de préserver cette rigidité.

Malgré tout, il se peut que la position retrouvée de certains des marqueurs en 2D soit erronée. Si cela se produit, la position erronée entraînera une incohérence dans le système d'équations qui se caractérisera soit par une matrice de transformation non rigide, soit par une matrice de transformation qui ne transformera pas correctement tous les points. Ces deux cas étant détectables par nos mesures d'erreur, nous résolvons ce problème en utilisant des sous-ensembles des marqueurs disponibles. Notre algorithme tente de trouver la transformation qui produira l'erreur minimale (somme des deux mesures) en itérant 500 fois avec un sous-ensemble aléatoire de 5 à n points parmi les n marqueurs visibles. Finalement la matrice entraînant la plus faible erreur est retenue comme matrice de transformation du piédestal. Ainsi, même si un ou plusieurs marqueurs sont mal retrouvés en 2D dans la caméra, ils n'altèrerons pas la matrice résultante. De plus, pour calculer les bonnes positions 2D des marqueurs perdus lors du suivi temporel, il suffit de transformer les positions 3D initiales par la matrice de transformation du piédestal et de reprojeter les positions résultantes dans la caméra.

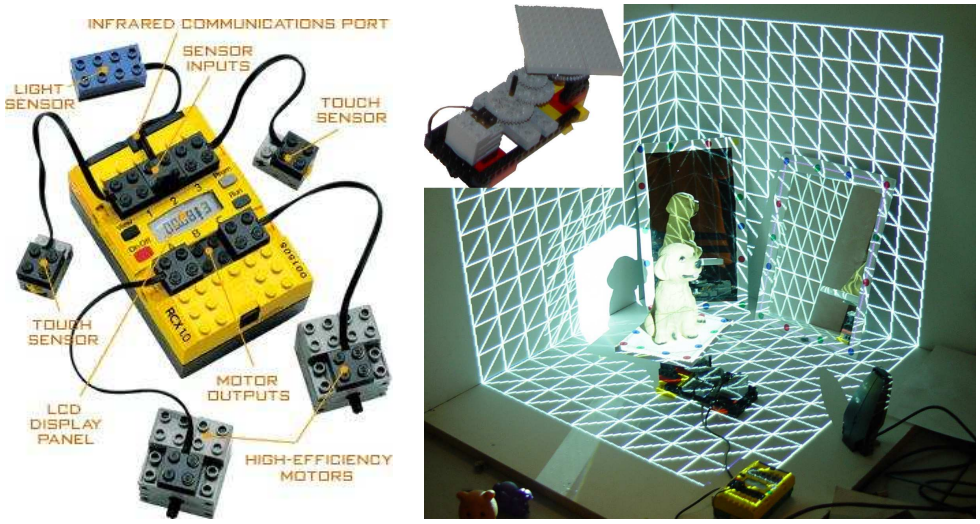
(a) (b) (c)

FIG. 5.2 – Suivi temporel du piédestal. (a) La boîte englobante reposant sur la base du piédestal est illuminée ; (b) seulement les marqueurs de couleur suivis temporellement sont illuminés ; (c) autre position suivie du piédestal.

Lors du suivi temporel, la recherche 2D dans la caméra peut être facilitée si le projecteur projette uniquement sur les marqueurs de couleur un carré illuminé. En même temps, cela offre à l'usager un retour visuel en lui indiquant si le système suit toujours correctement le piédestal. Les marqueurs se situant à l'arrière du volume englobant de l'objet, selon le point de vue du projecteur, ne sont pas illuminés pour éviter d'illuminer l'objet se trouvant sur le piédestal. Ainsi, seuls les marqueurs de couleur illuminés sont visibles pour la caméra, ce qui facilite beaucoup le suivi 2D dans l'image et donc le suivi en 3D.

5.1.3 *LEGO*^{TM}

Pour faciliter le processus de changement d'orientation de l'objet, nous avons développé une version automatique du piédestal. Nous avons utilisé les jouets *LEGO MINDSTORMS* [Min] qui permettent de construire facilement des petits robots électroniques. Le concept est le même que les *LEGO*^{TM} traditionnels ; il suffit simplement d'emboîter ensemble des blocs, des roues et des engrenages pour réaliser toutes sortes de constructions architecturales et mécaniques. La nouveauté réside dans certains blocs électroniques spéciaux : moteurs, capteurs de lumière et boutons poussoirs. Un bloc contrôleur particulier, le RCX (Figure 5.3 (a)), est constitué de plusieurs bornes d'interface reliées à un micro-processeur Hitachi 8300 possédant des batteries, de la mémoire et un émetteur/récepteur infrarouge qui permet le contrôle à distance par un ordinateur (ou une télécommande). Ce contrôleur a originalement été conçu pour pouvoir être programmé dans le langage propriétaire de *LEGO*^{TM} proche de l'assembleur. Grâce à certains logiciels libres (*open-source*) tels que les projets BrikOS [Ntbt] et Lejos [ASG^{+}], il est possible d'écrire des programmes dans des langages de plus haut niveau tels le *C++* et le *Java*.



(a) (b)

FIG. 5.3 – (a) Contrôleur RCX pour les *LEGO MINDSTORMS* (image tirée de kauts.mine.nu/rcx.html) ; (b) piédestal déplacé automatiquement et construit à partir des *LEGO MINDSTORMS*.

Grâce aux *LEGO MINDSTORMS* et au projet *Lejos*, nous avons développé une

version automatique du piédestal, qui est contrôlée directement par notre système de reconstruction (Figure 5.3 (b)). Pour trouver l'équation du plan de ce piédestal, nous procédons de manière similaire à l'estimation de la pose d'un miroir. Les positions des marqueurs de couleur dans la caméra sont initialisées par l'usager comme pour le piédestal normal. Une itération de lumière structurée est ensuite exécutée sur la scène. Après analyse des images de lumière structurée, tous les pixels se situant à l'intérieur du polygone défini par les marqueurs sélectionnés dans la caméra sont utilisés pour reconstruire les points 3D de la surface du piédestal. Ces points sont utilisés pour estimer le plan de support du piédestal par PCA, de la même manière que pour le miroir. La normale du plan retrouvé a généralement moins d'un degré d'écart avec le vecteur vertical (0, 1, 0). Le but de ce processus est principalement d'évaluer la hauteur du plan ($Y = d$).

La rotation du piédestal peut être commandée à tout moment soit par l'usager ou soit automatiquement par le système en mode "reconstruction continue". Dans ce mode, le système reconstruit l'objet automatiquement en procédant de la manière suivante :

1. Projeter les motifs pour compléter une itération de reconstruction ;
2. Faire glisser le code d'un pixel dans le projecteur et recommencer l'étape 1 jusqu'à ce que tous les glissements possibles soient effectués ;
3. Lancer le suivi temporel du piédestal, faire tourner l'objet de quelques degrés puis recommencer à l'étape 1.

Ce mode permet de reconstruire un objet sans avoir à intervenir dans le processus. L'usager peut, après avoir initialisé le système, déposer un objet sur le piédestal automatique, lancer la "reconstruction continue", puis laisser le système procéder pendant le temps désiré. Généralement, deux à trois minutes suffisent pour compléter deux rotations complètes de l'objet. Bien sûr, certains angles de vue restent inaccessibles, soit pour le projecteur soit pour la caméra, et les miroirs peuvent ici être utiles pour aller chercher les points de vue nécessaires. Des résultats de reconstruction avec le piédestal automatique et un miroir sont donnés à la Section 6.6.

5.2 Utilisation interactive de miroirs

5.2.1 Philosophie

Un aspect intéressant de l'utilisation des miroirs est la flexibilité qu'ils peuvent apporter à un système de reconstruction. Dans le cas de grandes statues ou d'environnements architecturaux, la solution de déplacer et tourner l'objet devient inappropriée ou même impossible. Par exemple le projet *The Digital Michelangelo Project* [LPC+00], réalisé par 30 professeurs et étudiants gradués de l'université Stanford et l'université de Washington, consistait à reconstruire au cours d'une année des sculptures et des architectures réalisées par Michelange. Les sculptures de Michelange étant grandes, lourdes et de très grande valeur, il est évident que la solution de les déplacer est impraticable. Ce projet a donc nécessité la construction d'un numériseur laser à balayage fait sur mesure par *Cyberware* [Cyb]. Monté sur un bras robotisé très précis (Figure 5.4), ce numériseur peut être déplacé dans différentes positions et orientations. Il est ainsi possible de connaître précisément les paramètres de position et d'orientation de la tête du numériseur à chaque itération de la numérisation, ce qui rend possible l'alignement des résultats pris sous différents points de vue. Cet appareil de très grande qualité est, en contrepartie, lourd, encombrant et coûteux.

(a) (b)

FIG. 5.4 – (a) Numériseur laser à balayage motorisé développé par *Cyberware* ; (b) tête du numériseur composée d'une source laser, d'une caméra vidéo pour la numérisation 3D, d'une source de lumière à fibre optique et d'une caméra couleur haute définition.

Dans un tel contexte (objet fixe), nous pensons qu'un miroir déplacé manuellement (ou automatiquement) et suivi temporellement dans l'espace pourrait permettre d'obtenir des résultats possiblement similaires à bien moindre coût et ce, avec une grande flexibilité. En effet, le système de reconstruction peut ainsi rester fixe pendant que le miroir est déplacé autour de l'objet pour obtenir les points de vue nécessaires à la reconstruction complète de ce dernier. Cette philosophie s'applique d'autant plus si l'environnement dans lequel se trouve l'objet est contraint en espace ou en accessibilité (*e.g.* intérieur d'une caverne ou d'une pyramide).

5.2.2 Initialisation

Le suivi d'un miroir nécessite une étape d'initialisation identique à l'estimation de pose d'un miroir fixe (Section 4.3). Après la sélection des marqueurs de couleur dans la caméra, les pixels dans un voisinage de 5×5 pixels autour de chaque marqueur sont utilisés pour créer les ellipsoïdes englobants représentant ces marqueurs. Ainsi comme pour le piédestal, il est possible de suivre en 3D le déplacement des marqueurs dans la caméra. Après l'estimation du plan de support du miroir Π_{mir} par lumière structurée, les positions 3D initiales des marqueurs de couleur sont obtenues en calculant l'intersection de ce plan Π_{mir} avec chaque rayon généré dans la caméra à partir des positions 2D de ces marqueurs.

Ainsi pour chaque marqueur de couleur, nous connaissons après l'initialisation : son ellipsoïde englobant RGB, sa position 2D dans la caméra et sa position 3D initiale.

5.2.3 Suivi temporel 3D

Pour suivre en 3D le miroir dans l'espace, nous avons développé un algorithme itératif qui consiste à projeter continuellement les motifs de lumière structurée sur le miroir. Les marqueurs du miroir sont suivis en 2D dans la caméra grâce aux ellipsoïdes englobants. Après chaque itération de lumière structurée, les motifs sont analysés sur le contour du miroir (défini par les marqueurs 2D) et un nouveau plan de support Π_{mir} est estimé par PCA. Les nouvelles positions 3D des marqueurs sont retrouvées en utilisant les positions 2D suivies dans la caméra et le nouveau plan Π_{mir}. À partir des positions 3D des marqueurs suivis et de leurs positions initiales, une matrice de transformation 3D est calculée permettant d'exprimer le déplacement du miroir par rapport à sa configuration

initiale. Ainsi si un marqueur n'est pas suivi correctement en 2D, sa nouvelle position 3D est retrouvée à partir de sa position 3D initiale et de la matrice de transformation calculée.

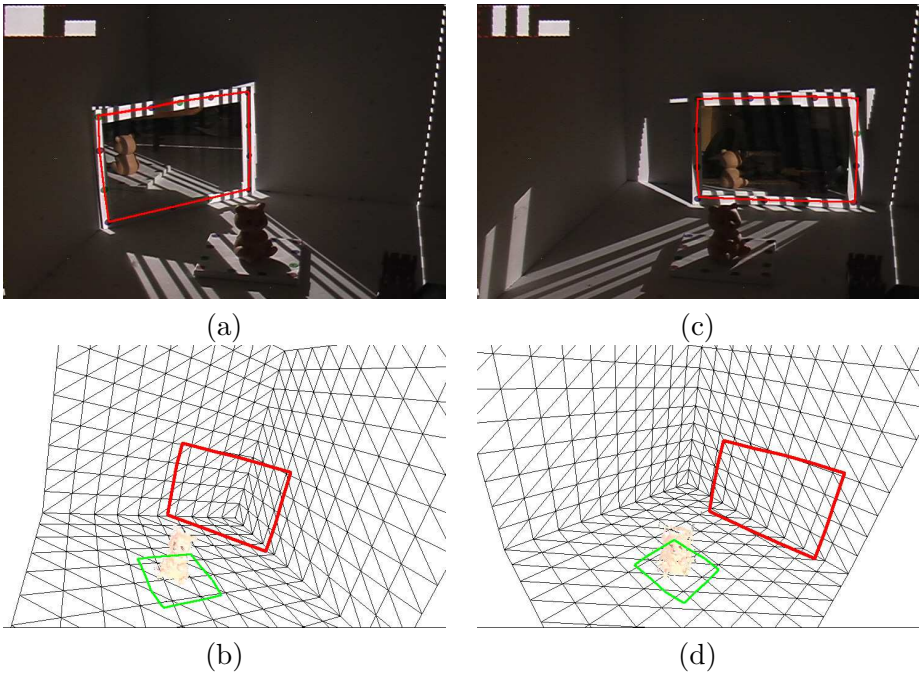


FIG. 5.5 – Suivi temporel 3D d'un miroir déplacé manuellement : (a) pose initiale du miroir lors de l'estimation de pose affichée par-dessus l'image caméra et (b) vue synthétique dans notre système ; (c) pose après déplacement manuel du miroir suivi automatiquement en 2D et en 3D par le système et (d) équivalent synthétique.

Notre implémentation actuelle du suivi interactif des miroirs est limitée au déplacement d'un seul miroir à la fois. L'usager déplace le miroir, puis indique au système de passer en mode reconstruction pour utiliser le nouveau point de vue fourni par le miroir. L'usager peut déplacer ainsi le miroir à volonté pour aller chercher de nouveaux points de vue. Le résultat d'une telle reconstruction est illustré à la Figure 6.4.

Le suivi des miroirs est un outil qui se révèle très utile lorsque l'on veut aller chercher des points de vue particuliers sans avoir à ré-estimer manuellement la pose du miroir. Cependant, dans notre implémentation actuelle, si l'usager déplace trop rapidement le miroir, il se peut que le système "perde" le suivi du miroir. Dans ce cas, il est nécessaire de refaire l'initialisation manuelle du miroir. Ce désagrément pourrait être facilement résolu si l'on remplaçait les marqueurs de couleur par des diodes luminescentes ou avec

l'utilisation d'un système de suivi magnétique [Asc].

5.3 Intégration et post-traitement du modèle final

Après chaque itération de reconstruction du modèle, nous obtenons généralement entre 500 et 10,000 points 3D provenant de la surface du modèle à reconstruire (à l'intérieur de la boîte englobante). Chaque point est caractérisé par une position dans l'espace et une couleur. À la fin de la reconstruction, nous devons donc intégrer ensemble les résultats de toutes ces itérations de reconstruction. Parfois, de faibles erreurs d'estimation de pose du miroir ou du piédestal créent de légers désalignements (position et rotation) entre les différents groupes de points reconstruits. Pour régler ce problème nous utilisons un algorithme d'alignement de points appelé ICP (*Iterative Closest Point*) [BM92]. Cet algorithme est communément utilisé pour aligner des modèles 3D en ne se basant que sur la géométrie, ou optionnellement, sur les couleurs des modèles. Le but de cet algorithme est de trouver itérativement, à partir d'un estimé initial, la transformation rigide qui permet d'aligner ensemble deux groupes de points sur leurs zones de recouvrement. Cette transformation est raffinée itérativement en générant des paires de points correspondants entre les deux modèles puis en minimisant une mesure d'erreur entre toutes les paires de points.

L'algorithme ICP fonctionne ainsi :

1. Sélectionner un sous-ensemble de points dans un des modèles ou dans les deux ;
2. Pour chaque point du sous-ensemble, sélectionner un point correspondant dans l'autre modèle. Ceci est généralement fait en prenant le point le plus proche appartenant à l'autre modèle ou éventuellement en utilisant un critère supplémentaire (couleur, normale) ;
3. Associer un poids à chaque paire de points correspondants, basé sur la distance entre les points, la compatibilité des normales ou la similarité des couleurs ;
4. Éliminer les paires de points invalides, soit séparées par une trop grande distance, soit selon un autre critère ;
5. Assigner une mesure d'erreur à l'alignement actuel basé sur les différentes paires de points. Généralement la distance euclidienne entre les paires de points est aussi utilisée ici. D'autres mesures peuvent être utilisées [CM92] ;

6. Minimiser la mesure d'erreur par transformation rigide, retrouvée à l'aide des moindres carrés.

Cette description de l'algorithme ICP est tirée de Rusinkiewicz *et al.* [RHHL02] et est plus précisément détaillée ailleurs [RL01].

Une variante de cet algorithme, l'alignement global, consiste à aligner plusieurs groupes de points selon le même principe. Nous utilisons l'implémentation de Pulli [Pul99] disponible dans le programme *Scanalyze* [Sta] distribué gratuitement par l'université Stanford. Les points provenant des différentes itérations de reconstruction sont ainsi alignés globalement en post-traitement. Cette étape prend généralement entre 20 secondes et 5 minutes selon le nombre d'itérations et le nombre de points reconstruits à chaque itération.

Une fois les différents groupes de points alignés et intégrés en un nuage de points (Figure 5.6 (a),(b)), l'usager peut éliminer les points isolés (*outliers*) à l'aide de deux paramètres. Tout point qui possède moins de n points voisins à l'intérieur d'une distance d donnée est éliminé. Ce processus utilise un "arbre kd" (*kd-tree*) pour accélérer la recherche des voisins de chaque point et ne nécessite que quelques secondes à l'usager pour choisir correctement les valeurs des deux paramètres. Le résultat de l'élimination des points isolés est illustré à la Figure 5.6 (c),(d). Par la suite, un filtre médian est appliqué sur le nuage de points pour éliminer le bruit résiduel présent à surface du modèle. Ce bruit est dû à un manque de précision de la localisation des frontières de bandes d'illumination lors de l'analyse des images. Ce manque de précision est, selon nous, induit par l'aliassage lié aux différences de résolution entre la caméra et le projecteur et à la compression MPEG du signal de notre caméra. Le filtre médian que nous avons développé se résume ainsi pour chaque point P du modèle :

1. Identifier les points V_P les plus proches de P ;
2. Trouver par PCA l'équation du plan Π_P qui passe par P et qui approxime le mieux les points voisins V_P ;
3. Calculer la médiane des distances entre chacun des points voisins V_P avec le plan Π_P ;
4. Déplacer P de cette distance médiane le long de la normale au plan Π_P.

Le résultat de cette étape de filtrage est illustré à la Figure 5.6 (e),(f). Le nombre de points voisins utilisé est un paramètre spécifié par l'usager.

Finalement, les couleurs du modèle obtenu sont filtrées à l'aide d'un filtre moyen sur le voisinage. Les étapes du filtrage des couleurs sont identiques à celles du filtrage médian à l'exception que pour chaque point, la moyenne des couleurs du voisinage est calculée et cette couleur est directement associée au point (Figure 5.6 (g),(h)).

Une fois le modèle filtré, il est nécessaire, comme nous l'illustrerons dans la Section 5.4, de calculer une normale pour chaque point reconstruit afin d'obtenir un rendu de qualité. La technique généralement utilisée pour assigner une normale aux points reconstruits, consiste à calculer par PCA un plan moyen approximant la surface composée des points voisins. La normale de ce plan est associée au point car elle représente localement l'orientation de la surface.

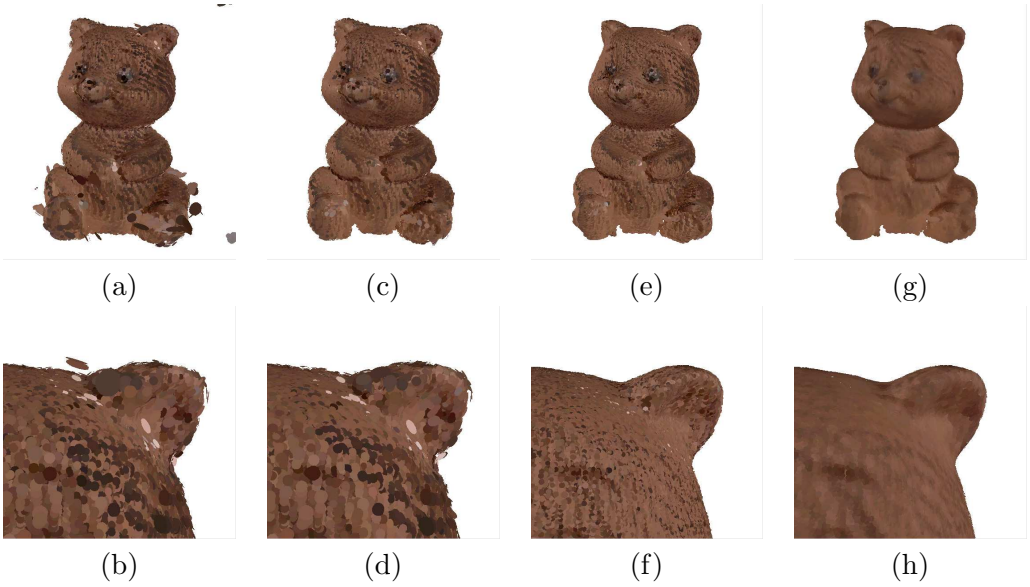


FIG. 5.6 – Traitement du modèle de l'ourson, les images du haut illustrent l'évolution du modèle après chaque étape de filtrage et les images du bas sont des gros plans correspondants. Chaque point est affiché avec un disque de couleur. (a),(b) Modèle après alignement global ; (c),(d) après élimination des points isolés ; (e),(f) après filtrage médian ; (g),(h) après filtrage des couleurs.

En infographie, la représentation polygonale est une des plus communes pour l'affichage et le traitement logiciel des modèles. Plusieurs techniques permettent de transformer un nuage de points ou des cartes de profondeurs en un modèle polygonal [PDH+97, DMGL02]. Cependant la représentation par *splats* (ou *surfels* : éléments de surface) a récemment connu beaucoup de progrès avec plusieurs techniques de rendu efficaces [RLat]

et de qualité [RPZ02, ABCO$^+$01] ainsi que de multiples outils d'édition [ZPKG3d] et de simplification [PGK02] de modèles en points. De plus, cette représentation est justifiée car la plupart des systèmes de reconstruction par triangulation créent des modèles sous forme de nuages de points. Ainsi, nous avons décidé de conserver cette représentation pour l'affichage plutôt que de faire une conversion en modèle polygonal.

5.4 Affichage du modèle

La visualisation du modèle final joue un grand rôle dans l'appréciation du résultat d'une numérisation 3D. Notre système supporte plusieurs types d'affichage différents, tous basés sur la représentation par points.

Pendant de la reconstruction, l'usager doit avoir un retour visuel de l'état actuel de la reconstruction. Pour que l'affichage de ce modèle soit efficace, nous affichons de simples points OpenGL avec, pour chacun, une couleur extraite de "l'image de référence" sans aucune fonction d'illumination (*shading*). La taille (en pixels) pour l'affichage des points est choisie par l'usager.

Une fois la reconstruction terminée et le modèle filtré, une normale est calculée pour chacun des points reconstruits. Le modèle est alors affiché en utilisant le modèle d'illumination locale de *Phong* (Figure 5.7).

Pour un rendu de meilleure qualité, le modèle peut aussi être affiché sous forme de *splats*. Cette représentation nécessite quatre paramètres pour chaque *splat* : une position, une normale, une couleur et une taille pour l'affichage. La taille utilisée pour chaque *splat* est la distance euclidienne avec son 7ème voisin le plus proche. La recherche des points voisins est, comme lors du calcul des normales, accélérée à l'aide d'un *arbre kd*. L'affichage en *splats* peut se faire, au choix, dans les programmes *QSplat* [RLat] ou *PointShop3D* [ZPKG3d], tous deux dont le code source est disponible. Notre modèle peut être converti dans le format de fichier de *QSplat* ou dans celui de *PointShop3D*.

L'avantage principal de *Qsplat* est sa capacité à afficher d'énormes modèles (plus de 500,000 *splats*) en temps réel, grâce à une structure de données optimisée basée sur une hiérarchie de sphères.

Le programme *PointShop3D* quant à lui offre un plus grand nombre de possibilités. Beaucoup de ses outils sont disponibles pour traiter et modifier des modèles représentés par points : filtrage gaussien de points, filtrage de couleurs, élimination de points isolés,

etc. Plusieurs types de rendu sont aussi fournis tels que : un rendu logiciel (CPU) de qualité, un rendu de *splats* EWA (*hardware EWA surface splatting*) [RPZ02] très efficace et de bonne qualité.

Généralement nous utilisons *QSplat* pour visualiser rapidement un modèle reconstruit et en observer les détails. Puis nous utilisons *PointShop3D* pour obtenir des images finales de qualité.

(a) (b) (c)

FIG. 5.7 – Affichage du modèle : (a) modèle affiché sous forme de points après traitement ; (b) modèle affiché en *splats* par *Qsplat* [RLat] ; (c) modèle affiché en *splats* par *PointShop3D* [ZPKG3d].

5.5 Système d'optimisation

Une des tâches les plus fastidieuses de notre système réside dans le placement des différents périphériques et dans l'initialisation des outils intervenant dans la reconstruction. En pratique, plusieurs contraintes s'appliquent à chaque périphérique. La caméra et le projecteur doivent être suffisamment distants pour que les "rayons" et les "plans" qu'ils génèrent dans l'espace s'intersectent avec un angle supérieur à 20 degrés, faute de quoi la stabilité numérique réduite des opérations d'intersection produit des points moins précis. La caméra doit avoir une vue en plongée sur le piédestal pour bien suivre temporellement les marqueurs de couleur. La boîte englobante de l'objet doit être visible depuis le point de vue de la caméra, du projecteur, et depuis leurs points de vue réfléchis dans chaque miroir présent dans le système. Cette dernière condition est de

loin la plus difficile à satisfaire manuellement.

Pour faciliter le placement des miroirs dans le système, nous avons développé un algorithme d'optimisation. Le but est ici d'utiliser la puissance de calcul du CPU et du GPU (*Graphic Processing Unit*) pour identifier, par simulation, une configuration optimale. Ainsi le système peut conseiller l'usager sur une disposition plus efficace pour la reconstruction en beaucoup moins de temps que celui-ci prendrait par "essais, et erreurs". Le problème à résoudre par cet algorithme d'optimisation s'énonce comme suit : *Étant donné une disposition de la caméra, du projecteur et du piédestal, et pour un nombre de miroirs donné, comment placer ces miroirs pour maximiser la couverture de l'objet, à la fois dans la caméra et dans le projecteur.* Dans notre implémentation, la métrique à maximiser est le nombre de pixels de la boîte englobante visibles directement et indirectement (dans un miroir) à la fois dans la caméra et dans le projecteur. Étant donnée une certaine disposition des miroirs, nous utilisons la carte graphique et OpenGL pour calculer les images rendues du point de vue de la caméra et du projecteur, avec une couleur d'identification particulière pour les pixels de la boîte englobante (Figure 5.8 (b)). Ces images sont ensuite récupérées en mémoire par le système, puis analysées pour comptabiliser les pixels valides. Le processus de calculer les masques pour la caméra et le projecteur, les récupérer de la carte graphique et analyser le nombre de pixels valides prend au total moins de 0.03 seconde.

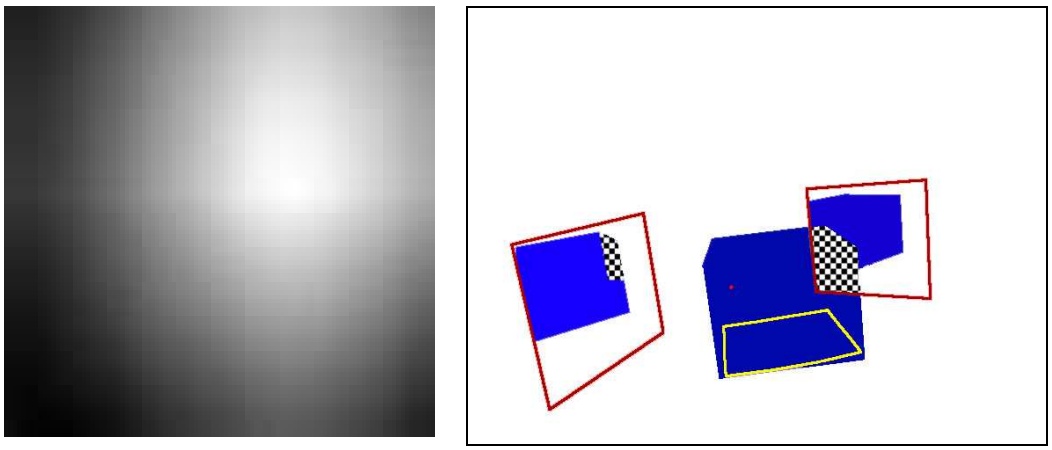
Pour trouver un placement optimal des miroirs, le système peut donc essayer des milliers de dispositions différentes (choisies aléatoirement ou régulièrement) et identifier celle qui offre la couverture maximale. Notre implémentation essaie simplement différentes positions pour deux miroirs et ce, le long des deux plans verticaux de notre environnement de reconstruction. Une image qui représente les résultats des configurations testées lors de cette optimisation est fournie à la Figure 5.8 (a). Dans cette image, chaque pixel correspond à une disposition particulière des deux miroirs et son intensité indique le nombre de pixels valides couverts par la boîte englobante de l'objet à reconstruire. Après avoir calibré la caméra, le projecteur et initialisé le piédestal, le système d'optimisation est lancé. Quelques minutes de calculs plus tard (4.5 minutes pour l'optimisation illustrée) le système indique à l'usager comment placer les miroirs (Figure 5.8 (b)).

Bien sûr, cette implémentation n'est qu'une preuve de concept en deux dimensions

(deux miroirs déplacés chacun sur une ligne) de l'utilité d'un algorithme d'optimisation dans un tel système de reconstruction. Il serait possible d'imaginer un algorithme d'optimisation beaucoup plus "intelligent" et flexible qui permettrait de déterminer le placement optimal de toutes les composantes du système et une mesure d'erreur qui prendrait en compte toutes les contraintes exprimées précédemment.

Cependant, la recherche d'une disposition optimale est loin d'être triviale. La pose de chacun des miroirs peut être exprimée en 6 dimensions (3 positions et 3 orientations). Les poses de la caméra et du projecteur peuvent être exprimées en 7 dimensions (3 positions, 3 rotations et le facteur *zoom*). La pose du piédestal peut être exprimée en 3 dimensions (2 déplacements et 1 rotation sur le plan). Ainsi optimiser le placement de deux miroirs et des autres éléments du système implique une recherche dans un espace à 29 dimensions. Cette recherche apparaît clairement infaisable, mais en pratique l'usager place aisément la caméra, le projecteur et l'objet de manière à respecter la contrainte de distance entre la caméra et le projecteur et la contrainte sur la vue en plongée des marqueurs de couleur du piédestal. Le placement des miroirs est de loin la tâche la plus fastidieuse. On peut donc limiter la recherche du système d'optimisation au placement des miroirs. Pour deux miroirs, l'espace de recherche est donc réduit à 12 dimensions.

D'autre part, la métrique à maximiser est non-linéaire et les domaines réalisables peuvent être très "fins", car comme nous le constatons dans nos expérimentations, une faible rotation sur un miroir peut changer complètement sa couverture dans la reconstruction. Ainsi, certaines méthodes standard comme "diviser pour régner" ne fonctionneraient pas efficacement pour résoudre ce problème. Nous pensons qu'une méthode de recherche aléatoire avec raffinement autour des meilleures solutions pourrait donner des résultats satisfaisants. Toutefois, même si une bonne solution est obtenue, il semble impossible de déterminer si celle-ci est "la" meilleure ou non. Mais est-ce si important d'identifier la meilleure solution ?



(a) (b)

FIG. 5.8 – (a) Image représentant la couverture des configurations testées pour le placement de deux miroirs carrés de 5×5 pouces en fonction de la couverture qu'ils apportent ; chacun des deux miroirs est déplacé le long du plan de l'environnement contre lequel il repose ; l'axe des abscisses représente le déplacement du miroir de droite et l'axe des ordonnées représente le déplacement du miroir de gauche ; (b) positionnement optimal indiqué par le système d'optimisation ; une couleur spéciale est assignée aux pixels de la boîte englobante observée directement ou dans un miroir.

Chapitre 6

Résultats

6.1 Implémentation

L'implémentation actuelle de notre système utilise une caméra *Panasonic PVGS-70* qui transmet à l'ordinateur ses images entrelacées en 720×480 pixels. Ces images sont transmises à une fréquence de 29.97 images par seconde à l'aide du protocole IEEE 1394 (*firewire*). La communication avec le pilote (*driver*) IEEE 1394 est assurée par le programme *DVGRAB* [Sch], dont le code source est disponible gratuitement. Nous avons modifié la version 1.2 de ce programme pour en faire une librairie statique directement intégrée dans notre système. Notre projecteur DLP *Compaq MP4800* émet des images d'une résolution de 1024×768 pixels avec 24 bits de couleur par pixel. Ce projecteur a un taux de rafraîchissement de 60 images par seconde et une puissance maximale de 2100 lumens.

L'ordinateur sur lequel s'exécute notre système de reconstruction possède deux processeurs Pentium IV Xeon 2.4 GHz, 1 Giga-octet de mémoire vive et une carte graphique *nVidia GeForce-4 Ti4200*. Chaque processeur Xeon a la particularité d'avoir une partie de son *pipeline* interne dédoublée, ce qui virtuellement dédouble chaque processeur. Cette particularité est très intéressante pour le parallélisme des applications car ainsi notre ordinateur possède virtuellement quatre processeurs. Pour tirer avantage de ces multiples processeurs, notre système est scindé en différentes tâches parallélisées. Un processeur est dédié uniquement à l'acquisition des images de la caméra. Un deuxième s'occupe de l'analyse du code de synchronisation dans chaque image, du stockage des lignes paires et impaires de chacune de ces images et de l'émission des motifs de lumière

structurée par le projecteur. Finalement, un troisième processeur est chargé de la reconstitution des images entrelacées, du décodage des motifs de lumière structurée et de la reconstruction des points 3D.

Nous avons mis en place deux environnements de reconstruction de différentes dimensions. Le plus grand mesure $81 \times 61 \times 63$ *cm* et est principalement utilisé pour les objets de grandes dimensions, ou lorsque beaucoup d'espace est nécessaire (déplacement interactif des miroirs, piédestal automatique et miroir). L'autre environnement de reconstruction est chronologiquement le deuxième à avoir été mis en place et les points de calibration y ont été indiqués plus précisément. Il mesure exactement $10 \times 10 \times 10$ *pouces* et permet une calibration plus précise pour la reconstruction des petits objets.

Les temps approximatifs pour les différentes étapes d'initialisation du système sont donnés au Tableau 6.1.

Étape d'initialisation	Temps approximatif
Calibration manuelle du projecteur	2 min.
Calibration manuelle de la caméra	1 min.
Initialisation du piédestal	20 sec.
Sélection et estimation de la pose d'un miroir	30 sec.

TAB. 6.1 – Temps approximatifs des étapes d'initialisation du système.

Nous avons expérimenté avec un grand nombre d'objets et de configurations. La Figure 6.1 en montre quelques-uns représentatifs. Dans les sections qui suivent, nous présentons plusieurs résultats de configurations différentes afin d'en analyser les avantages et les inconvénients. Pour tous ces résultats de reconstructions, nous avons utilisé notre implémentation du code des frontières de bandes pour les motifs projetés avec les modifications que nous y avons apportées pour l'utilisation des miroirs. Les motifs verticaux seulement sont projetés, et le décodage s'effectue en parcourant les lignes de pixels verticales et horizontales.

Bien qu'il serait très intéressant d'analyser quantitativement la précision des points reconstruits par notre système, les contraintes de temps pour ce projet de maîtrise ne nous ont pas permis de nous lancer dans ces investigations. Nous nous sommes contentés d'une analyse qualitative basée sur l'appréciation visuelle de nos résultats.

L'affichage des modèles dans les figures de ce chapitre est fait avec *QSplat* et *PointShop3D*. Dans ces deux logiciels, la taille des points affichés est inversement proportion-

nelle à la densité du modèle autour de ce point. Ainsi de plus gros *splats*, de couleur uniforme, sont discernables dans les régions des modèles où moins de points sont reconstruits. De plus, comme le miroir atténue en partie la lumière du projecteur et que nous ne traitons pas ce phénomène, l'ajout de points de différents miroirs produit des zones de couleurs différentes dans le modèle reconstruit.

FIG. 6.1 – Photographies de quelques objets que nous avons utilisés pour la reconstruction. On peut observer que le tigre, le globe terrestre et les yeux du chiot réfléchissent la lumière de manière spéculaire.

6.2 Trois miroirs fixes et objet fixe

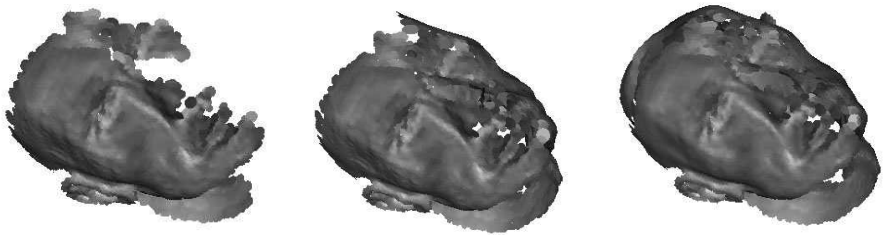


FIG. 6.2 – Reconstruction statique de la tête : tous les éléments du système de reconstruction restent fixes, seul le nombre de miroirs utilisés change. Gauche : aucun miroir. Centre : un miroir ; Droite : trois miroirs.

Tout d'abord, nous illustrons la différence de couverture que procurent les miroirs à l'intérieur d'une configuration entièrement statique pour reconstruire une tête en argile grise (Figure 6.2).

Le côté droit de la tête est assez bien reconstruit sans miroir, mais comme le côté gauche est moins visible dans la caméra, peu de points y sont reconstruits. En posant un miroir derrière la tête, ces points sont plus densément reconstruits. Deux autres miroirs améliorent la reconstruction du haut de la tête.

Le Tableau 6.2 montre les temps approximatifs de reconstruction, le nombre de points reconstruits, et le nombre d'itérations de lumière structurée. Une itération correspond à une séquence de 5 images de motifs utilisées pour la reconstruction, plus une image de référence. Il est à noter dans ce cas-ci que les motifs projetés ont été "glissés" à quelques reprises afin de générer plus de points sur le modèle.

Nombre de miroirs	Temps de reconstruction	Nombre de points	Nombre d'itérations
0	15 sec.	16,000	14
1	26 sec.	21,000	18
3	36 sec.	28,000	28

TAB. 6.2 – Statistiques pour la statue de la tête.

6.3 Trois miroirs fixes et objet sous multiples poses

FIG. 6.3 – Reconstruction de la tête avec trois miroirs. L'objet a été déplacé manuellement avec le piédestal et reconstruit selon quatre poses différentes.

En utilisant trois miroirs comme illustré à la Figure 4.5, la tête d'argile grise est déplacée selon quatre différentes poses à l'aide du piédestal. Tel qu'illustré à la Figure 6.3, quatre poses et trois miroirs ont été suffisants pour remplir la plupart des trous présents dans le modèle si aucun miroir n'avait été utilisé. Les différences de cou-

leurs dans certaines régions du modèle sont aussi dues à l'illumination changeante du projecteur lorsque l'objet est déplacé.

6.4 Miroir déplacé interactivement et objet fixe

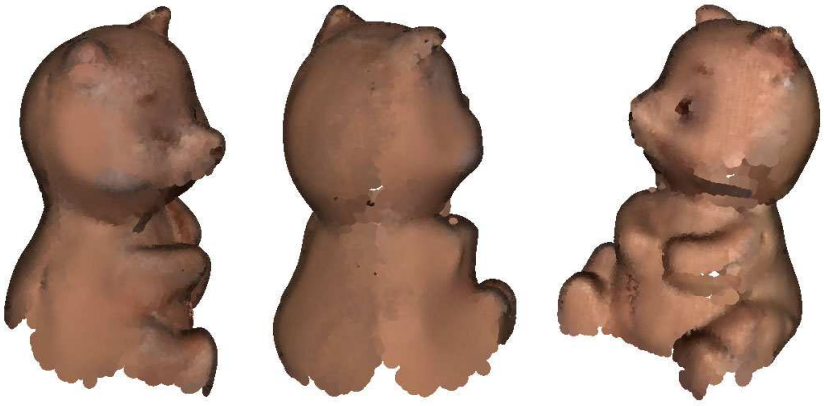


FIG. 6.4 – Ces images montrent les résultats de reconstruction d'un objet fixe avec un miroir déplacé interactivement. Temps total de reconstruction : 10 minutes ; temps total de post-traitement : 5 minutes ; définition : 56,000 points.

Cette section illustre la reconstruction d'un objet fixe à l'aide d'un miroir mobile. L'objet reste fixe tout au long de la reconstruction et seul le miroir est déplacé manuellement autour de l'objet pour obtenir de nouveaux points de vue virtuels. L'usager alterne entre le mode suivi temporel et estimation de pose du miroir, et le mode reconstruction de l'objet avec miroir. Ainsi, l'usager déplace le miroir jusqu'à obtenir un point de vue satisfaisant, puis il demande au système de reconstruire pour cette configuration. Le passage d'un mode à l'autre est fait à l'aide d'une seule touche sur le clavier de l'ordinateur. L'objet illustré à la Figure 6.4 a été reconstruit entièrement sans être déplacé à l'aide d'un seul miroir en 15 minutes. Une fois encore, les couleurs captées n'étant pas traitées, on peut observer des jonctions de couleurs différentes le long du milieu du dos dans l'image centrale.

Ce résultat illustre que l'utilisation de miroirs pour la reconstruction d'objets fixes est possible. Cependant, on peut observer sur l'image de droite de la Figure 6.4 que quelques zones restent difficilement accessibles pour un seul miroir, et peu de points sont reconstruits sous le bras gauche et autour du cou. En effet, lorsque le miroir se

retrouve à des angles trop rasants par rapport au projecteur ou à la caméra, il devient plus difficile de bien estimer sa pose. Il est possible d'imaginer l'utilisation simultanée de deux miroirs en exploitant l'interréflexion pour extraire de l'information dans ces zones inaccessibles. Cependant, nous n'avons jamais essayé d'estimer la pose d'un miroir au travers d'un autre miroir et l'accumulation d'erreurs pourrait éventuellement poser problème dans ce cas.

6.5 Piédestal automatique sans miroir

Les résultats de reconstructions présentés ici ont été réalisés sans intervention de l'usager lors de la reconstruction. L'objet est déposé sur le piédestal automatique et le système de reconstruction est lancé en mode "reconstruction continue". L'ourson illustré à la Figure 6.5, est un de nos modèles les mieux reconstruits. Il a été reconstruit sans intervention de l'usager en 3 minutes puis filtré en moins de 3 minutes (filtrage médian et filtrage de couleur). Les quelques artefacts résiduels (points isolés) encore présents ont été éliminés manuellement dans *PointShop3D* [ZPKG3d] en moins de 5 minutes. On peut malgré tout encore observer qu'il manque des points sur le bout museau (sombre, qui réfléchit peu la lumière) et sur la patte inférieure droite du modèle.

Les résultats des reconstructions du tigre et du globe terrestre sont montrés aux Figures 6.6 et 6.7, respectivement. Le tigre illustre la robustesse de la reconstruction en présence d'une texture visuellement importante. Les rayures noires sur le flanc du tigre poseraient problème avec l'algorithme de décodage de Rusinkiewicz *et al.* [RHHL02]. De plus comme ces rayures sont noires, elles ne réfléchissent que très peu la lumière reçue. Leur présence dans le modèle final est due à leur spécularité. En effet, les surfaces du globe terrestre et du tigre sont spéculaires et ne réfléchissent suffisamment la lumière que sous certains angles particuliers. Le piédestal automatique a permis d'obtenir un grand nombre d'angles de vue tout autour de ces deux objets pour les reconstruire. Le globe terrestre illustre particulièrement la reconstruction en présence d'une texture fine complexe. Le flou observable sur cette texture est causé par le filtrage spatial des couleurs. La queue du tigre, bien visible dans l'image en bas à gauche de la Figure 6.6, est très étroite (environ 1 cm) par rapport à la distance du projecteur (environ 3 mètres). Elle est assez bien reconstruite considérant la difficulté d'y détecter des frontières de bandes de lumière structurée.

FIG. 6.5 – Reconstruction de l'ourson sur le piédestal automatique. Aucun miroir n'est utilisé, seul l'objet est tourné automatiquement. Temps total de reconstruction : 4 minutes ; temps total de post-traitement : 8 minutes ; définition : 265,000 points. L'image en haut à gauche est une photo de l'ourson, fournie à titre comparatif et prise avec un éclairage différent.

Nous pouvons observer pour les modèles présentés dans cette section que les éléments de surface orientés vers le sol ne sont généralement pas bien reconstruits. Cela est dû à la contrainte de vue en plongée de la caméra pour suivre temporellement le piédestal automatique avec précision. Nous utilisons le piédestal automatique conjointement avec un miroir dans la section suivante pour mieux remplir ces trous.

6.6 Piédestal automatique avec miroir

L'utilisation du piédestal implique une contrainte sur le placement de la caméra et du projecteur. Ceux-ci doivent avoir une vue en plongée sur le piédestal pour bien observer ses marqueurs de couleur et permettre le suivi temporel. Ainsi, les éléments de surface

FIG. 6.6 – Le tigre reconstruit à l'aide du piédestal automatique illustre la reconstruction en présence de texture colorée. Temps total de reconstruction : 8 minutes ; temps total de post-traitement : 2 minutes ; définition : 152,000 points.

orientés vers le sol ne sont ni visibles ni illuminés, ce qui empêche leur reconstruction.

Dans cette dernière section nous présentons les résultats de l'utilisation conjointe du piédestal automatique avec un miroir. Le miroir est placé de manière à offrir sur l'objet une vue en contre-plongée à la caméra et au projecteur, comme le montre la photographie de la Figure 6.8.

La Figure 6.9 montre la reconstruction d'une pomme en plastique. Cet objet illustre la reconstruction complète d'un objet sphérique, sombre, spéculaire et texturé. La pomme réelle est faite d'un plastique sombre mais assez spéculaire pour permettre la reconstruction ; cependant peu de points sont reconstruits à chaque itération. Grâce au miroir et au piédestal automatique, il est possible de produire suffisamment de points de vue pour obtenir une reconstruction complète de la pomme. Sur l'image de gauche on peut noter la présence de plusieurs *highlights* de lumière dans la partie sombre de la pomme causés, par les reconstructions selon plusieurs points de vue différents. La figure montre le modèle reconstruit selon trois points de vue et on peut observer que la forme et la texture sont correctement reproduites. La tige de la pomme n'a pas été reconstruite car elle est trop fine et ne réfléchit presque pas la lumière.

Une comparaison entre la reconstruction à l'aide du piédestal automatique avec et sans miroir est illustrée à la Figure 6.10. Les deux images du haut montrent le modèle

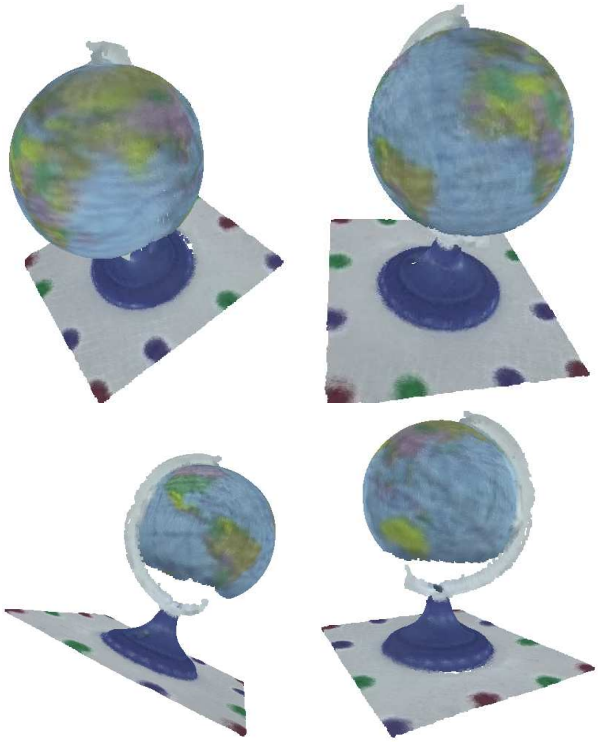


FIG. 6.7 – Le globe terrestre, reconstruit à l'aide du piédestal automatique illustre la reconstruction en présence de texture fine complexe. On peut observer que la forme sphérique du globe est assez bien préservée. Temps total de reconstruction : 5 minutes ; temps total de post-traitement : 3 minutes ; définition : 228,000 points.

reconstruit sans miroir ; les deux images du bas montrent la reconstruction du même objet, dans des conditions identiques, mais avec un miroir.

On peut y observer que le miroir a permis de reconstruire des points sur le bout et la partie inférieure du museau, de même qu'en arrière des oreilles. Ces parties, qui ne sont pas présentes dans les images du haut, sont reconstruites grâce aux points de vue en contre-plongée fournis par le miroir. Cependant, on doit noter dans les images du bas que le filtrage appliqué sur les points et les couleurs de ce modèle a été plus important pour diminuer le bruit présent.

La Figure 6.11 montre un autre modèle du chiot toujours reconstruit avec la même configuration. Les trois points de vue montrent le modèle complet obtenu en 20 minutes au total.

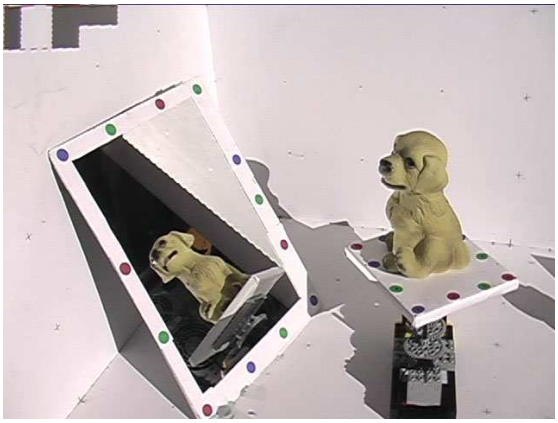


FIG. 6.8 – Image de la caméra illustrant les conditions de reconstruction lors de l'utilisation du miroir avec le piédestal automatique. On peut observer que la caméra a une vue en plongée sur l'objet pour permettre le suivi précis des marqueurs de couleur du piédestal. Le miroir est placé pour offrir une vue en contre-plongée.

FIG. 6.9 – Pomme en plastique reconstruite à l'aide du piédestal automatique et d'un miroir en un seul tour complet. Temps total de reconstruction : 5 minutes ; temps total de post-traitement : 2 minutes ; définition : 50,000 points.

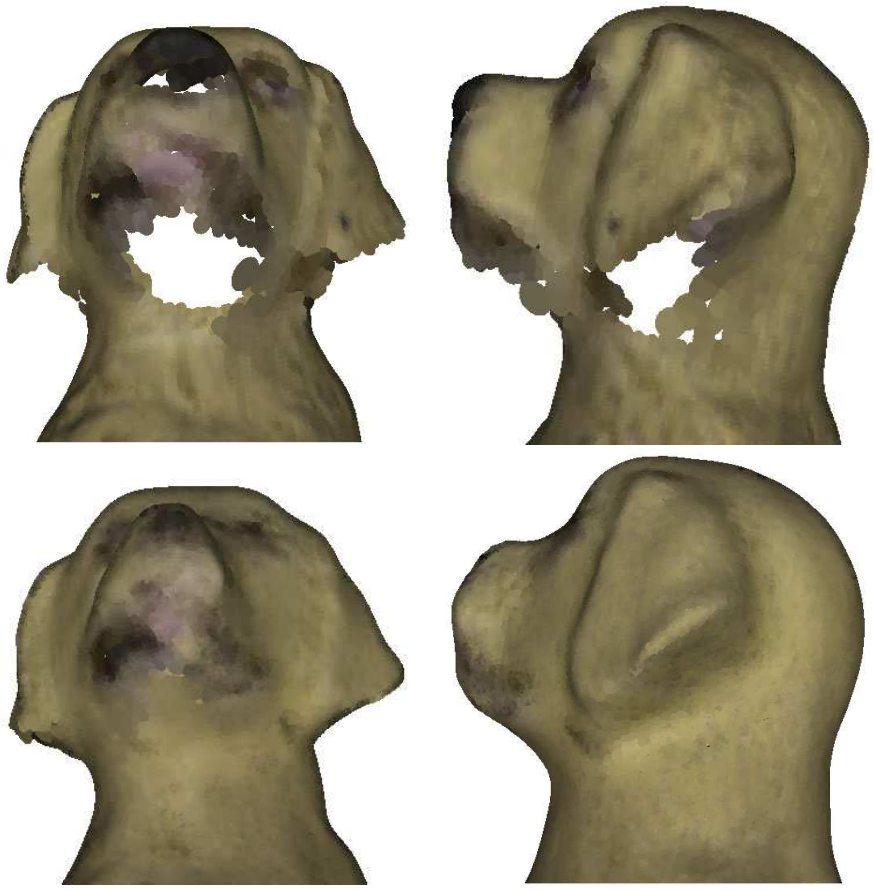


FIG. 6.10 – Comparaison de la reconstruction du modèle du chiot à l'aide du piédestal automatique, avec et sans miroir. Les conditions de reconstruction sont exactement les mêmes pour les deux modèles, seul le miroir a été désactivé pour le modèle illustré en haut. Ligne du haut : aucun miroir ; temps total de reconstruction : 5 minutes ; temps total de post-traitement : 5 minutes ; définition : 80,000 points. Ligne du bas : un miroir ; temps total de reconstruction : 8 minutes ; temps total de post-traitement : 7 minutes ; définition : 142,000 points.

FIG. 6.11 – Modèle du chiot reconstruit avec le piédestal automatique et un miroir. On peut observer que la partie inférieure museau est reconstruite malgré que cette surface pointe vers le sol. Temps total de reconstruction : 12 minutes ; temps total de post-traitement : 10 minutes ; définition : 307,000 points.

Chapitre 7

Conclusion

La reconstruction 3D d'objets et de scènes reste un défi de taille depuis plusieurs décennies. Bien que plusieurs techniques ont été introduites et que chacune continue à être améliorée, il n'en existe aucune à l'heure actuelle qui répond à toutes les exigences selon toutes les conditions.

Ce mémoire adresse l'intégration de miroirs dans un système de reconstruction interactif basé sur la lumière structurée. Même si les résultats de notre implémentation ne sont pas d'aussi bonne qualité (avec ou sans miroirs) que ceux de systèmes commerciaux parfaitement calibrés [Cyb, Ins], notre système nous a permis d'investiguer plusieurs avenues importantes et de proposer des améliorations aux systèmes de reconstruction par lumière structurée en général, et plus spécifiquement dans les situations où la caméra, le projecteur et l'objet à reconstruire sont contraints dans leurs positions et orientations. Nous pensons ainsi que les outils développés dans notre système se révèlent très appréciables lors d'une séance de reconstruction et que ceux-ci pourraient être autant bénéfiques dans d'autres systèmes de lumière structurée, ou s'intégrer aisément dans d'autres techniques de reconstruction 3D.

Nos contributions à l'intérieur de ce projet de recherche ont été les suivantes :

1. L'intégration d'un miroir dans un système de lumière structurée pour augmenter la couverture à la fois du projecteur et de la caméra.

2. L'extension à plusieurs miroirs avec le traitement approprié des interréflexions.

3. Le développement de motifs de lumière structurée uniques, robustes aux transformations projectives et aux réflexions miroir.

4. Plusieurs modifications au décodage du code des frontières de bandes de Hall-Holt et Rusinkiewicz [HHR01] permettant de reconstruire plus de points et d'éliminer la contrainte d'absence de texture sur les objets à reconstruire.

5. Le calcul efficace de masques, accéléré par la carte graphique, qui permettent d'éviter les conflits lors de la projection et de l'analyse des motifs peu importe la configuration des différents éléments du système.

6. Une mesure quantitative de la couverture totale du système (caméra, projecteur, objet, miroirs) simple et rapide à calculer pour déterminer la contribution d'un miroir et ainsi permettre le développement d'un système d'optimisation pour le placement des miroirs.

7. L'estimation simple et précise de la pose de miroirs en 3D par lumière structurée.

8. Une technique de suivi 2D couplée à l'estimation de pose permettant de déplacer interactivement un miroir dans le système.

9. Le développement d'un piédestal pour déplacer manuellement ou automatiquement l'objet à reconstruire tout en connaissant sa position de façon simple et robuste.

Bien que nous nous sommes concentrés sur le développement d'un système flexible et interactif, il est tout à fait possible de concevoir un système encore plus précis. L'estimation de pose de miroir pourrait intégrer une étape de projection de motifs sur le miroir qui, une fois réfléchis sur l'environnement de reconstruction, permettrait d'obtenir des points reconstruits sur l'environnement. Il serait alors possible de raffiner la pose du miroir en se basant sur les positions des points retrouvés après réflexion miroir et les directions dans l'espace des plans séparant les motifs projetés (avant réflexion). La technique d'optimisation de Mitchell et Hanrahan [MH92] pourrait alors améliorer la pose estimée du miroir.

Notre implémentation du suivi interactif du miroir est très limitée. Nous pensons que l'utilisation en parallèle d'un système de localisation commercial [Pol, Asc] permettrait d'obtenir un estimé suffisamment précis et rapide (à 60 Hz) pour projeter et analyser les motifs sur le miroir à tout moment. Il serait ainsi possible de déplacer librement les miroirs autour de l'objet sans contrainte sur la continuité des déplacements des

miroirs. Cependant, il faut tout de même placer le miroir dans un état statique pendant la projection de 6 motifs de lumière structurée pour permettre la reconstruction de l'objet. Avec l'utilisation conjointe d'un projecteur LCD et d'une carte de type *genlock* pour la synchronisation des signaux projetés et acquis, cela nous permettrait de reconstruire encore plus rapidement les objets (projection/acquisition à 30 Hz). Il serait alors possible d'implémenter le suivi des frontières de bandes et le miroir pourrait alors être déplacé librement tandis que l'estimation de la pose et la reconstruction seraient exécutées simultanément sans contraindre le miroir à rester dans un état statique durant ces projections.

L'utilisation de la lumière infrarouge (facilement projetée à l'aide d'un projecteur conventionnel muni d'un filtre bloquant la lumière visible) et d'une caméra infrarouge pourrait éliminer les effets "stroboscopiques" indésirables de projection des motifs lors d'une séance de reconstruction interactive [HHR01]. Une autre solution potentielle développée par Raskar *et al.* [RWC+98] consiste à projeter alternativement chaque motif et son inverse à une fréquence supérieure à 60 Hz. Cette technique, grâce à la persistance rétinienne, inhibe la perception visuelle des motifs de lumière structurée tout en permettant l'acquisition de ces motifs par une caméra haute vitesse (60 à 200 Hz). L'exploitation de la couleur pour différencier les motifs de lumière structurée permettrait éventuellement d'utiliser plusieurs miroirs simultanément lors de la reconstruction, mais il est évident que cela poserait problème pour les cas où l'objet à reconstruire est coloré ou texturé.

Nous n'avons porté que très peu d'attention à l'acquisition et au traitement des couleurs dans notre système. Une couleur est obtenue pour chaque point à l'instant où celui-ci est reconstruit (dans l'image de référence), puis les couleurs du modèle sont filtrées spatialement lors du post-traitement pour éliminer le bruit sur les couleurs des points (Figure 5.6). Étant donnée la quantité d'informations que nous avons à notre disposition, il serait possible d'extraire des paramètres de réflectance pour chaque point à partir des différentes images obtenues lorsque l'objet est déplacé ou à partir des points de vue virtuels fournis par les miroirs.

Nous avons aussi ignoré l'atténuation de la lumière lors de la réflexion dans un miroir, qui est visuellement observable dans nos résultats. Debevec utilise des sphères [Deb02] pour analyser l'illumination locale et globale d'une scène réelle et ainsi ré-illuminer un

objet virtuel intégré dans cette scène. Il serait possible de s'inspirer de cette technique pour tenter d'analyser l'atténuation de la lumière par les miroirs et compenser cette atténuation lors de la projection ou de l'acquisition d'informations à travers un miroir.

L'utilisation de miroirs courbes, à notre connaissance, n'a pas été rapportée dans le contexte de la lumière structurée et pourrait se révéler très intéressante. Cependant la localisation des séparations entre les bandes d'illumination serait plus complexe et il faudrait porter une attention particulière aux problèmes potentiels de mise au point pour la caméra et le projecteur. Dans un cadre plus simple, l'utilisation d'une boule miroir (multiples facettes planaires) pourrait être une piste intéressante pour la reconstruction d'environnements de l'intérieur vers l'extérieur. Il serait aussi possible de concevoir un environnement de reconstruction composé d'un nombre "optimisé" de miroirs précisément précalibrés entre eux.

Le système d'optimisation que nous avons développé n'est qu'une preuve de concept. Nous pensons que notre mesure d'erreur, basée sur le nombre de pixels valides dans les cartes des zones de la caméra et du projecteur, est une technique rapide pour évaluer l'efficacité d'une disposition donnée des éléments (caméra/projecteur/miroirs/piédestal) intervenant dans la reconstruction. Il serait intéressant de développer un système d'optimisation plus intelligent qui puisse tester une grande variété de dispositions des éléments du système, selon tous leurs degrés de liberté et ainsi obtenir un agencement idéal. Il serait aussi possible de prendre en compte les différentes contraintes : angle et distance entre caméra/projecteur, vue en plongée de la caméra, taille des motifs observés par la caméra, *etc.*, à l'intérieur de la mesure d'erreur, pour améliorer encore plus les résultats du système d'optimisation. Par ailleurs, un tel système d'optimisation développé spécifiquement pour un système de reconstruction robotisé [LPC+00] pourrait être très bénéfique car il permettrait de décider automatiquement du placement de la tête du numériseur 3D en fonction de la progression de la reconstruction. Ainsi il serait possible de procéder à une reconstruction complète en laissant le système d'optimisation déplacer le numériseur robotisé pour reconstruire les zones où il manque d'informations jusqu'à obtenir un modèle complet.

L'article de Levoy *et al.* sur l'imagerie homofocale (*confocal imaging*) [LCV+04] offre de nouvelles perspectives en proposant d'utiliser un mur de multiples miroirs pour simuler une grille de caméras et de projecteurs et ainsi obtenir ce qu'ils nomment un "appareil

photographique homofocal" et un "projecteur homofocal". Il pourrait être intéressant d'investiguer l'utilisation de la reconstruction par lumière structurée à l'intérieur de ce prototype qui permettrait possiblement d'obtenir une carte de profondeurs à couches multiples (*layered depth map*). Ce type de système permettrait différentes applications comme la reconstruction géométrique de scènes comportant beaucoup d'occlusions (buisson de feuilles ou arbre) ou la reconstruction de scènes en mouvement à très haute fréquence.

Finalement, nous pensons que notre recherche sur l'utilisation interactive de miroirs conjointement avec la lumière structurée offre des résultats concluants et intéressants. L'intégration fonctionnelle de miroirs ayant déjà été démontrée dans un système de reconstruction par stéréoscopie catadioptrique [BN99, GN01], nous pensons que les miroirs pourraient potentiellement être bénéfiques à toutes les autres techniques de reconstruction 3D. Aussi, comme l'indiquent les résultats et conclusions des articles sur la reconstruction par stéréoscopie spatio-temporelle (*space-time stereo*) [ZCS03, DRR03], nous pensons qu'une seule technique ne peut résoudre tous les problèmes de la reconstruction 3D. La solution viendra probablement de l'assemblage de plusieurs techniques et, selon nous, l'utilisation de miroirs pourrait faire partie de cet assemblage. Ainsi, un système hybride intégrant de multiples caméras virtuelles et projecteurs virtuels, tous simulés par des miroirs, et possiblement un mélange de techniques de reconstruction parmi la lumière structurée, la stéréovision, la sculpture d'espace, *etc.*, pourrait s'avérer plus flexible et précis que tous les systèmes existants qui utilisent ces techniques individuellement.

Chapitre 8

Glossaire

- ***Boîte englobante*** : *Bounding box* ; Parallélépipède rectangle synthétique englobant virtuellement l'objet à reconstruire. La boîte englobante est utilisée par le système pour savoir où projeter les motifs de lumière structurée et où analyser l'information dans les images de la caméra.
- ***Code de lumière structurée*** : Réfère généralement au type d'encodage de lumière structurée ou encore à une valeur particulière dans une séquence de codes de lumière structurée.
- ***Code de synchronisation*** : Motif de lumière structurée projeté sur l'environnement de reconstruction qui permet de connaître le numéro d'identification de toute image acquise par la caméra.
- ***Ellipsoïde de couleur*** : Forme géométrique qui englobe une distribution gaussienne de couleurs dans l'espace RGB. Il est utilisé pour l'identification des points de calibration, le suivi du piédestal et la segmentation de l'arrière-plan.
- ***Environnement de reconstruction*** : Trois panneaux rectangulaires en bois, mesurés et peints en blanc mat. Cet intérieur de cube contient les objets à reconstruire et sert à la calibration du projecteur et de la caméra.
- ***Lumière structurée*** : *Structured light.* Technique de reconstruction optique active.
- ***Marqueur de couleur*** : Petit autocollant rond et de couleur collé sur le piédestal et les miroirs pour faciliter le suivi 2D.
- ***Modèle 3D*** : Résultat de la numérisation de l'objet réel.
- ***Motif de lumière structurée*** : Séquence de codes de lumière structurée émise

par le projecteur.

- ***Objet*** : Objet réel à reconstruire.
- ***Piédestal*** : Petite boîte rectangulaire sur laquelle l'objet est placé lors de la reconstruction. Il est suivi dans la caméra grâce à des marqueurs de couleur et permet de déterminer où se trouve l'objet dans l'environnement de reconstruction.
- ***Usager*** : Personne qui interagit avec notre système de reconstruction.

Bibliographie

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin et C.T. Silva. « Point Set Surfaces ». Dans *IEEE Visualization 2001*, pages 21–28, octobre 2001.

[Ali] Alias|Wavefront. « Maya ». `http://www.aliaswavefront.com`.

[Asc] Ascension. « Flock of Birds ». `http://www.ascension.com`.

[ASG+] P. Andrews, J. Stuber, L. Griffiths, B. Bagnall, T. Rinkens et J. Solorzano. « Lejos - Java for the RCX ». `http://lejos.sourceforge.net`.

[Aut] Autodesk. « 3D Studio MAX ». `http://www4.discreet.com/3dsmax`.

[BM92] P.J. Besl et N.D. McKay. « A Method for Registration of 3D Shapes ». *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 14, numéro 2, pages 239–256, 1992.

[BN99] S. Baker et S.K. Nayar. « A Theory of Single-viewpoint Catadioptric Image Formation ». *International Journal of Computer Vision*, volume 35, numéro 2, pages 1 – 22, 1999.

[BR00] F. Bernardini et H. Rushmeier. « 3D Model Acquisition ». Dans *State of the Art Reports – Eurographics*, pages 41–62, 2000.

[BR02] F. Bernardini et H. Rushmeier. « The 3D Model Acquisition Pipeline ». *Computer Graphics Forum*, volume 21, numéro 2, pages 149–172, 2002.

[CM92] Y. Chen et G.G. Medioni. « Object Modelling by Registration of Multiple Range Images. ». *Image Vision Comput.*, volume 10, numéro 3, pages 145–155, 1992.

[CM99] Q. Chen et G. Medioni. « A Volumetric Stereo Matching Method : Application to Image-based Modeling ». Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 29–34, juin 1999.

[Cyb] Cyberware.
`http://www.cyberware.com`.

[DC01] J Davis et X. Chen. « A Laser Range Scanner Designed for Minimum Calibration Complexity. ». Dans *Proc. of International Conference on 3D Digital Imaging and Modeling*, pages 91–98, 2001.

[Deb02] P.E. Debevec. « Image-based Lighting. ». *IEEE Computer Graphics and Applications*, volume 22, numéro 2, pages 26–34, 2002.

[DMGL02] J. Davis, S. Marschner, M. Garr et M. Levoy. « Filling Holes in Complex Surfaces using Volumetric Difusion ». Dans *First International Symposium on 3D Data Processing, Visualization, and Transmission*, juin 2002.

[DRR03] J. Davis, R. Ramamoorthi et S. Rusinkiewicz. « Spacetime Stereo : A Unifying Framework for Depth from Triangulation ». Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[DTM96] P.E. Debevec, C.J. Taylor et J. Malik. « Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-based Approach ». *Proc. of SIGGRAPH '96*, volume 30, pages 11–20, 1996.

[EGPP04] E. Epstein, M. Granger-Piché et P. Poulin. « Exploiting Mirrors in Interactive Reconstruction with Structured Light ». Dans *Vision, Modeling, and Visualization 2004*, pages 125–132, novembre 2004.

[Enc] Wikipedia The Free Encyclopedia. « Gray Code ».
`http://en.wikipedia.org/wiki/Gray_coding`.

[Fau93] O. Faugeras. *Three-dimensional Vision — A Geometric Viewpoint*. MIT Press, 1993.

[GGSC96] S.J. Gortler, R. Grzeszczuk, R. Szeliski et M.F. Cohen. « The Lumigraph. ». Dans *Proc. of SIGGRAPH '96*, pages 43–54, 1996.

[GLL+04] M. Goesele, H.P.A. Lensch, J. Lang, C. Fuchs et H.-P. Seidel. « DISCO - Acquisition of Translucent Objects ». Dans *Proc. of SIGGRAPH '04*, juillet 2004.

[GN01] J. Gluckman et S.K. Nayar. « Catadioptric Stereo Using Planar Mirrors ». *International Journal of Computer Vision*, volume 44, numéro 1, pages 65–79, 2001.

[GP05] M. Granger-Piché. « Reconstruction interactive de modèle hierarchique par sculpture d'espace ». Mémoire de maîtrise, Département d'Informatique et Recherche Opérationnelle, Université de Montréal, janvier 2005.

[GPEP04] M. Granger-Piché, E. Epstein et P. Poulin. « Interactive Hierarchical Space Carving with Projector-based Calibrations ». Dans *Vision, Modeling, and Visualization 2004*, pages 159–166, novembre 2004.

[HHR01] O. Hall-Holt et S. Rusinkiewicz. « Stripe Boundary Codes for Real-time Structured-light Range Scanning of Moving Objects ». Dans *Proc. of International Conference on Computer Vision*, 2001.

[Ins] Inspeck. « 3D Capturor ».
`http://www.inspeck.com`.

[JC98] H. W. Jensen et P.H. Christensen. « Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps ». Dans *Proc. of SIGGRAPH '98*, volume 32, pages 311–320, juillet 1998.

[KS00] K.N. Kutulakos et S.M. Seitz. « A Theory of Shape by Space Carving ». *International Journal of Computer Vision*, volume 38, numéro 3, pages 199–218, 2000.

[LCV+04] M. Levoy, B. Chen, V. Vaish, M. Horowitz, I. McDowall et M. Bolas. « Synthetic Aperture Confocal Imaging ». Dans *Proc. of SIGGRAPH '04*, volume 23, pages 825–834, août 2004.

[LPC+00] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade et D. Fulk. « The Digital Michelangelo Project : 3D Scanning of Large Statues ». Dans *Proc. of SIGGRAPH '00*, pages 131–144, juillet 2000.

[Man] Manfrotto. « Super Clamp ».
`http://www.manfrotto.com`.

[MBR+00] W. Matusik, C. Buehler, R. Raskar, S. Gortler et L. McMillan. « Image-based Visual Hulls ». Dans *Proc. of SIGGRAPH '00*, pages 369–374, juillet 2000.

[MH92] D.P. Mitchell et P. Hanrahan. « Illumination From Curved Reflectors ». Dans *Proc. of SIGGRAPH '92*, volume 26, pages 283–291, juillet 1992.

[Min] Lego Mindstorms. `http://mindstorms.lego.com/eng/default.asp`.

[MP04] W. Matusik et H. Pfister. « 3D TV : a Scalable System for Real-time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes ». Dans *Proc. of SIGGRAPH '04*, volume 23, pages 814–824, 2004.

[MPN+02] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler et L. McMillan. « Image-based 3D Photography using Opacity Hulls ». Dans *Proc. of SIGGRAPH '02*, volume 21, pages 427–437, juillet 2002.

[Ntbt] M.L. Noga et the brickOS team. « BrickOS - an Alternative Operating System for the Lego Mindstorms RCX Controller ». `http://brickos.sourceforge.net`.

[Ope] SGI OpenGL. « Advanced Rendering Tutorials ». `www.sgi.com/software/opengl/advanced98/notes/node125.html`.

[PDH+97] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro et W. Stuetzle. « Robust Meshes from Multiple Range Maps ». Dans *Proc. of IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling*, mai 1997.

[PGK02] M. Pauly, M. Gross et L. Kobbelt. « Efficient Simplification of Point-sampled Surfaces ». Dans *Proc. of IEEE Visualization*, 2002.

[Pin01] C. Pinhanez. « The Everywhere Displays Projector : A Device to Create Ubiquitous Graphical Interfaces ». Dans *Proc. of Ubiquitous Computing '01*, volume 2201 de *Lecture Notes in Computer Science*, pages 315–331, 2001.

[POF98] P. Poulin, M. Ouimet et M.-C. Frasson. « Interactively Modeling with Photogrammetry ». Dans *Proc. of Eurographics Workshop on Rendering '98*, pages 93–104, juin 1998.

[Pol] Polhemus. « FastSCAN 3D Handheld Laser Scanner ». `http://www.polhemus.com/fastscan.htm`.

[Pro] ProjectionDesign. « iVision F1+ SXGA Projector ». `http://www.projectiondesign.com`.

[PSD+03] P. Poulin, M. Stamminger, F. Duranleau, M.-C. Frasson et G. Drettakis. « Interactive Point-based Modeling of Complex Objects from Images ». Dans *Proc. of Graphics Interface 2003*, pages 11–20, juin 2003.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling et B.P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992.

[Pul99] K. Pulli. « Multiview Registration for Large Data Sets ». Dans *Proc. of International Conference on 3D Digital Imaging and Modeling*, 1999.

[QL98] L. Quan et Z.D. Lan. « Linear $N \geq 4$-Point Pose Determination ». Dans *Proc. of International Conference on Computer Vision*, janvier 1998.

[RBB+03] R. Raskar, J.v. Baar, P. Beardsley, T. Willwacher, S. Rao et C. Forlines. « iLamps : Geometrically Aware and Self-configuring Projectors ». Dans *Proc. of SIGGRAPH '03*, volume 22, pages 809–818, juillet 2003.

[RCM+01] C. Rocchini, P. Cignoni, C. Montani, P. Pingi et R. Scopigno. « A Low Cost 3D Scanner based on Structured Light ». *Computer Graphics Forum*, volume 20, numéro 3, pages 299–308, 2001.

[Rea] RealViz S.A. « RealViz — Image Processing Software and Solutions for Content Creation ». `http://www.realviz.com`.

[RHHL02] S. Rusinkiewicz, O. Hall-Holt et M. Levoy. « Real-Time 3D Model Acquisition ». Dans *Proc. of SIGGRAPH '02*, volume 21, pages 438–446, juillet 2002.

[RL01] S. Rusinkiewicz et M. Levoy. « Efficient Variants of the ICP Algorithm ». Dans *Proc. of International Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[RLat] S. Rusinkiewicz et M. Levoy. « QSplat : A Multiresolution Point Rendering System for Large Meshes ». Dans *Proc. of SIGGRAPH '00*, pages 343–352,

2000

`http://graphics.stanford.edu/software/qsplat`.

[RPZ02] L. Ren, H. Pfister et M. Zwicker. « Object Space EWA Surface Splatting : A Hardware Accelerated Approach to High Quality Point Rendering ». Dans *Proc. of Eurographics 2002*, pages 461–470, 2002.

[RTJ94] E. Reinhard, L.U. Tijssen et F.W. Jansen. « Environment Mapping for Efficient Sampling of the Diffuse Interreflection ». Dans *Proc. of Eurographics Workshop on Rendering '94*, pages 410–422, juin 1994.

[RWC+98] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin et H. Fuchs. « The Office of the Future : A Unified Approach to Image-based Modeling and Spatially Immersive Displays ». Dans *Proc. of SIGGRAPH '98*, pages 179–188, juillet 1998.

[RWLB01] Ramesh Raskar, Greg Welch, Kok-Lim Low et Deepak Bandyopadhyay. « Shader Lamps : Animating Real Objects With Image-Based Illumination ». Dans *Proc. of Eurographics Workshop on Rendering '01*, pages 89–102, 2001.

[SCBB02] C. Samson, I. Christie, J.-A. Beraldin et F. Blais. « Neptec 3D Laser Scanner for Space Applications : Impact of Sensitivity Analysis on Mechanical Design ». Dans *Proc. of SPIE : Optoelectronics, Photonics, and Imaging*, pages 29–34, mai 2002.

[Sch] A. Schirmacher. « DVgrab - Digital Video for Linux ».
`http://kino.schirmacher.de`.

[SG99] R. Szeliski et P. Golland. « Stereo Matching with Transparency and Matting ». *International Journal of Computer Vision*, volume 32, numéro 1, pages 45–61, août 1999.

[Sof] Softimage, Corp. « SOFTIMAGE|XSI ».
`http://www.softimage.com/products/xsi`.

[SS03] D. Scharstein et R. Szeliski. « High-accuracy Stereo Depth Maps using Structured Light. ». Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[SSZ01] D. Scharstein, R. Szeliski et R. Zabih. « A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms ». Dans *Proc. of IEEE Workshop on Stereo and Multi-baseline Vision*, 2001.

[Sta] Stanford. « Scanalyze ».
`http://graphics.stanford.edu/software/scanalyze`.

[ZCS03] L. Zhang, B. Curless et S. Seitz. « Spacetime Stereo : Shape Recovery for Dynamic Scenes ». Dans *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[ZPKG3d] M. Zwicker, M. Pauly, O. Knoll et M. Gross. « Pointshop 3D : An Interactive System for Point-based Surface Editing ». Dans *Proc. of SIGGRAPH '02*, 2002
`http://graphics.ethz.ch/pointshop3d`.