Université de Montréal

Reconstruction interactive de scènes tridimensionnelles à partir d'images

par

Marie-Claude Frasson

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

5 septembre 1998

© Marie-Claude Frasson, 1998

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Reconstruction interactive de scènes tridimensionnelles à partir d'images

présenté par

Marie-Claude Frasson

a été évalué par un jury composé des personnes suivantes :

Président :	Jean Meunier
Directeur de recherche :	Pierre Poulin
Membre :	François Major

Sommaire

La production de modèles photoréalistes est un domaine de recherche devenu très actif dans les années quatre-vingt dix. Ces modèles sont maintenant nécessaires pour une quantité d'applications infographiques telles les environnements virtuels immersifs, les jeux vidéo, la visualisation d'environnements urbains pour des simulations de tout genre, etc. Pour obtenir des modèles fidèles au monde réel, quoi de mieux que de tenter de les extraire directement du monde réel ?

Nous présentons un système *interactif* permettant de retrouver la géométrie tridimensionnelle d'une scène à partir d'images de cette scène. Cette récupération s'effectue à l'aide de contraintes établies entre des primitives 2D qu'un usager doit tracer sur les images. Les différentes contraintes sont exprimées sous forme d'équations linéaires qui engendrent des systèmes d'équations surdéterminés. Ces systèmes sont solutionnés comme des problèmes de moindres carrés en utilisant les techniques de décomposition en valeurs singulières. Cette méthode permet de retrouver les matrices de projection qui approximent la projection qui a produit les images. Une fois ces matrices calculées, on peut récupérer la géométrie de la scène. Des itérations entre ces deux étapes permettent de raffiner le modèle obtenu. Des contraintes additionnelles visant à l'amélioration du modèle sont introduites et leurs effets analysés. Elles permettent en effet d'obtenir des modèles plus précis qu'avec de simples correspondances entre primitives.

Notre système est interactif car l'usager est intégré dans toutes les étapes du processus de reconstruction. Il place les primitives sur les images, les met en correspondance d'une image à l'autre et spécifie les contraintes qui relient les parties du modèle 3D afin de l'améliorer. Il s'occupe du jugement qualitatif des images et des résultats obtenus. Le système se charge de la partie quantitative ; il utilise les contraintes fournies par l'usager afin de récupérer les caméras et la géométrie de la scène. Nous obtenons donc un système général, flexible et robuste qui produit des modèles satisfaisants, utilisables en synthèse d'images.

Mots-clés :

Infographie, image de synthèse, reconstruction 3D, calibration de caméra, contraintes géométriques, systèmes d'équations linéaires, SVD, système interactif.

Table des matières

Liste des tableaux

Table des figures

À mes parents . . .

Remerciements

C’est enfin fini ! Ce n’est pas seulement moi qui lance cette phrase de libération mais aussi mon directeur de recherche, Pierre Poulin, ma famille et mes amis. Ces différentes personnes ont été d’une grande importance durant mes années de maîtrise. Sans eux, je n’aurais pu arriver jusqu’au bout.

Pierre a été d’une patience impressionnante tout au long de mon parcours. Il faut dire que j’ai mis cette patience à rude épreuve avec mes nombreux voyages, mes délais et retards de toutes sortes, mes incertitudes quant à mes décisions et choix et mes *bugs*. Je tiens énormément à le remercier pour son soutien permanent, ses conseils judicieux (ceux que j’ai suivi et ceux que j’aurais dû suivre) et tout le temps qu’il a pris pour m’aider durant ma recherche, mon projet, ma rédaction et mes décisions !

J’aimerais remercier ma famille pour leur aide, leur encouragement et leur amour. Ce dépôt est également un soulagement pour eux. Je souhaite bon courage à ma petite soeur, Corinne, qui entame actuellement le même cheminement. J’espère lui faire profiter de mon expérience.

Quant à mes amis au Canada et à travers le monde, la source d’énergie qu’ils m’ont apportée m’a permis de tenir le coup durant les durs moments de solitude et de découragement que l’on traverse aux études supérieures. Je leur suis infiniment reconnaissante pour leurs encouragements permanents et je m’excuse pour avoir si peu donné de nouvelles durant ces longs mois de recherche et de rédaction. Je félicite ma meilleure amie Zeina et mon grand ami et supporter Vincent pour leur dépôt et j’encourage mon amie d’enfance et compagne de sport, Natacha, ainsi que le grand Francis pour leur dernière ligne droite. Vous m’avez été indispensables ! J’adresse également une pensée particulière pour leur soutien et leurs attentions à Myriam, Guillaume, Jean-François, Sébastien, Emmanuelle, Martin, Mike, Rick, Agnieszka, Eva, Jamila, Martine, Diane, Bob, Juan et Kim ainsi qu’aux gens du laboratoire d’infographie dont les caractères très différents ont fait de ce labo un endroit agréable pour travailler et rigoler. Je suis sûre que le département d’informatique (DIRO), après y avoir passé six ans, va me manquer.

Les organismes subventionnaires du CRSNG et du FCFDU m’ont permis d’effectuer ma maîtrise sans avoir à me soucier de mes finances. Je les en remercie grandement.

Et finalement, j’aimerais également dédier ce mémoire à mes deux grands-mères qui auraient été fières de moi et qui doivent quand même l’être là où elles sont.

Chapitre 1

Introduction

"Reality is that which, when you stop believing in it, doesn't go away."

—Phillip K. Dick

1.1 Problématique

La technologie entourant les environnements dits *virtuels* progresse rapidement. On a donc de plus en plus besoin de modèles intéressants et surtout réalistes pour meubler ces environnements. Malheureusement, la production de modèles photoréalistes[1] est une tâche très fastidieuse lorsque l'on utilise les techniques traditionnelles de synthèse d'images. La modélisation de structures géométriques, la simulation de textures et d'illumination demandent des connaissances préalables et ne sont donc pas accessibles immédiatement. Une alternative intéressante et plus simple serait d'extraire des modèles directement de la réalité elle-même. Si ces modèles sont trop complexes, on pourrait se contenter d'extraire un modèle géométrique simple approximant l'original puis d'en simuler les détails par l'application de textures provenant également des images. On pourrait donc récupérer les textures et l'illumination du monde réel, ce qui nous éviterait d'avoir à utiliser des approximations de ces phénomènes telles que celles actuellement utilisées en synthèse d'images traditionnelle. La perspective de pouvoir créer des modèles réalistes à partir de simples photographies prises par des appareils photos ordinaires est de fait très attirante : plus besoin d'avoir des talents d'artiste pour pouvoir créer un modèle convaincant et fidèle à l'original. Bien évidemment, la technique utilisée pourrait aussi être appliquée à toutes sortes d'images réelles ou de synthèse, peintures, dessins (en autant que les

[1] Modèles dont on peut générer des images indiscernables des photographies de la réalité. Dans ce texte, nous utiliserons le terme "réaliste" au même sens que "photoréaliste".

règles de perspective soient respectées), etc.

Ces dernières années, plusieurs équipes de recherche à travers le monde se sont intéressées au problème de produire des modèles tridimensionnels réalistes de scènes à partir d'une séquence d'images de ces scènes. Mais ces images posent deux problèmes importants. Elles représentent la scène avec deux dimensions seulement et la notion de profondeur est donc perdue. De plus, l'information sur les paramètres ou le mouvement des caméras entre les prises d'images est souvent absente, ce qui complique la tâche de reconstruction.

Malheureusement, les méthodes provenant du domaine de la vision par ordinateur pour construire automatiquement de tels modèles à partir de projections 2D ne sont pas encore très efficaces ou robustes. Elles se heurtent encore aux problèmes auxquels fait face le domaine depuis des années : bruit dans les images causant des erreurs dans les correspondances stéréo, mauvaise détection de contours, parties cachées, ambiguïtés nombreuses auxquelles doit faire face le processus de segmentation (détermination des types de discontinuités), etc.

Une solution à ces divers problèmes consiste à utiliser l'usager qui peut effectuer lui-même la segmentation des images et les correspondances entre les diverses primitives 2D présentes dans les images disponibles. Nous avons donc bâti un système interactif de reconstruction de modèles tridimensionnels à partir de photographies : le système *Rekon* [POF98]. Avec l'aide de l'usager, *Rekon* peut retrouver les matrices de projection des images qu'on lui donne en entrée et, une fois certaines vues calibrées, il s'attache à essayer de reconstruire le modèle géométrique de la scène représentée par les images.

Afin de rendre le processus de reconstruction assistée moins long et pénible pour l'usager et d'améliorer le modèle géométrique récupéré, nous avons introduit dans le système des contraintes supplémentaires à celles qui en forment la base. Ces contraintes dérivent directement des caractéristiques inhérentes à la plupart des scènes qui nous entourent, telles la planarité, le parallélisme et la perpendicularité. Elles permettent d'obtenir des modèles plus satisfaisants et augmentent le contrôle de l'usager sur la reconstruction.

1.2 Structure du mémoire

Après avoir introduit les notions importantes dans le domaine de la reconstruction tridimensionnelle, nous survolerons différentes approches actuellement utilisées dans les domaines de vision, d'infographie et d'intelligence artificielle. Nous décrirons ensuite l'approche que nous proposons ainsi que le système qui en a découlé. Les contraintes ajoutées afin d'améliorer les modèles obtenus seront

explicitées plus en détails et les résultats obtenus par l'application de chaque type de contrainte seront étudiés. Nous montrerons ensuite les résultats obtenus sur divers types de scènes. Nous pourrons constater que notre approche se démarque des autres par sa simplicité et sa robustesse et produit des modèles d'une qualité équivalente sinon supérieure.

1.3 Remarques préliminaires

Nous présentons dans cette section les conventions de notation adoptées tout au long du mémoire, la nomenclature utile à la bonne compréhension des principes mathématiques ainsi que la terminologie utilisée dans le texte.

Au niveau des tests réalisés, nous avons utilisé un ordinateur Silicon Graphics Indigo2 Impact possédant un processeur R10 000 cadencé à 195 MHz et une mémoire vive de 128 mégaoctets (*Megabytes* ou *Mb*).

1.3.1 Conventions

Nous avons adopté certaines conventions de notation pour les formules mathématiques utilisées tout au long du mémoire.

Les entités 3D sont signalées par des lettres majuscules, et les entités 2D par des minuscules. Ainsi, le point P dans l'espace se projette en un point p sur un plan.

Les matrices sont en majuscules et de style gras, par exemple la matrice $\mathbf{A}$. Les vecteurs sont en minuscules et en gras, sauf s'il s'agit de notre représention vectorielle des points, lignes, plans, etc. Par exemple, le vecteur $\mathbf{v}$, mais le point $p = \begin{bmatrix} p_x & p_y & p_h \end{bmatrix}^T$.

Certains livres d'infographie [RA90] ont comme convention de pré-multiplier les matrices de transformation par des vecteurs lignes, plutôt que de post-multiplier par des vecteurs colonnes. Nous adoptons dans ce texte la deuxième façon, qui est celle utilisée dans les principaux livres actuels [FvDFH90] [HB94]. Ainsi, $p = \mathbf{T}P$ représente la projection d'un point 3D P en un point 2D p sur un plan à l'aide de la matrice de projection $\mathbf{T}$.

Nous allons travailler avec des systèmes d'équations linéaires. Nous utilisons la représentation matricielle de tels systèmes dans laquelle $\mathbf{Ax} = \mathbf{b}$ où $\mathbf{A}$ est une matrice $m \times n$ de coefficients, $\mathbf{x}$ un vecteur $n \times 1$ d'inconnues et $\mathbf{b}$ un vecteur $m \times 1$. Les n inconnues de $\mathbf{x}$ sont reliées par les m équations de $\mathbf{A}$ donnant les m valeurs se trouvant dans $\mathbf{b}$.

1.3.2 Nomenclature

Nous utilisons des principes de géométrie projective et donc nos primitives géométriques ont une dimension de plus qui constitue leur *coordonnée homogène*. Elles sont donc *homogènes*, ce qui signifie que leur position euclidienne reste inchangée si on multiplie, dans l'espace homogène, leurs coordonnées par une constante.

Un point 2D en coordonnées homogènes s'exprime comme $p(p_x, p_y, p_h)$ ou, sous forme vectorielle, $p = \begin{bmatrix} p_x & p_y & p_h \end{bmatrix}^T$. Si $p_h = 1$, le point est normalisé et ses coordonnées sont équivalentes à ses coordonnées euclidiennes $(\frac{p_x}{p_h}, \frac{p_y}{p_h})$. Le plan euclidien est le plan défini par $h = 1$.

Un point 3D homogène s'exprime comme $P(P_x, P_y, P_z, P_h)$ ou $P = \begin{bmatrix} P_x & P_y & P_z & P_h \end{bmatrix}^T$. A certaines occasions, nous utiliserons la notation $P = \begin{bmatrix} x & y & z & h \end{bmatrix}^T$ et $p = \begin{bmatrix} x & y & h \end{bmatrix}^T$ pour plus de clarté.

Une ligne 2D est représentée par son équation $ax + by + c = 0$. Tout point $p(x, y)$ vérifiant cette équation appartient à la ligne. Sous forme vectorielle, on peut écrire $\begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} x & y & 1 \end{bmatrix}^T = 0$, où "$\cdot$" est le produit scalaire de deux vecteurs. Le premier vecteur contient les coefficients de la droite et le second les coordonnées homogènes d'un point. On peut ainsi tester l'incidence d'un point sur une ligne.

La représentation d'une ligne 3D est décrite dans la section 3.4 et plus en détails à l'annexe A.

Un plan 3D Π est représenté par l'équation $ax + by + cz + d = 0$. Tout point $P(x, y, z)$ vérifiant cette équation appartient au plan. Sous forme vectorielle, $\Pi \cdot P = \begin{bmatrix} a & b & c & d \end{bmatrix} \cdot \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T = 0$ lorsque P homogène normalisé, $P = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$, appartient au plan $\Pi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$. La normale à ce plan est $N = \begin{bmatrix} a & b & c \end{bmatrix}^T$. Une normale, comme tout vecteur, possède une orientation (sens du vecteur) et une direction (pente de la droite colinéaire au vecteur).

1.3.3 Terminologie

Tout au long de ce mémoire, nous emploierons des termes relatifs au domaine étudié et qui sont équivalents.

Une *scène* est un ensemble de primitives en trois dimensions, connectées ou non. Ces primitives peuvent former des *objets*. On parle également de *modèle* ou de *géométrie*.

Une *image* est le résultat de la projection d'une scène sur un plan de projection. Une image a donc deux dimensions. Cette scène peut avoir préalablement subit une série de *transformations*. Nous utiliserons parfois le terme *projection d'un objet* ou *vue* pour signifier l'image d'un objet.

Nous allons étudier la *reconstruction* de scènes à partir d'images de ces scènes. On parlera indifféremment de reconstruction, de *récupération de géométrie* ou de *structures 3D* et de *calcul de modèle*.

Chapitre 2

Motivation

"To steal ideas from one person is plagiarism ; to steal from many is research."

—Unknown

Il existe dans la littérature scientifique un nombre incalculable de documents ayant trait à la reconstruction 3D. Ceci est dû au fait que plusieurs domaines de recherche s'y intéressent fortement mais à des fins différentes. Il convient donc de bien discerner les buts et les besoins de chacun de ces domaines afin de ne pas s'égarer devant une telle quantité d'information. C'est ce que nous tentons de faire dans ce chapitre. Nous expliquons également en quoi consistent la reconstruction 3D et l'indissociable étape de *calibration* qui l'accompagne. Nous présentons ensuite des systèmes de reconstruction actuels provenant de différents domaines. Nous faisons enfin une brève description du système *interactif* de reconstruction que nous proposons avant de l'exposer plus en détails au cours des chapitres subséquents.

2.1 La reconstruction 3D

2.1.1 Principaux domaines concernés

Plusieurs domaines s'intéressent au problème de reconstruire des modèles à partir de leurs images. Certains depuis longtemps (photogrammétrie, domaine de la vision et surtout de la robotique) et d'autres plus récemment (intelligence artificielle en vision, infographie). Les raisons qui incitent chacun d'eux à tenter de reconstruire des scènes à partir de leurs projections sont variées et il existe ainsi quelques techniques applicables à un ou plusieurs de ces domaines. Les problèmes auxquels ils se heurtent sont parfois bien différents, rendant certaines techniques propres à un domaine inutiles pour un autre. Par contre, il arrive également que certaines approches puissent être adaptées ou simplifiées.

Il est donc important de pouvoir faire la différence entre la reconstruction nécessaire à un domaine et celle nécessaire à un autre, les résultats escomptés et obtenus étant parfois bien différents selon le type d'application visé.

Photogrammétrie

La photogrammétrie applique les techniques de stéréoscopie[1], et plus particulièrement de la photographie stéréoscopique, aux levés topographiques, aux relevés des formes et dimensions des objets, aux reliefs, etc. La forme d'un objet est déterminée à l'aide de photographies qui se chevauchent, prises à l'aide de caméras soigneusement calibrées[2]. Les applications visées sont typiquement des applications 2D$\frac{1}{2}$, par exemple des cartes topographiques. Les principes de la photogrammétrie traditionnelle se retrouvent dans le manuel de photogrammétrie de la société américaine de photogrammétrie [ASP66].

L'extraction de bâtiments tridimensionnels à partir d'images est nécessaire pour un nombre grandissant de tâches relatives au mesurage, à la planification, la construction, l'architecture, l'environnement, les transports, la gestion de propriétés, etc. Les méthodes photogrammétriques traditionnelles se montrent parfois inefficaces en raison de la vaste quantité de données à traiter comparativement aux applications 2D$\frac{1}{2}$.

Avec l'arrivée des techniques de reconstruction par ordinateur, d'acquisition de données tridimensionnelles (caméras digitales, scanners 3D), et de visualisation, une discipline connexe a vu le jour, la *modélisation photogrammétrique* ou *photogrammétrie digitale* [Str94]. On peut la définir comme une méthode pour récupérer interactivement des modèles 3D et des positions de caméras à partir de photographies, associée à des techniques photogrammétriques traditionnelles. L'intégration d'outils d'analyse d'images automatiques ou semi-automatiques en photogrammétrie permet d'augmenter l'efficacité des algorithmes traditionnels pour l'acquisition de telles données. L'opérateur peut, avec ces techniques, effectuer la reconstruction complète d'objets 3D sans mesures manuelles sur le site.

Vision par ordinateur

Le processus de reconstruction de structures en trois dimensions à partir d'images est depuis longtemps un des principaux champs de recherche du domaine de la *vision par ordinateur*. Il n'existe pas encore de technique générale mais plusieurs domaines connexes à la vision ont apporté des résultats

[1] Procédé donnant l'impression du relief par utilisation de deux images d'un sujet prises avec des points de vues différents.

[2] Les techniques de calibration seront explicitées un peu plus loin.

intéressants applicables au problème de reconstruction. Parmi ceux-ci, on peut citer l'inférence de structure à partir du mouvement, la stéréoscopie, la modélisation à partir d'images (ou cartes) de profondeur (*range images*) et le rendu à partir d'images (*image-based rendering*).

La reconstruction de scènes 3D est cruciale en robotique. La navigation d'un robot dans un environnement nécessite la reconstruction du monde qui l'entoure afin de pouvoir détecter les collisions éventuelles et les chemins à emprunter. Par contre, un modèle 3D précis n'est pas nécessaire, la connaissance de volumes englobants et de positions relatives au robot étant souvent suffisante. Le succès d'un système robotique intelligent dépend des performances de son système de vision, qui lui, dépend en grande partie de la qualité de sa calibration.

Dans le domaine de la reconnaissance de formes, certaines techniques permettent de construire un modèle 3D partiel afin de le comparer à d'autres contenus dans une base de données pour tenter de le reconnaître. Il n'est pas nécessaire que le modèle construit soit rigoureusement exact ou correct. Il n'est pas destiné à être affiché et donc sa qualité visuelle n'est pas essentielle, contrairement à un modèle destiné à des applications infographiques.

Il en résulte qu'en vision, les modèles reconstruits ne sont la plupart du temps pas formés de facettes reliées entre elles. Ce sont souvent des points ou des segments "flottant" en 3D ou même seulement des cartes de profondeur. Il serait donc difficile d'y appliquer une méthode de rendu traditionnelle sans obtenir de nombreux défauts. Étant donné que ces modèles ne sont pas destinés à être traités par des algorithmes infographiques, cette représentation est suffisante pour les besoins du domaine de la vision. La plupart du temps, les modèles reconstruits dans les divers champs d'étude de la vision ne constituent qu'une première étape pour une application quelconque, alors qu'en infographie, le but ultime de la reconstruction est de récupérer un modèle visuellement satisfaisant. Il convient donc de bien distinguer entre la reconstruction en vision et celle en infographie. Les propriétés du modèle désiré sont en effet souvent très différentes.

Intelligence artificielle

A la fin des années soixante-dix, des techniques d'intelligence artificielle ont été introduites dans le domaine de l'analyse et de l'interprétation d'images. Cette interprétation s'effectue normalement en assignant des étiquettes (*labels*) aux différentes caractéristiques extraites des images suite à leur analyse. En ajoutant des connaissances symboliques telles que des relations géométriques entre les différentes caractéristiques, on réduit le nombre de possibilités quant à leur étiquetage durant le processus d'interprétation ainsi que le nombre d'erreurs d'appariements entre primitives. On appelle ce

processus l'*analyse de scènes*.

Des systèmes de reconstruction automatiques utilisant des connaissances a priori sur la scène ont donc vu le jour. Par contre, le désavantage principal que possède cette technique est que, la plupart du temps, les bases de connaissances utilisées pour la reconstruction ne sont spécifiques qu'à un domaine. Il faut alors adapter les contraintes selon le type de scènes que l'on cherche à reconstruire, ce qui diminue la généralité de tels systèmes. Les modèles recherchés sont très similaires à ceux demandés par la vision.

Infographie

Les environnements virtuels immersifs, les jeux vidéo, la réalité augmentée, les effets spéciaux dans les films, etc. font partie de la multitude d'applications graphiques qui nécessitent l'utilisation de plus en plus de modèles intéressants ou connus. Par exemple, de tels modèles sont nécessaires dans le cadre de simulateurs de conduite ou de systèmes de planification architecturale ou paysagère. Jusqu'à récemment, tous ces modèles étaient créés à la main, à l'aide de systèmes de modélisation (*CAD*) de plus en plus sophistiqués.

La reconstruction de modèles 3D à partir d'images est un champ de recherche récent en infographie. En effet, les modèles actuellement produits par des techniques infographiques traditionnelles ne réussissent toujours pas à tromper les observateurs attentifs. La plupart des textures sont facilement décelables comme étant générées par ordinateur ; elles sont trop "propres", trop régulières, pas assez réalistes. Les modèles, quant à eux, sont parfois très fastidieux à construire et exhibent également des défauts plus ou moins détectables, d'où une perte de réalisme inévitable. Les modèles sont souvent trop parfaits, trop propres ou trop simplifiés. Le défi de l'infographie est effectivement de taille : simuler la réalité qui est accessible à n'importe qui. Excepté pour des environnements complètements fictifs et loin de la réalité qui nous entoure, chacun est en mesure de pouvoir juger de la qualité d'un environnement infographique voulant simuler le monde réel. Nul besoin pour cela d'avoir de connaissances préalables, le seul fait de vivre dans le milieu que l'on tente de reproduire nous rend automatiquement critiques qualifiés. La tendance actuelle est donc de créer des modèles de plus en plus réalistes. Pour cela, on tente d'extraire les modèles et les textures directement des images plutôt que de tenter de les construire ou de les simuler. La nécessité de modèles réalistes est bien justifiée dans le domaine de la réalité augmentée où l'on voudrait que l'observateur ne puisse distinguer entre les objets qui existent effectivement dans une scène réelle et les objets synthétiques qui y ont été ajoutés.

Le but de la reconstruction en infographie est donc de créer plus rapidement des modèles qui correspondent à des objets dont on possède des images, que ce soit des photographies, des images de synthèse, des dessins, des plans ou des peintures. Par contre, il est important que les modèles obtenus répondent aux attentes des observateurs humains. Les distorsions géométriques sont particulièrement détectables et donc critiques. Certaines imprécisions et l'inexactitude des modèles sont reconnues immédiatement et peuvent déranger l'observateur. Il faut donc trouver un moyen de corriger les anomalies éventuelles d'une reconstruction. De plus, les modèles générés doivent être utilisables par des applications graphiques, et donc similaires aux modèles créés à l'aide de systèmes de modélisation assistée par ordinateur.

2.1.2 Principes généraux

Le problème auquel on fait face en tentant de reconstruire une scène à partir de ses projections est que la projection n'est pas bijective. En passant du monde tridimensionnel au monde image bidimensionnel, on perd une dimension qui n'est pas facilement récupérable. Les projections en deux dimensions à partir d'objets en trois dimensions sont étudiées depuis longtemps en géométrie. Le problème inverse de reconstruire *automatiquement* la structure d'un objet en trois dimensions à partir de ses projections a commencé à attirer l'attention dans les années soixante-dix, avec le développement des ordinateurs.

Le principe de base pour la récupération de l'information 3D à partir d'images 2D est la *triangulation.* Lorsque l'on dispose de la projection 2D d'un point de l'espace, sa position 3D peut être n'importe où sur un rayon passant par le point 2D et le centre optique de la caméra (centre de projection). Si l'on possède les positions des projections dans les images d'un même point 3D, on peut calculer sa position en trouvant l'intersection des rayons passant par chaque projection du point dans chaque image et par le centre optique de la caméra correspondante (figure 2.1). Typiquement, dû à l'imprécision des images (discrétisation des pixels), à l'incertitude sur la position du centre optique et à la représentation en points flottants, les rayons ne s'intersecteront pas exactement. Toutefois, on peut trouver un point d'intersection *moyen* par une technique de moindres carrés minimisant la somme des distances au carré des rayons au point.

Ce calcul implique que les paramètres (position, orientation, réglages, etc.) des caméras sont connus. La détermination de ces paramètres est une étape cruciale en reconstruction et sera étudiée dans la section suivante. Dans plusieurs systèmes, cette étape ainsi que celle de reconstruction sont séparées, la calibration précédant la reconstruction. Nous verrons qu'il est également possible de les

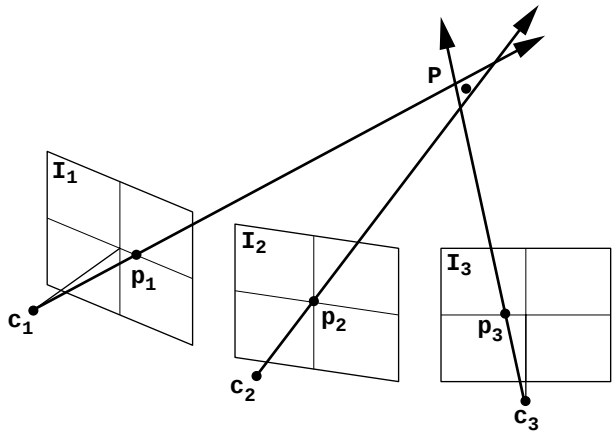


FIG. 2.1: Triangulation multi-images. c_1, c_2 et c_3 sont les centres de projection des images I_1, I_2 et I_3. p_1, p_2 et p_3 sont les projections du point 3D P sur ces images.

entrelacer, chacune aidant l'amélioration de l'autre.

Si l'on possède un nombre suffisant de projections 2D de plusieurs points 3D dans plusieurs images, on peut théoriquement retrouver leur position 3D ainsi que la position des caméras, à un facteur d'échelle près. En 1913, Kruppa [Kru13] a prouvé que si l'on dispose de deux vues de cinq points, on peut récupérer la rotation et la translation entre les deux caméras ainsi que la position dans l'espace des points, à un facteur d'échelle près. Ce résultat fondamental est encore utilisé par la plupart des techniques actuelles (équations de Kruppa).

2.1.3 Approches possibles

Il existe deux courants de pensée sur la reconstruction. On peut essayer de la faire automatiquement ou intégrer l'utilisateur dans le processus.

La reconstruction automatique

C'est une approche employée dans le domaine de la vision par ordinateur où ce genre d'automatisme est nécessaire, dans des systèmes robotiques par exemple. Plusieurs travaux [Zha98] [RC98] [FHM+93] ont essayé de produire des systèmes de stéréovision[3] "automatiques". Le principe de base est l'*appariement* (ou mise en correspondance) de points 2D sur les images provenant de deux caméras afin de retrouver leur position 3D par triangulation. Pour chaque point dans une image, son correspondant dans l'autre image doit être trouvé automatiquement, le plus souvent en comparant leurs intensités (ou couleurs) et en choisissant celles qui correspondent le mieux.

[3]Utilisation de deux caméras faiblement espacées pour tenter de récupérer la structure 3D de la scène observée simultanément par les deux caméras.

Cette mise en correspondance est souvent difficile en raison de plusieurs facteurs tels le bruit dans les images, les différences d'intensité causées par l'illumination variable de la scène, les occlusions, la faiblesse de contraste, etc. Pour maximiser les chances de réussite, on veut éviter les différences d'illumination entre les deux images, et pour cela, on ne peut avoir un espacement (*baseline*) trop grand entre les caméras (*cohérence spatiale*). Cette contrainte empêche l'utilisation de la stéréoscopie pour la reconstruction de grands environnements (par exemple, une ville), étant donné qu'il nous faudrait des milliers d'images pour respecter cette contrainte d'espacement. De plus, plus les caméras sont proches l'une de l'autre, plus l'incertitude sur la profondeur des points en correspondance est élevée. Par contre, si on écarte les caméras afin d'obtenir une meilleure précision sur cette profondeur, l'espace de recherche pour les correspondances est élargi et donc plus susceptible de produire de mauvais résultats, en plus d'allonger la durée du processus. Également, les images doivent avoir été prises à peu près au même moment pour disposer de la même illumination (*cohérence temporelle*) et pour éviter que les objets soient déplacés et donc plus considérés comme statiques. Ces deux contraintes de cohérence permettent d'utiliser le suivi incrémental de pixels pour favoriser la recherche du correspondant.

L'ajout de contraintes géométriques inhérentes à la caméra ainsi que de relations structurales dans les images pour la mise en correspondance constitue une manière de diminuer les erreurs. Plusieurs chercheurs se sont penchés sur l'utilisation de telles contraintes durant le processus de mise en correspondance [CT97]. Également, l'utilisation de la contrainte épipolaire (décrite plus loin) ou d'autres types de contraintes découlant de la géométrie projective permet de réduire la dimension du problème de correspondance. Dans le domaine de l'intelligence artificielle, on essaie d'améliorer la reconstruction par l'utilisation d'une base de contraintes propre au domaine auquel appartient le modèle à reconstruire.

La reconstruction automatique pose aussi le problème qu'il n'est pas possible de sélectionner ce que l'on voudrait reconstruire. Des images et en particulier des photographies contiennent une quantité importante d'information considérée inutile et potentiellement nuisible si on ne voudrait qu'extraire certains objets de la scène imagée. Par exemple, si on ne veut reconstruire qu'un bâtiment, toute la végétation qui l'entoure, les voitures et les personnes qui se trouvent là au moment de la photographie, constituent du "bruit" (si l'on peut dire) qui complique la tâche d'un algorithme automatique, voire le déroute complètement.

La plupart des systèmes actuels qui se disent "automatiques" nécessitent une initialisation manuelle puis améliorent la solution donnée par des techniques automatiques de vision. En réalité, vou-

loir tout faire automatiquement est pour l'instant utopique. Il n'existe pas encore de méthodes de segmentation et de correspondance automatiques à toute épreuve.

La reconstruction assistée

Lorsqu'un système automatique de reconstruction ne réussit pas à produire un modèle satisfaisant, il est très difficile de déterminer quelle étape de la reconstruction a posé problème. On peut passer beaucoup de temps à chercher où se trouve la faille dont on constate seulement l'effet sur le résultat final. Par exemple, on peut avoir obtenu une mauvaise segmentation à cause d'un mauvais choix de seuil d'extraction de contours, ceci engendrant des erreurs d'appariements. Ce manque de contrôle est une des motivations à l'intégration de l'usager dans le processus de reconstruction. Cette intégration se fait à des niveaux différents selon les approches et peut concerner chacune des deux étapes principales de calibration et reconstruction. L'usager peut aider à la segmentation de l'image, la mise en correspondance, l'ajout de relations, de contraintes, etc.

2.1.4 Calibration

La récupération de la géométrie 3D à partir de plusieurs images est un problème beaucoup plus facile lorsque les caméras qui ont produit ces images ont été préalablement *calibrées*. Avant d'exposer quelques techniques de calibration, voyons d'abord rapidement quelques définitions relatives au domaine. Plus de détails sur la calibration et les caméras peuvent être trouvés dans le livre de Faugeras sur la vision 3D par ordinateur [Fau93].

Qu'est-ce qu'une caméra ?

On définit une *caméra* comme étant un moyen pour obtenir une image 2D d'une structure 3D, en d'autres termes un *système d'acquisition d'images* ou un système d'imagerie. Ce système peut être un appareil photo de n'importe quel type (réel ou synthétique[4]), une caméra vidéo (réelle ou synthétique), un peintre qui respecte les règles de la perspective, etc.

Un modèle de caméra est souvent défini par des paramètres *intrinsèques* et *extrinsèques*. Les paramètres intrinsèques d'une caméra modélisent les caractéristiques internes de la caméra et ne dépendent donc pas de sa position et de son orientation dans l'espace. Par exemple, pour le

[4]Une simulation de l'effet obtenu par un certain type d'appareil photo, fonctionnalité souvent fournie dans un système de modélisation.

modèle *sténopé*[5] (*pinhole*) , ils incluent la distance focale, le centre de projection et les coefficients d'échantillonnage des pixels. Les objectifs réels d'appareil photos (contrairement aux objectifs virtuels des caméras synthétiques) peuvent introduire des distorsions radiales non-linéaires dans les images, effet modélisé par certains auteurs [KHM95] et normalement bien approximé par une simple paramétrisation.

Les paramètres *extrinsèques*, quant à eux, relient le système de coordonnées de la caméra au système de coordonnées du monde par une série de transformations 3D. Ils représentent la position et l'orientation de la caméra par rapport à un référentiel fixe. Selon l'application, ce référentiel peut être lié à l'objet observé, à la caméra dans une autre position ou à une autre caméra.

Que signifie calibrer une caméra ?

Le but de la calibration d'une caméra consiste à déterminer la relation analytique entre les coordonnées 3D du monde et leurs coordonnées 2D correspondantes sur une image issue de la caméra. Une fois cette relation établie, l'information 3D peut être inférée à partir du 2D et vice-versa. La calibration est donc une opération prérequise à toute application pour laquelle cette relation entre images 2D et monde 3D est nécessaire. Une fois une caméra choisie, le problème de sa calibration consiste à calculer ses paramètres numériques particuliers.

On peut modéliser une caméra par une matrice 4×4 qui représente la transformation (incluant la plupart du temps une projection) entre le système de coordonnées du monde et le système de coordonnées de l'image. Cette matrice contient les paramètres intrinsèques et extrinsèques de la caméra qu'elle représente. Lorsque cette matrice est connue ou peut être dérivée car les paramètres de la caméra ont été mesurés, on dit que la caméra est *calibrée*. Si on ne peut mesurer les paramètres ou si la transformation due à la caméra est complexe (par exemple, non-linéaire), on peut tenter de trouver une matrice qui va constituer une *approximation* de la transformation effectuée par la caméra. Pour certaines applications, une telle approximation peut être suffisante.

Certaines techniques mesurent à l'avance tous les paramètres de la caméra utilisée (plus d'une caméra pouvant compliquer la calibration exacte) et contrôlent exactement tout déplacement ou changement d'orientation. Cette façon exhaustive de procéder calibre complètement la caméra mais est assez dispendieuse et contraignante dû au coût élevé du matériel et des caméras utilisées, souvent très encombrantes et fragiles. De plus, c'est un processus délicat et instable qu'il faut recommencer

[5] Modèle de caméra dont l'hypothèse principale est que la relation entre les coordonnées du monde et les coordonnéees des pixels est linéaire projective. C'est une caméra qui n'effectue aucune distorsion et dont le centre de projection est un point. On peut utiliser les outils de la géométrie projective pour modéliser ses effets.

à chaque fois que la caméra est déplacée ou que ses paramètres sont modifiés. D'autres techniques dérivent la caméra empiriquement à partir d'un ensemble de points 3D de coordonnées connues et de leurs correspondants sur les images.

Si on est capable de retrouver une matrice de transformation pour la caméra et que l'on connaît le modèle de caméra utilisé ou supposé (par exemple, une caméra de type sténopé), il est alors possible d'extraire de la matrice les différents paramètres de la caméra. Cette opération est appellée la *décomposition* de la matrice de transformation. Strat [Str87] et Ganapathy [Gan84] se sont penchés sur le problème. Le premier utilise une interprétation géométrique de la matrice en la décomposant en différentes transformations afin d'en extraire les divers paramètres. Le second a développé une méthode algébrique qui résoud un système non-linéaire de onze équations pour obtenir les onze coefficients indépendants de la matrice de transformation de la caméra. Les deux techniques supposent une lentille et un environnement idéaux qui n'introduisent aucune distorsion dans les images. Lorsque le modèle supposé est incorrect, la précision de la décomposition diminue car la matrice retrouvée est en fait une approximation de la transformation effectuée par la caméra.

Comment calibrer une caméra ?

La calibration ou *étalonnage* de caméra est un problème qui a été grandement étudié dans les domaines de la photogrammétrie et de la vision. Un nombre important de techniques très diverses a été proposé.

Lorsqu'une caméra est calibrée, on connaît la transformation de projection entre le système de coordonnées du monde et le système de coordonnées de l'image. Cette transformation peut parfois s'exprimer sous forme d'une matrice ou être approximée par une matrice. Étant donné que la calibration *exacte* d'un système d'acquisition d'images nécessite un appareillage de mesure précis, on dispose souvent d'images non calibrées et il faut alors tenter de retrouver la meilleure approximation de la matrice représentant la caméra.

On peut classifier les méthodes de calibration en deux grandes classes, la calibration traditionnelle et la calibration automatique (auto-calibration) ou semi-automatique.

Calibration traditionnelle

Les méthodes classiques de calibration [FT86] [Rob96] [Fau93] sont basées sur l'observation d'un objet 3D dont la géométrie est connue (*model-based*). On connaît alors les coordonnées 3D $P_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ de quelques points de référence sur l'objet dans un système de coordonnées

propre (local) à l'objet. Les projections p_i de ces points sont mesurées dans le repère des images et on obtient des coordonnées de pixels $p_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$. Les objets de référence généralement utilisés sont des grilles de calibration composées de motifs répétitifs (cercles ou rectangles) choisis afin de définir des points d'intérêts dont les coordonnées image peuvent être mesurées avec une plus grande précision. On appelle ces objets des *mires de calibration* (dont on peut voir un exemple à la figure 2.2). Les paramètres extrinsèques se réduisent donc au déplacement entre le système de coordonnées de la mire (que l'on prend comme étant identique au système de coordonnées du monde) et le système de coordonnées de la caméra. Lorsque l'on dispose de plusieurs points, il est possible d'estimer indépendamment la matrice de transformation perspective pour chaque caméra. Cette méthode est souvent utilisée pour calibrer des systèmes de stéréovision et semble assez efficace. Par contre la précision de la calibration dépend bien sûr de la précision de la grille et de celle des points d'intérêts choisis.

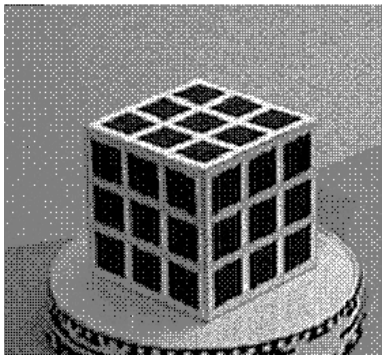


FIG. 2.2: Exemple d'une mire de calibration (de Szeliski [SK94]).

Calibration automatique ou semi-automatique

Il existe plusieurs applications pour lesquelles une mire de calibration n'est pas disponible et où l'on ne dispose d'aucune information sur les paramètres de la caméra. On se trouve en présence d'*images non calibrées*. Les méthodes qui permettent de résoudre le problème du calcul du mouvement d'une caméra supposent que seuls les paramètres intrinsèques sont connus. Il existe de telles méthodes fondées sur des points, notamment la méthode de huit points avec deux vues de Longuet-Higgins [LH81].

Lorsque les paramètres intrinsèques sont inconnus, plusieurs chercheurs calibrent leurs caméras en utilisant des outils standards de *géométrie projective*. Ce sont le plus souvent des contraintes géométriques qu'ils identifient dans les images. L'*auto-calibration* ne nécessite donc aucun appareillage de calibration et aucune connaissance a priori sur le système d'imagerie ou la scène, si ce

n'est une liste de primitives 2D mises en correspondance entre différentes vues.

La plupart des méthodes d'auto-calibration actuelles utilisent des correspondances établies automatiquement ou manuellement entre les primitives des images [FLM92]. Liu *et al.* [LHF90] retrouvent la position de la caméra à l'aide de correspondances entre six points ou huit lignes (algorithme linéaire), ou entre trois points ou trois lignes (algorithme non-linéaire). De même, Faugeras et son équipe [Fau92] [Fau93] démontrent qu'il est possible de reconstruire des scènes en trois dimensions à partir de correspondances entre points seulement mais qu'une telle reconstruction est définie à une transformation projective près (facteur d'échelle). En se basant sur la théorie de la géométrie épipolaire, il démontre que la calibration complète de la caméra n'est pas nécessaire pour obtenir une reconstruction utile d'une scène vue par un système stéréo. La *contrainte épipolaire* est une contrainte tirée de la géométrie projective qui relie deux caméras ensemble et se retrouve facilement à partir de simples correspondances entre points. Elle provient de l'existence de deux points de vue d'une même scène. Pour un point dans une image, la géométrie épipolaire nous donne une ligne 1D dans l'autre image. Cela permet de réduire la "dimensionnalité" du problème de correspondance 2D initial à une recherche sur une ligne plutôt que dans toute l'image. Elle est de plus en plus utilisée dans le domaine de la vision. La figure 2.3 ainsi que les applications présentes sur le site *web* de Bougnoux [Bou] expliquent bien ce qu'est la contrainte épipolaire.

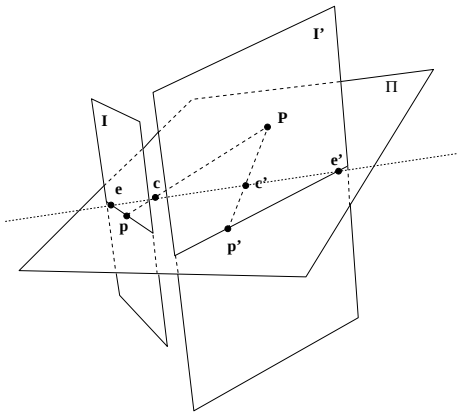


FIG. 2.3: La géométrie épipolaire. Pour deux images (I et I') d'une même scène, il existe un plan Π passant par les centres de projection (c et c') de ces images et tout point 3D P de la scène. Ce plan coupe éventuellement les images (donc les plans de projections) en deux lignes : les lignes épipolaires. La projection de P dans chaque image se trouve sur la ligne épipolaire correspondante. Toutes les lignes épipolaires d'une image se coupent en un ou plusieurs *épipoles* (e et e').

Luong et Faugeras [LF93] calibrent automatiquement des systèmes de stéréovision à deux ou trois caméras en utilisant les relations qui existent entre elles (*trilinéarités* ou contraintes trilinéaires). Ils n'utilisent aucune mire et font tous leurs calculs dans le système de coordonnées des caméras,

en utilisant comme cadre de référence la première caméra. Ils tentent de déterminer l'ensemble des paramètres intrinsèques et extrinsèques des caméras, ces derniers constituant le déplacement relatif entre deux ou trois caméras, calculé dans le système de coordonnées de la première caméra. Étant donné qu'ils ne disposent d'aucune information métrique, ce déplacement est calculé à un facteur d'échelle près. Ils utilisent des appariements de primitives (points d'intérêts) obtenus au cours de déplacements inconnus, sans aucune connaissance a priori sur les scènes observées. Par contre, leur méthode ne donne des résultats comparables à ceux des méthodes traditionnelles que si les points qu'ils mettent en correspondance à travers les images le sont avec une très haute précision.

En pratique, la détermination automatique de correspondances pose encore de sérieux problèmes aux algorithmes de vision comme nous l'avons vu précédemment. La plupart des algorithmes de calibration qui se basent sur ces correspondances sont donc souvent assistés par l'usager pour obtenir de bons résultats. Celui-ci indique manuellement les appariements de primitives à travers les images. C'est ainsi que le système *TotalCalib* [BR97] calibre des séquences d'images de façon semi-automatique. L'usager indique des correspondances pour le calcul des matrices de projection et le système lui fournit plusieurs outils pour rendre sa tâche moins fastidieuse.

Malheureusement, la plupart des techniques de vision voulant calibrer automatiquement leurs caméras à partir de correspondances supposent que la segmentation, la détection des contours et la mise en correspondance de primitives sont déjà effectuées ou ne posent aucun problème. Mais ces étapes constituent des sources de difficultés importantes pour un algorithme de reconstruction automatique qui se trouve confronté à des problèmes tels le bruit dans les images ou les ambiguïtés d'appariements. Afin d'illustrer les résultats typiques des techniques actuelles, la figure 2.4 montre le résultat d'une détection de contour. Les segments sont souvent incomplets et non droits. Les polygones ne sont pas fermés et la différence entre les discontinuités géométriques, de texture et d'illumination n'est souvent pas facilement discernable.

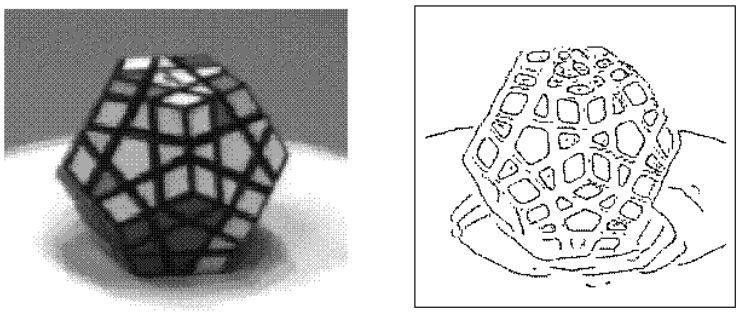


FIG. 2.4: Résultat de la segmentation d'une image (de Szeliski [Sze93]).

Pour l'instant, il ne semble pas exister de méthode robuste et complètement automatique pour une technique d'auto-calibration basée sur les correspondances.

Il est à noter que la méthode de calibration choisie dépend de l'application visée. Dans la méthode de reconstruction que nous proposons dans ce mémoire, nous n'utilisons pas une méthode de calibration complexe telles que celles décrites dans cette section. Nous ne cherchons à retrouver aucun paramètre de caméra mais seulement une matrice de transformation qui nous permet de passer des coordonnées 3D de primitives dans l'espace à leurs coordonnées 2D sur le plan de projection. Cette matrice constitue une approximation plus ou moins précise de la vraie caméra mais nous verrons que cette calibration est suffisante pour nos besoins. Nous n'introduisons donc pas une nouvelle méthode de calibration. Dorénavant, lorsque nous utiliserons le mot *caméra*, nous ferons référence à cette matrice d'approximation.

2.2 Systèmes de reconstruction à partir d'images

Il existe un besoin réel pour une méthode pratique d'acquisition de modèles réalistes à partir d'images de modèles existants. Nous avons pu constater que plusieurs techniques dites automatiques se heurtent à des problèmes importants tels la segmentation des contours ou la mise en correspondance. Nous considérons que vouloir tout faire automatiquement, tout en restant général, est pour l'instant utopique.

Sacrifier cet automatisme au profit de meilleurs résultats est un compromis intéressant. L'intégration de l'utilisateur au sein du processus de reconstruction permet d'éviter des erreurs du système qui sont souvent fatales à la bonne réussite de la reconstruction. Plusieurs équipes ont donc décidé de faire ce sacrifice et offrent à l'usager plusieurs outils pour lui permettre de "guider" la reconstruction dans la bonne direction. Nous présentons dans cette section quelques systèmes représentatifs appartenant à divers domaines de recherche.

2.2.1 Vision/infographie

Les domaines de la vision et de l'infographie sont actuellement tous deux intéressés à l'obtention de modèles réalistes à partir d'images. Autrefois opposés, l'infographie tentant de construire des objets et la vision tentant de les reconstruire à partir de leurs images, les deux domaines commencent à associer de plus en plus leurs techniques pour pallier aux problèmes qu'ils rencontraient jusque-là.

La génération de modèles réalistes en infographie est un processus qui prend beaucoup de temps,

autant au niveau de la modélisation de structures géométriques complexes que du rendu des effets subtils de lumière et d'ombre. La simulation des détails géométriques est souvent réalisée en appliquant une texture sur un modèle plus "grossier" de l'objet. Quant à la vision, nous avons vu plus tôt les problèmes rencontrés lors de reconstructions automatiques. En associant certaines techniques de chaque domaine, on peut obtenir des reconstructions de qualité.

***Façade* : Modélisation et rendu d'architecture à partir de photographies**

Debevec *et al.* [DTM96] présentent une méthode très intéressante qui permet de modéliser et d'effectuer le rendu de scènes architecturales à partir de quelques images prises de points de vue arbitraires. *Façade* est un système interactif de modélisation qui permet à un usager de construire un modèle géométrique de la scène à l'aide de photographies digitalisées en exploitant les contraintes inhérentes à des scènes de type architectural (parallélisme, symétrie, planarité, etc.).

Pour effectuer sa reconstruction, *Façade* utilise des primitives appelées *blocs* (figure 2.5 (a)). Chaque bloc contient un certain nombre de paramètres (spécifiant sa taille, sa forme et sa position) que le système tente de retrouver, en même temps que les paramètres des caméras. Pour cela, l'usager doit modéliser la scène à l'aide d'un modeleur intégré au système. Ce modeleur contient un certain nombre de ces blocs paramétrisés que l'usager peut choisir et placer dans sa scène 3D en construction. L'usager établit ensuite des correspondances entre les segments de ces blocs et leurs projections respectives dans une ou plusieurs images. Il peut également spécifier d'autres contraintes comme des contraintes de positionnement relatif entre blocs (tel bloc au-dessus de tel bloc), de symétrie, de position et de taille, etc. De telles contraintes permettent de réduire grandement le nombre de correspondances à établir et de paramètres à résoudre. En général, une bonne reconstruction est obtenue s'il y a autant de correspondances dans les images qu'il y a de degrés de liberté dans les caméras et le modèle.

Façade ne considère donc pas une scène comme un ensemble de points dont il faut retrouver les coordonnées mais plutôt comme un ensemble de blocs dont on cherche les paramètres (figure 2.5 (b)). Les coordonnées des points, lignes et polygones sont implicitement contenues dans ces blocs (combinaison linéaire des paramètres du bloc). Le modèle est représenté par une hiérarchie de tels blocs qui contient leurs paramètres ainsi que leurs relations spatiales (figure 2.5 (c)).

Une fois toutes ces contraintes spécifiées, le système tente de retrouver les paramètres libres en effectuant la minimisation d'une fonction objectif à l'aide d'une méthode non-linéaire d'optimisation. Les reconstructions obtenues semblent assez précises, comme par exemple sa reconstruction du

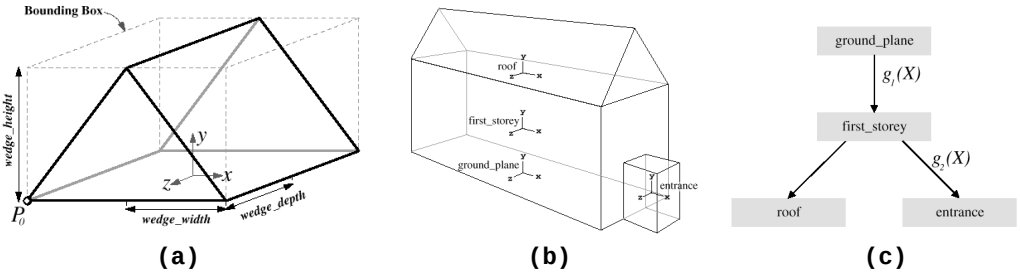


FIG. 2.5: Exemple d'éléments du système *Façade* (de Debevec [DTM96]).

campus de l'Université Berkeley en Californie [Deb]. *Façade* permet de reconstruire des modèles polyédriques et traite également les surfaces de révolution [Bor97]. Le modèle reconstruit permet ensuite de contraindre un algorithme stéréo de manière efficace afin de récupérer les détails qu'il aurait été trop long et difficile de modéliser à l'aide de ces blocs de base.

Sa technique de *modélisation photogrammétrique* est donc très intéressante même si elle nécessite l'utilisation d'un modeleur à la base, ce qui la classifierait peut-être plus comme une méthode de *construction* à partir de photographies. Par ailleurs, on pourrait voir un certain manque de généralité au système car si une géométrie ne correspond à aucun bloc disponible dans le système, il faut définir un nouveau type de bloc.

Yedid *et al.* [MYTG94] utilisent le même genre d'approche interactive pour construire des objets d'une scène à partir de différentes images de cette scène. Ils commencent par déterminer la matrice de transformation par une technique de calibration interactive à base de primitives telles que des rectangles ou des segments. Puis, à l'aide d'un modeleur, l'usager peut créer des modèles en fil de fer avec des primitives déformables (cubes, sphères, cônes, etc.) dont la projection est superposée, durant leur construction, sur les images calibrées. Afin d'éviter des erreurs de construction dues à la profondeur, chaque nouvelle primitive construite doit avoir au moins un point de contact avec une primitive bien placée. La représentation de scène utilisée est hiérarchique et son degré de précision est déterminé par l'utilisateur.

***REALISE* : Reconstruction de modèles de bâtiments à partir d'images non calibrées**

Le projet *REALISE* [FLR+95] [Lbar96] réunit plusieurs équipes de recherche européennes. Son but principal est d'extraire, de plusieurs images d'une scène, les informations nécessaires à des simulations en réalité virtuelle. Le projet vise l'utilisation d'images non calibrées qu'il exploite afin de retrouver les descriptions géométriques (projective, euclidienne et affine) et photométriques de la

scène qu'elles représentent. Là encore, un système interactif de reconstruction demande à l'usager de spécifier un certain nombre de contraintes. *REALISE* mise donc également sur une association des communautés de vision et d'infographie pour produire des données 3D réalistes. Son but ultime est de procurer à un usager en train de bâtir un modèle à partir d'images, de bonnes techniques de vision pour assister sa tâche.

Le système commence par établir une série de correspondances entre les images afin de retrouver la géométrie épipolaire qui leur est associée. Ensuite, il estime les matrices de projection perspective de chacune des images choisies par l'usager. Ces matrices sont définies à une transformation projective près. Ceci reflète le fait que si l'on dispose seulement de correspondances entre images, on ne peut retrouver que la *géométrie projective* de la scène 3D, i.e. les propriétés qui sont invariantes après des transformations projectives.

Si on le désire, on peut ensuite estimer la *géométrie affine* de la scène, i.e. les propriétés de la scène qui sont invariantes sous l'effet de transformations affines 3D, telles que le parallélisme entre lignes ou des ratios de segments colinéaires. Il utilise ce parallélisme pour calculer le "plan à l'infini" qu'il dérive des points de fuite des images à partir d'au moins trois ensembles de directions de lignes non coplanaires.

Enfin, il peut estimer la *géométrie euclidienne* de la scène, i.e. les propriétés de la scène qui sont invariantes sous l'action de transformations euclidiennes (par exemple, les notions de longueurs) en utilisant des informations sur la scène comme des angles entre lignes, entre plans, des ratios de segments, etc. L'extraction de paires de lignes orthogonales permet de "rectifier" la base de coordonnées affine. Plus d'information sur ces trois types de géométrie se retrouve dans l'article de Faugeras [Fau95]. Ici encore, on exploite le fait que le genre d'information nécessaire à la récupération des différents niveaux de géométrie se retrouve facilement dans les scènes à reconstruire.

Une observation des modèles obtenus par ce système nous a permis de constater que plusieurs comportaient des trous, un manque de symétrie en plusieurs endroits ainsi que des polygones non planaires.

L'intégration d'outils de vision, au sein de ce système, afin d'aider l'usager (par exemple, l'indication des lignes épipolaires dans toutes les images lorsque l'on sélectionne un point dans une image) nous semble être utile. En ce qui concerne la contrainte épipolaire, la position d'un point est contrainte à être sur une ligne spécifique de l'image. Il n'est pas clair jusqu'à quel point cela réduit effectivement l'interaction de l'usager mais cela peut s'avérer pratique lorsque le point que l'on cherche à placer se trouve à être caché sur l'image mais que sa position peut être "interpolée" visuellement par l'usager.

***PhotoModeler* : un système commercial de photogrammétrie digitale**

Le système *Photomodeler* [Pho] est un logiciel commercial qui permet d'effectuer des mesures photogrammétriques sur des modèles reconstruits à partir de photographies. La calibration de caméra s'effectue manuellement en indiquant tous les paramètres de la caméra utilisée ou automatiquement si certaines propriétés sont présentes dans les images (par exemple, une primitive cubique à indiquer). Le système de reconstruction est interactif. L'usager doit placer des primitives (points et triangles) sur les images puis les mettre en correspondance. Le système calcule alors le modèle 3D correspondant.

Les modèles obtenus semblent satisfaisants lorsqu'une texture leur est appliquée, mais en les regardant de plus près avec un affichage de style "fil de fer", on constate qu'ils comportent un nombre beaucoup plus élevé que nécessaire de triangles, dont beaucoup sont dégénérés[6]. Nous avons aussi observé la présence de trous. Ces défauts compliquent, voire empêchent les traitements typiques en infographie, tel le rendu du modèle. La figure 2.6 montre le modèle d'une imprimante obtenu à l'aide de quatre images.

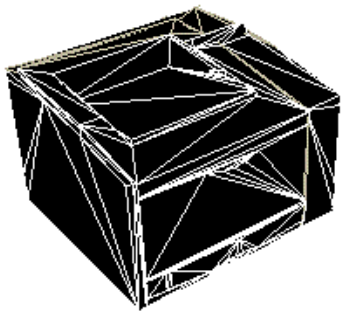


FIG. 2.6: Modèle d'imprimante obtenu avec *Photomodeler.*

Le processus de reconstruction semble également assez long et fastidieux. On doit d'abord placer les points, puis utiliser ces points pour former des triangles. Tel que remarqué par Hanke [Han96], la précision des mesures de distances et de coordonnées obtenues sur un modèle reconstruit est très élevée. Par contre, cette qualité de précision peut être obtenue seulement si on utilise des images de haute résolution (lors de l'étude, les images avaient une résolution d'environ 3000×4000 pixels) et une configuration optimale spécifique des positions des caméras, celles-ci devant être complètement calibrées (incluant les paramètres de distorsion). D'autres facteurs dont le type de caméra utilisé et son utilisation elle-même, l'objet reconstruit, la définition des points mesurés, la qualité des images, etc. influencent également la précision finale.

[6] Allongés et étroits.

2.2.2 Intelligence artificielle

AIDA : Un système de reconstruction à base de connaissances

Le système de reconstruction *AIDA* [LGG95] [WG96] se veut entièrement automatique (même si, dans la pratique, tous les résultats montrés ont nécessité une interaction manuelle) et est destiné à la génération de modèles 3D pour des applications graphiques 3D. Ses concepteurs ont remarqué que la reconstruction automatique de scènes à l'aide de plusieurs images donne souvent des anomalies comme des murs ou des rues non planaires, des portes et fenêtres non perpendiculaires, des murs non connectés, etc. Cela les a conduit à introduire une base de contraintes contenant des connaissances concernant les objets que l'on s'attend à rencontrer dans les scènes traitées, des propriétés géométriques 3D et 2D (telles qu'elles apparaissent dans des images provenant de caméras), des relations entre objets, etc. Ces relations sont organisées dans un graphe sémantique. Cette base de contraintes est exploitée une fois les images calibrées et segmentées avec des processus de vision traditionnels. Chaque contrainte est représentée par une fonction de coût *pondérée* et la minimisation de leur somme permet d'obtenir une description de surface qui satisfait le mieux possible l'information mesurée ainsi que les contraintes dérivées de la base de contraintes. Après avoir effectué une phase d'interprétation, le système sélectionne automatiquement ces contraintes et les applique à la surface en reconstruction.

Le système, bien que limité par son domaine d'application (bâtiments), permet d'apprécier l'utilité et les améliorations obtenues par l'introduction de contraintes de planarité, parallélisme, perpendicularité, d'angle, etc. ainsi qu'une manière de les spécifier. Par contre, la base de connaissances doit être adaptée au type de scènes à modéliser. Elle est spécifique au domaine concerné et utilise donc de l'information a priori. Ceci représente une exigence restrictive. L'interprétation de la scène créant des associations automatiques de primitives pourrait également introduire des erreurs difficilement corrigibles. Une approche qui permettrait de laisser l'usager choisir les contraintes à appliquer à partir d'un ensemble de contraintes disponibles serait plus générale et plus sûre.

2.3 Au sujet des contraintes

Le mot contrainte a plusieurs définitions et est employé de diverses façons à travers la littérature scientifique. Dans notre contexte, une contrainte est une relation numérique ou géométrique entre des objets. Les contraintes possèdent une nature déclarative et constituent une manière naturelle de décrire les relations entre les objets. Les contraintes ne sont pas visibles en tant que tel, seuls leurs

effets le sont.

Une littérature exhaustive existe sur l'utilisation de contraintes en infographie. Elle concerne surtout la conception et l'utilisation d'interfaces graphiques à l'aide de contraintes. Sutherland a été l'un des pionniers de l'idée avec son système *SketchPad* [Sut63]. Il pouvait spécifier dans *SketchPad* le parallélisme entre lignes 2D et le système "relâchait" les dessins jusqu'à ce que les contraintes soient satisfaites. Il essayait ensuite de maintenir la contrainte au cours des manipulations suivantes.

Gleicher [Gle94] a étudié la manipulation de contraintes de type graphique ainsi que la plupart des problèmes qu'elles peuvent engendrer. Il considère que les contraintes sont un moyen puissant pour restreindre le comportement d'objets graphiques ainsi que pour spécifier des relations entre objets. Tout comme les objets peuvent être vus comme des entités physiques, les contraintes peuvent être vues comme des interactions mécaniques. Les aspects qu'il adresse concernent plus les systèmes interactifs de création de modèles graphiques mais certains principes sont toutefois intéressants à considérer pour des projets similaires au nôtre.

2.4 Approche proposée : le système Rekon

Nous proposons une approche au problème différente à plusieurs niveaux de celles actuellement existantes. Tout comme pour les concepteurs de *Façade* et *REALISE*, nous sommes conscients des problèmes rencontrés par les algorithmes de vision et proposons d'intégrer l'usager dans le processus afin de résoudre les problèmes auxquels ces algorithmes se heurtent le plus souvent. En effet, celui-ci peut aisément différencier les primitives dans une image ainsi que décrire les relations qui existent entre elles puisqu'il possède la faculté de vision tridimensionnelle[7]. De plus, tel que démontré par les méthodes développées en intelligence artificielle, l'ajout de contraintes géométriques 3D lors de la reconstruction permet d'obtenir une nette amélioration du modèle et justifie donc leur utilité. Étant donné qu'un humain est en mesure d'effectuer des jugements qualitatifs sur ce qu'il voit, nous proposons également d'intégrer ce type de contraintes. Elles seront spécifiées par l'usager et leur but sera de produire des modèles subjectivement meilleurs.

Nous avons donc réalisé un système *interactif* général pour reconstruire la géométrie 3D d'une scène à partir d'un ensemble d'images de cette scène prises avec une ou plusieurs caméras dont les paramètres sont arbitraires. Ces images peuvent être des photographies, des images de synthèse, des plans, des peintures, des radiographies, etc. On peut le considérer comme un *système interactif de*

[7] La "reconstruction automatique" dans notre esprit des scènes que nous voyons est un phénomène étudié par plusieurs, dont Marr [Mar82].

modélisation photogrammétrique.

Afin de calibrer nos images, nous utilisons une méthode très simple. Nous ne cherchons à calculer aucun paramètre de caméra, extrinsèque ou intrinsèque. Nous voulons plutôt simplement retrouver une matrice approximant la projection perspective ou la série de transformations ayant abouti à l'image. Cette calibration s'effectue habituellement à l'aide de six points dont les positions 3D relatives sont connues, soit parce qu'elles ont été explicitement spécifiées par l'usager ou calculées par le système. Si la position réelle de ces points est inconnue, la reconstruction s'effectue à un facteur d'échelle près. Nous ne sommes pas contraints d'utiliser des points pour la calibration, d'autres primitives et/ou paramètres peuvent également servir au calcul de la matrice. Notre méthode de calibration sera décrite plus en détails au chapitre 3.

Le fonctionnement de notre système de reconstruction est lui aussi assez simple. On demande à l'usager de tracer des primitives sur les images puis d'établir entre elles des relations de natures différentes qui contraignent leurs positions dans l'espace. Ces contraintes seront décrites plus en détails aux chapitres 3 et 4.

Une fois deux vues calibrées, on peut obtenir la géométrie 3D des primitives dessinées et mises en correspondance sur ces images par l'usager. Ces nouvelles positions vont éventuellement permettre de calculer de nouvelles matrices de projection. Des itérations successives entre ces deux étapes permettent d'améliorer tour à tour le modèle obtenu ainsi que les matrices de projection. On parle ainsi de reconstruction interactive *incrémentale* et *itérative*.

Par contre, les modèles obtenus uniquement à l'aide de correspondances entre primitives ne sont pas toujours visuellemement satisfaisants. Tout comme *Façade* [DTM96] dont le système de modélisation est effectif et robuste car il exploite les contraintes qui sont caractéristiques des scènes architecturales, nous nous proposons d'utiliser le même genre de contraintes pour améliorer nos modèles. Par contre, contrairement à *Façade* qui manipule des équations non-linéaires, les contraintes que nous avons introduites sont exprimées sous forme d'équations linéaires pour des raisons de simplicité et de robustesse et afin d'être compatibles avec le reste du système de reconstruction.

Les modèles résultant de l'ajout de ces contraintes sont nettement plus précis et correspondent mieux à la réalité. Pour l'instant, les scènes reconstruites ne sont que des scènes polyédriques mais ces scènes forment déjà un très large éventail de scènes que nous voudrions reconstruire : objets polyédriques, scènes urbaines et architecturales, etc. La reconstruction de surfaces courbes est un problème plus ardu dont nous discuterons en conclusion. Nous ne nous sommes également pas attachés à l'extraction des textures des images, afin de les appliquer aux modèles récupérés. Cela fait

l'objet d'un projet de recherche conjoint [Oui98] et constitue une partie essentielle à l'obtention de scènes photoréalistes.

Nous exprimons nos contraintes à l'aide d'équations linéaires qui engendrent des systèmes d'équations surdéterminés. Afin de solutionner efficacement de tels systèmes, nous utilisons une technique de résolution bien connue : la décomposition en valeurs singulières (*singular value decomposition* ou *SVD*) [PFTV92]. Celle-ci garantit de toujours nous fournir la "meilleure" solution au sens des moindres carrés, c'est-à-dire une solution qui minimise l'erreur globale du système. Les contraintes nous servent autant à calibrer nos vues qu'à déterminer la position de nos primitives 3D avec le moins d'erreurs possibles. On peut donc voir notre méthode comme la résolution d'un problème d'optimisation contraint. On veut trouver le meilleur ensemble de positions pour nos primitives qui satisfasse notre ensemble de contraintes.

Nous présentons donc un système simple, général, efficace et robuste pour effectuer de la modélisation photogrammétrique interactive. De plus, la méthode de résolution utilisée (*SVD*) nous fournit toujours une solution. Nous visons la simplicité et la généralité à plusieurs niveaux. Tout d'abord, notre technique de calibration ne nécessite aucune connaissance préalable quant aux caméras utilisées. Plusieurs sortes de caméras, réelles ou synthétiques, peuvent être employées pour produire des images d'une même scène, et leurs paramètres peuvent varier d'une image à l'autre (distance focale, *zoom*, profondeur de champ, position, orientation, etc.). Tous ces paramètres sont implicitement contenus dans la matrice de projection que nous obtenons. Évidemment, les en extraire serait une tâche plus difficile mais notre technique de reconstruction ne le requiert pas.

Étant donné que nous nous basons essentiellement sur l'intervention de l'usager dans le processus de reconstruction, la qualité des images utilisées ne semble pas aussi importante que pour certaines des techniques décrites précédemment. En effet, nous avons réalisé nos tests avec des images de faible résolution (640×480 pixels) occupant environ 50 kilooctets en mémoire (avec encodage *JPEG*). Les résultats que nous obtenons sont comparables à ceux produits par des systèmes utilisant des images de très haute résolution (3271×4165 pixels) [Pho] dont l'espace de stockage pour chacune est de l'ordre de plusieurs mégaoctets. Le placement de primitives par l'usager ainsi que les contraintes qu'il spécifie compensent largement le manque éventuel de qualité d'image. Également, contrairement à plusieurs systèmes existants, nous pouvons créer des modèles satisfaisants à partir de quelques images seulement. L'espacement des caméras peut être très grand, ce qui permet une récupération plus précise de la profondeur et l'utilisation de moins d'images. De plus, l'extraction des détails peut se faire aussi finement que le permet la résolution des images.

Enfin, au niveau de notre méthode de reconstruction, nous utilisons des principes simples de géométrie projective afin d'exprimer nos différentes contraintes sous forme de systèmes d'équations linéaires. Ces systèmes sont souvent plus robustes que leurs équivalents non-linéaires, plus faciles à résoudre et ne nécessitent pas de solution initiale. La méthode de résolution adoptée (*SVD*) est une méthode fiable et connue qui produit toujours une solution.

L'interactivité avec l'usager est un élément crucial dans notre méthode. En effet, c'est à lui que revient la tâche de résoudre les conflits qui peuvent survenir lorsqu'un système essaie d'analyser seul des images ; il constitue en quelque sorte le "système de vision intelligent" de notre système de reconstruction. Les avantages d'utiliser l'usager pour la spécification des différentes relations entre les primitives sont multiples. Tout d'abord, un usager peut facilement éviter les difficultés et ambiguïtés auxquelles font face les systèmes de vision qui se veulent automatiques. Ces difficultés sont malheureusement encore nombreuses et incluent les problèmes de segmentation, les erreurs de mises en correspondance de primitives, les parties cachées, les pertes d'information dues à des perspectives défavorables (*foreshortening*), les ombres, les textures ou leur absence, les spécularités, les réflexions miroirs, le bruit dans les images, les différences causées par l'utilisation de plusieurs types de caméra, les problèmes causés par des images espacées dans l'espace ou dans le temps, etc. Également, si le modèle obtenu ne satisfait pas l'usager, il peut choisir d'y appliquer certaines contraintes pour le raffiner. Il peut facilement juger quelles contraintes seraient susceptibles d'améliorer les anomalies qu'il perçoit sur certaines parties du modèle et ainsi les corriger de manière appropriée. Il effectue un jugement qualitatif de la reconstruction et le système s'occupe de la partie quantitative, c'est-à-dire des calculs nécessaires à l'obtention des positions des diverses primitives en accord avec les indications de l'usager.

De plus, l'usager peut *juger* ce qui est important pour lui dans une scène et ainsi spécifier exactement ce qu'il désire reconstruire. Il sait ce qu'il veut modéliser et avec quelle précision il aimerait le faire. Ainsi, il peut décider de concentrer ses efforts dans la reconstruction géométrique d'un modèle compliqué, ou bien se contenter d'en faire une approximation géométrique suffisante dont les détails seront simulés par une texture. Si les images sont chargées d'objets et de détails, l'usager peut facilement discerner les objets qui l'intéressent parmi les autres. Un système automatique se perdrait dans la multitude de segments et pourrait difficilement effectuer une segmentation correcte. De plus, il serait très ardu de spécifier automatiquement à un système de vision le modèle particulier que l'on désire extraire d'une image complexe. Il est intéressant de remarquer que les images fournissent une sorte de "patron" que l'usager n'a qu'à tenter de suivre pour la reconstruction de certains modèles com-

plexes à créer avec un modeleur. L'application d'une texture sur le modèle résultant pourra cacher les éventuels défauts et donner l'apparence d'un modèle approprié aux besoins de l'infographie.

Évidemment, un système interactif est synonyme de temps. Il est en effet plus long et fastidieux de dessiner des primitives sur des images, surtout qu'en moyenne il nous en faut au moins deux fois plus que le nombre de primitives 3D à reconstruire puisqu'on doit la plupart du temps les mettre en correspondance pour obtenir leur position. Il faut de plus spécifier une à une les contraintes qui les relient. Le but de cette recherche n'est pas de créer un système efficace au point de vue temps de reconstruction mais plutôt efficace au niveau de la reconstruction elle-même. Il nous permet de vérifier l'effet de l'introduction de nouvelles contraintes sur les modèles obtenus. Une fois l'utilité de notre méthode démontrée, il sera possible et sûrement souhaitable d'automatiser certaines des tâches les plus fastidieuses. Ainsi, l'implantation d'une boîte à outils provenant des domaines de traitement et d'analyse d'images ainsi que de synthèse d'images pourra être considérée. Elle contribuera à alléger la tâche de l'usager et ainsi à raccourcir le temps de modélisation. Il devrait même être possible de raccorder le système directement à des systèmes de vision ou des modeleurs existants. De plus, ces différents outils permettront d'améliorer la précision au niveau du placement des primitives sur les images. En effet, celle-ci dépend largement de la qualité des images et de la fiabilité de l'usager.

Dans ce qui suit, nous expliquons plus en détails les principes et le fonctionnement de notre système de reconstruction ainsi que les différentes contraintes utilisées. Nous pourrons constater que l'introduction de nouvelles contraintes permet d'obtenir des modèles plus précis qu'avec de simples correspondances entre primitives.

Chapitre 3

Le système Rekon

"L'homme n'est point fait pour méditer, mais pour agir."

—Jean-Jacques Rousseau

Nous avons implanté un système interactif de reconstruction à partir de photographies, appelé *Rekon.* Dans ce chapitre, nous en présentons les principes de base, qui découlent principalement de la géométrie projective. Nous détaillons les structures principales utilisées ; les caméras ainsi que différentes primitives géométriques de bas niveau. Finalement, nous décrivons le fonctionnement du système, i.e. les rôles respectifs de l'usager et du système dans la reconstruction. Le prochain chapitre adressera plutôt les contraintes que nous y avons ajoutées afin d'améliorer les modèles reconstruits par le système de base.

3.1 La géométrie projective

Il existe plusieurs sortes de géométries dont les plus connues sont la géométrie euclidienne, la géométrie affine et la géométrie projective. La plupart des principes à la base de notre système découlent de la géométrie projective et sont relativement simples. L'appendice du livre de Mundy et Zisserman [MZ92] ainsi que le livre de Semple et Kneebone [SK52] expliquent bien les principes de cette géométrie.

La différence principale entre la géométrie projective et la géométrie affine est le concept de parallélisme. En géométrie affine, deux lignes (plans) sont considérés parallèles si ils ne s'intersectent pas. En géométrie projective, deux lignes (plans) s'intersectent toujours en un point (une ligne) [Han93] [SK52]. Ce point et cette ligne se trouvent à l'infini. Ainsi, un point à l'infini représente une direction dans l'espace puisqu'il est le point d'intersection de toute une famille de droites parallèles.

Les notions de distance, parallélisme, symétrie, angle, etc. ne sont plus conservées sous leur forme euclidienne et peuvent donc difficilement être exploitées par des méthodes de reconstruction automatiques. Par exemple, certains tentent de détecter le parallélisme en cherchant les points de fuite. Avec notre capacité de voir en perspective et notre connaissance des scènes nous entourant, nous pouvons facilement extraire ces relations des projections 2D de ces scènes. En effet, le fait de voir en perspective nous a amené naturellement à développer des aptitudes pour comprendre et extraire l'information 3D à partir d'images résultant de projections perspectives.

3.1.1 L'incidence

La notion clé que nous allons utiliser le plus souvent est que la géométrie projective conserve l'incidence (tout comme la colinéarité). On dit que c'est une propriété *projectivement invariante*. Ainsi, si un point (ou une ligne) se trouve sur un plan et qu'il est projeté, le point sera toujours sur le plan après la projection. On dit que le point ou la ligne est *incident* au plan. Inversement, on dit que le plan est *incident* à la ligne ou au point [Han93] s'il passe par le point ou la ligne.

On peut représenter cette notion d'incidence sous forme matricielle. Ainsi,

$$P = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \in \Pi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T \text{ lorsque } \Pi \cdot P = 0,$$

ce que l'on peut exprimer également comme

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0.$$

En géométrie projective, un point en 2D est incident à une ligne en 3D et une ligne en 2D est incidente à un plan en 3D. Si on a une ligne 2D l d'équation $\begin{bmatrix} a & b & c \end{bmatrix}^T$, $\begin{bmatrix} a & b & 0 & c \end{bmatrix}^T$ représente un plan parallèle à l'axe OZ [Vyg90] (et donc perpendiculaire au plan image) et passant par la ligne. De même, $\begin{bmatrix} a & b & \star & c \end{bmatrix}^T$ représente tous les plans passant par la droite l. l est l'*axe* du *faisceau* de plans d'équation $\begin{bmatrix} a & b & \star & c \end{bmatrix}^T$ où $\star$ peut prendre une valeur arbitraire[1].

Ainsi, par une ligne 3D L et sa projection l passe un seul plan Π, qui appartient au faisceau dont l est l'axe. Voir figure 3.1.

3.1.2 Axiomes de géométrie projective

AXIOME 1 *Trois points non deux à deux confondus déterminent uniquement un plan en 3D.*

[1] $\begin{bmatrix} a & b & \star & d \end{bmatrix}^T$ représente n'importe quel plan d'équation $ax + by + \star z + d = 0$ passant à travers la ligne d'équation $ax + by + d = 0$ du plan $z = 0$.

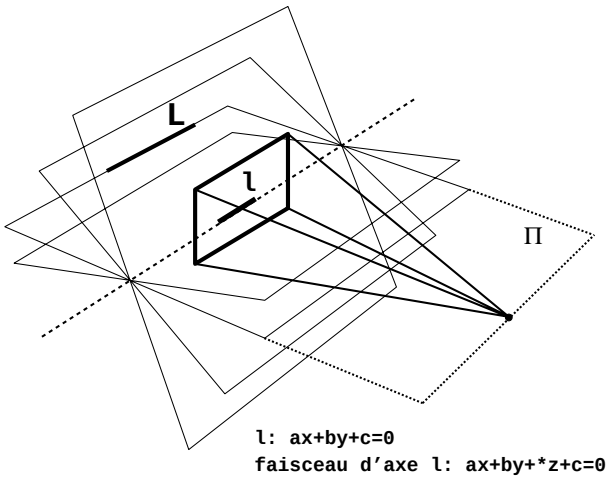


FIG. 3.1: Faisceau de plans dont l'axe appartient au plan de projection.

AXIOME 2 *Trois plans non deux à deux parallèles déterminent uniquement un point en 3D (dual de l'axiome 1).*

AXIOME 3 *Deux points non confondus déterminent uniquement une ligne en 2D ou en 3D.*

AXIOME 4 *Deux lignes non parallèles déterminent uniquement un point en 2D.*

AXIOME 5 *Deux plans non parallèles déterminent uniquement une ligne en 3D.*

3.2 Caméra

Afin d'être le plus général possible dans notre reconstruction, nous voulons être en mesure de pouvoir utiliser des images provenant de sources variées, que ce soit des appareils photographiques réels ou virtuels, des images provenant de l'utilisation d'un modeleur, des images anciennes ou actuelles, etc. La seule contrainte qu'il nous faut respecter est que les objets que nous cherchons à modéliser ne doivent pas être modifiés d'une image à l'autre. Sauf exception (par exemple en cas d'occlusion), les objets qui ne nous intéressent pas n'ont la plupart du temps aucune influence sur notre reconstruction géométrique. Nous désirons donc une représentation de la caméra qui soit la moins contraignante possible.

3.2.1 Dérivation de la matrice de transformation approximant la caméra

Nous allons dériver une matrice de transformation qui permettrait d'approximer le processus qui a permis d'obtenir l'image (i.e. la caméra). Tout d'abord, rappelons qu'une transformation linéaire

générale est représentée par une matrice 4×4 $\mathbf{F}$ de la forme

$$\begin{bmatrix} F_0 & F_1 & F_2 & F_3 \\ F_4 & F_5 & F_6 & F_7 \\ F_8 & F_9 & F_{10} & F_{11} \\ F_{12} & F_{13} & F_{14} & F_{15} \end{bmatrix}.$$

Cette matrice permet de transformer un point 3D en coordonnées homogènes en un autre point par

$$\begin{bmatrix} x' \\ y' \\ z' \\ h' \end{bmatrix} = \mathbf{F} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

Cette matrice peut également représenter la série de transformations de *normalisation* qui permettent de projeter un modèle sur un plan de projection quelconque (*transformation de visualisation* qui place le modèle sous forme canonique avant de le projeter avec une projection traditionnelle). Elle effectue dans ce cas un changement de coordonnées du système global XYZ (le monde) au système de coordonnées UVW du plan de projection [FvDFH90].

Une fois le point transformé, on peut le projeter sur un plan de projection se trouvant à $w = 0$ en utilisant la matrice 4×4 [FvDFH90]

$$\mathbf{M_{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/e & 1 \end{bmatrix}.$$

Le centre de projection est à $w = -e$. Le terme $1/e$ est égal à 0 dans le cas d'une projection orthographique. En effet, $\frac{1}{e}$ tend vers 0 lorsque e tend vers ∞ (centre de projection à l'infini, donc rayons de projection parallèles).

En composant les deux matrices, on obtient la matrice

$$\mathbf{T} = \mathbf{M_{per}F} = \begin{bmatrix} T_0 & T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 & T_7 \\ 0 & 0 & 0 & 0 \\ T_8 & T_9 & T_{10} & T_{11} \end{bmatrix}$$

que l'on va utiliser comme matrice de transformation approximant une caméra. T_{11} est souvent appelé le facteur d'échelle de $\mathbf{T}$. On peut donc normaliser $\mathbf{T}$ en divisant tous les termes par T_{11}.

3.2.2 Utilisation de la matrice

Projection d'un point

Si on *pré-multiplie* un point 3D par $\mathbf{T}$ normalisée, elle agit comme une matrice de projection. Un point 3D homogène $P = \begin{bmatrix} P_x & P_y & P_z & 1 \end{bmatrix}^T$ est alors transformé en

$$\mathbf{T}P = \begin{bmatrix} P_xT_0 + P_yT_1 + P_zT_2 + T_3 \\ P_xT_4 + P_yT_5 + P_zT_6 + T_7 \\ 0 \\ P_xT_8 + P_yT_9 + P_zT_{10} + 1 \end{bmatrix} = \begin{bmatrix} p_u \\ p_v \\ 0 \\ h \end{bmatrix} = p. \tag{3.1}$$

p est donc la projection de P sur le plan $w = 0$, i.e. sur le plan de projection sur lequel se forme l'image. Les coordonnées de p sont exprimées dans le système de coordonnées de l'espace transformé.

Obtention du plan passant par une ligne 3D et sa projection

La méthode de reconstruction que *Rekon* adopte est basée sur l'utilisation de plans 3D passant à travers des lignes en 3D et leurs projections sur les images. Ces plans sont dérivés à l'aide des matrices de transformation $\mathbf{T}$ des images, de la manière décrite dans cette section. Les intersections des différents plans calculés permettent d'obtenir les positions 3D des primitives disponibles dans le système, comme cela sera vu à la section 3.4.

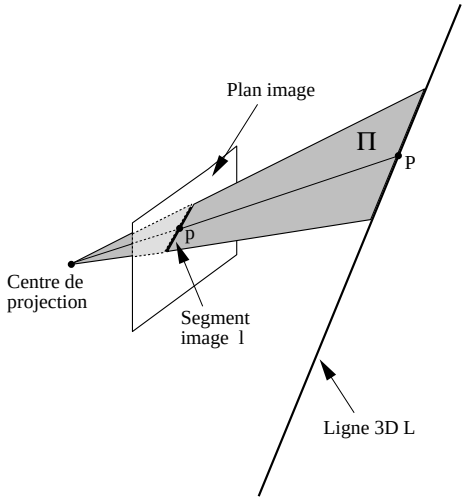


FIG. 3.2: Le plan Π passant par une ligne 3D L et sa projection l sur une image.

Soit une ligne L dans l'espace 3D qui se projette en une ligne l sur l'image (ou plan de projection). Soit Π le plan passant par l et L. Soit P un point sur L et p sa projection sur l. On a donc, tel que vu dans la section précédente, $p = \mathbf{T}P$ (Voir la figure 3.2 pour mieux visualiser ces divers éléments).

$P \in \Pi$, donc on a $P \cdot \Pi = 0$. Par propriété du produit vectoriel[2], cela donne également

$$\Pi^T P = 0. \tag{3.2}$$

Si $p \in l$, p appartient aussi au faisceau[3] de plans $f = \begin{bmatrix} a & b & \star & c \end{bmatrix}^T$ passant par l et on a donc $p \cdot f = 0$. Ainsi,

$$f^T p = 0.$$

Etant donné que $p = \mathbf{T}P$, on a

$$f^T \mathbf{T} P = 0. \tag{3.3}$$

Ainsi, 3.2 et 3.3 permettent d'écrire que

$$f^T \mathbf{T} P = \Pi^T P$$

et donc

$$f^T \mathbf{T} = \Pi^T.$$

Ainsi, on obtient le plan Π passant par l et L avec

$$\begin{bmatrix} a & b & \star & c \end{bmatrix} \begin{bmatrix} T_0 & T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 & T_7 \\ 0 & 0 & 0 & 0 \\ T_8 & T_9 & T_{10} & 1 \end{bmatrix} = \begin{bmatrix} aT_0 + bT_4 + cT_8 \\ aT_1 + bT_5 + cT_9 \\ aT_2 + bT_6 + cT_{10} \\ aT_3 + bT_7 + c \end{bmatrix} = \Pi^T.$$

3.2.3 Calibration d'une image

Nous pouvons modéliser notre caméra par cette matrice de transformation normalisée dont les effets correspondent à nos besoins. Dans notre contexte, la *calibration* d'une image consiste donc à retrouver les onze paramètres de $\mathbf{T}$ pour cette image

$$\mathbf{T} = \begin{bmatrix} T_0 & T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 & T_7 \\ 0 & 0 & 0 & 0 \\ T_8 & T_9 & T_{10} & 1 \end{bmatrix}.$$

Cette méthode ne permet pas d'obtenir une calibration complète de la caméra, c'est-à-dire la détermination de tous ses paramètres. Nous ne cherchons qu'à récupérer la transformation (ou une approximation de la transformation) qui fait passer du monde tridimensionnel au monde bidimensionnel, et ce à l'aide d'un nombre relativement faible de données 3D mesurées ou évaluées de façon

[2] Pour deux vecteurs $\mathbf{u}$ et $\mathbf{v}$, on a $\mathbf{u} \cdot \mathbf{v} = \mathbf{v}^T \mathbf{u}$.

[3] Nous avons vu à la section 3.1 que par la ligne 2D d'équation $\begin{bmatrix} a & b & c \end{bmatrix}^T$ sur le plan image passe une infinité de plans d'équation $\begin{bmatrix} a & b & \star & c \end{bmatrix}^T$.

simple.

Dans notre système, cette calibration s'effectue par résolution de systèmes d'équations linéaires comme nous le verrons dans la section suivante. Hartley [Har97] a démontré que l'utilisation de systèmes linéaires pour la calibration de caméras (comme celui de Longuet-Higgins [LH81]) peut être aussi efficace et robuste que celle, normalement adoptée, de systèmes non-linéaires. Cette équivalence est conditionnée par le suivi de certaines règles numériques simples pour la résolution des systèmes, par exemple la normalisation (translation et mise à l'échelle) des coordonnées des primitives utilisées dans la calibration. Il décrit les techniques possibles pour effectuer cette normalisation des données, techniques qui n'ajoutent presque rien à la complexité de l'algorithme de résolution des systèmes. Nous verrons un peu plus loin les avantages de n'utiliser que des systèmes d'équations linéaires par rapport aux non-linéaires.

Nous utilisons cette matrice pour reconstruire la scène 3D ainsi que pour la reprojeter sur les images si nécessaire. Nous n'avons donc pas besoin du déplacement entre deux positions de caméra(s), ni de leurs orientations. Il est donc inutile de chercher à extraire de la matrice tout paramètre réel de la caméra, extrinsèque ou intrinsèque. De fait, cette représentation est suffisante pour nos besoins. De plus, une telle extraction de paramètres (décomposition) suppose que l'on utilise un certain modèle de caméra et que notre matrice possède la structure propre à ce type de caméra. Etant donné que nous voulons rester le plus général possible, nous ne faisons aucune hypothèse quant au modèle de caméra utilisé et évitons donc les erreurs dues à la non conformité des coefficients de la matrice retrouvée au modèle supposé. En effet, en assumant un certain modèle de caméra, on doit respecter des contraintes additionnelles imposées par ce modèle, par exemple le fait que le plan image soit perpendiculaire à l'axe de projection ou que la lentille soit idéale et n'introduise aucune distorsion. La matrice que l'on récupère ne fait qu'approximer la transformation due à la caméra utilisée. Par contre, on "perd" par le fait même les contraintes inhérentes au type de projection ou à la caméra que l'on utilise et qui auraient pu nous aider dans la reconstruction (par exemple, la contrainte épipolaire résultant de l'utilisation d'une caméra de type sténopé ("pin-hole") ou le fait que la matrice de rotation entre deux positions de caméra soit orthonormale). De plus, nous assumons la linéarité du processus de projection. Cette hypothèse, bien qu'elle rende notre solution robuste et efficace de par sa méthode de détermination, est le plus souvent fausse étant donné la non linéarité de la plupart des processus d'imagerie. Nous verrons que, malgré ces désavantages, nous obtenons des résultats équivalents aux méthodes utilisant des techniques non-linéaires plus complexes.

3.3 Principes de bases de *Rekon*

On peut écrire la transformation d'un point 3D P comme

$$p = \mathbf{T}P \Rightarrow \begin{bmatrix} p'_u & p'_v & 0 & h \end{bmatrix}^T = \mathbf{T} \begin{bmatrix} P_x & P_y & P_z & 1 \end{bmatrix}^T.$$

En se référant à l'équation 3.1, on obtient le système

$$\begin{aligned} T_0 P_x + T_1 P_y + T_2 P_z + T_3 &= h p_u \\ T_4 P_x + T_5 P_y + T_6 P_z + T_7 &= h p_v \\ T_8 P_x + T_9 P_y + T_{10} P_z + 1 &= h \end{aligned}$$

où $p_u = \frac{p'_u}{h}$ et $p_v = \frac{p'_v}{h}$. $\begin{bmatrix} p_u & p_v & 0 & 1 \end{bmatrix}^T$ représente le point 2D p après normalisation.

Par substitution de h dans les deux premières équations, on obtient

$$(T_0 - p_u T_8)P_x + (T_1 - p_u T_9)P_y + (T_2 - p_u T_{10})P_z + T_3 - p_u = 0 \tag{3.4}$$

et

$$(T_4 - p_v T_8)P_x + (T_5 - p_v T_9)P_y + (T_6 - p_v T_{10})P_z + T_7 - p_v = 0. \tag{3.5}$$

Nous pouvons utiliser ces équations de trois manières différentes.

1. Si $\mathbf{T}$ et P sont connues, nous pouvons obtenir les coordonnées de p, la projection de P par $\mathbf{T}$ sur le plan de projection associé à $\mathbf{T}$.
2. Si nous connaissons les matrices de transformation $\mathbf{T_1}$ et $\mathbf{T_2}$ de deux images ainsi que les coordonnées p_1 et p_2 de la projection d'un même point P sur ces deux images, nous obtenons quatre équations pour les trois inconnues qui composent la position du point P en 3D. Nous pouvons donc calculer cette position.
3. La matrice de transformation $\mathbf{T}$ d'une image contient onze paramètres. Un point 2D et sa position 3D nous procurent deux équations. Etant donné qu'il nous faut au moins onze équations pour déterminer $\mathbf{T}$, nous avons besoin de six points 3D et de leurs projections dans l'image pour calculer $\mathbf{T}$. C'est la méthode de calibration que nous emploierons.

Ces trois aspects fondent les bases de plusieurs systèmes de reconstruction [Car95] et également du nôtre.

3.4 Primitives géométriques disponibles

Les modèles reconstruits par notre système sont constitués de points, lignes et polygones. Chacune de ces primitives peut nous aider à calculer les paramètres de la matrice de transformation d'une image et, une fois celle-ci déterminée pour certaines images, on peut calculer la position 3D des primitives sur ces images.

3.4.1 Point

Le point est la primitive de base pour la plupart des systèmes de reconstruction. Il est très facile d'en trouver plusieurs dans les images, que ce soit des coins ou des points dans une texture. La façon de calculer la position 3D d'un point à partir de ses projections a déjà été étudiée dans la section 3.3 ; nous allons montrer ici que la même méthode de calcul peut être vue sous un autre angle.

Soit Π_{proj} un plan de projection quelconque et $P = \begin{bmatrix} P_x & P_y & P_z & 1 \end{bmatrix}^T$ un point dans l'espace. P se projette sur Π_{proj} en un point $p = \begin{bmatrix} p_u & p_v & p_h \end{bmatrix}^T$. P et p définissent une ligne 3D L, sauf si P et p sont confondus, auquel cas P se trouve sur le plan de projection (voir figure 3.3).

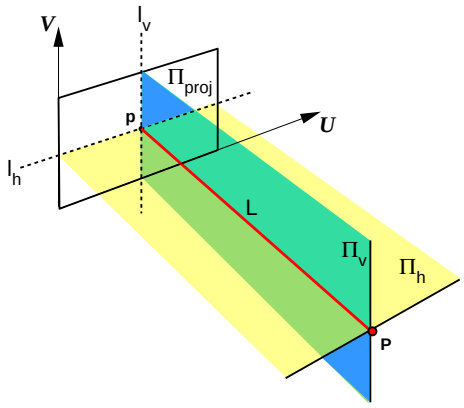


FIG. 3.3: Contrainte de position d'un point.

Il existe une infinité de plans 3D qui passent à travers cette ligne, i.e. la ligne est l'axe d'un *faisceau* de plans. Soit un plan Π de ce faisceau (évidemment non parallèle au plan de projection Π_{proj}). Son intersection avec Π_{proj} donne une ligne. Choisissons deux tels plans orthogonaux entre eux Π_h et Π_v tels que les lignes résultant de leur intersection avec Π_{proj} soient respectivement parallèles aux axes de coordonnées horizontal (U) et vertical (V) et s'intersectent en p (voir figure 3.3).

Π_h coupe donc Π_{proj} en l_h, la ligne horizontale d'équation $v = p_v$ que l'on peut exprimer sous la forme $\begin{bmatrix} 0 & 1 & -p_v \end{bmatrix}^T$.

Tel que vu dans la section précédente, on obtient l'équation de Π_h à l'aide de $\mathbf{T}$

$$\Pi_h = \begin{bmatrix} 0 & 1 & \star & -p_v \end{bmatrix} \mathbf{T} = \begin{bmatrix} T_4 - p_v T_8 \\ T_5 - p_v T_9 \\ T_6 - p_v T_{10} \\ T_7 - p_v \end{bmatrix}^T .$$

$$P \in \Pi_h \quad \text{si} \quad \Pi_h \cdot P = 0.$$

On obtient donc une première contrainte à la position de $P = \begin{bmatrix} P_x & P_y & P_z & 1 \end{bmatrix}^T$,

$$(T_4 - p_v T_8)P_x + (T_5 - p_v T_9)P_y + (T_6 - p_v T_{10})P_z = p_v - T_7, \tag{3.6}$$

qui correspond à l'équation 3.5.

De même, Π_v coupe Π_{proj} en l_v, la ligne verticale d'équation $x = p_u$ que l'on peut exprimer sous la forme $\begin{bmatrix} 1 & 0 & -p_u \end{bmatrix}^T$.

Le plan Π_v s'obtient donc avec

$$\Pi_v = \begin{bmatrix} 0 & 1 & \star & -p_u \end{bmatrix} \mathbf{T} = \begin{bmatrix} T_0 - p_u T_8 \\ T_1 - p_u T_9 \\ T_2 - p_u T_{10} \\ T_3 - p_u \end{bmatrix}^T .$$

$$P \in \Pi_v \quad \text{si} \quad \Pi_v \cdot P = 0.$$

Ainsi,

$$(T_0 - p_u T_8)P_x + (T_1 - p_u T_9)P_y + (T_2 - p_u T_{10})P_z = p_u - T_3 \tag{3.7}$$

correspond à l'équation 3.4.

Tel que décrit dans la section 3.3, ces deux contraintes peuvent nous permettre de déterminer $\mathbf{T}$ ainsi que la position 3D du point. Nous appellons ce système de deux équations reliant un point 2D à sa position 3D une *contrainte de position*.

Il nous faut trois plans pour déterminer uniquement un point en 3D. Nous obtenons deux plans pour chaque image sur laquelle ce point est identifié. Il faut donc au moins deux images pour calculer la position 3D d'un point. Ces deux images nous permettent de calculer quatre plans et nous obtenons ainsi un système surdéterminé d'équations que l'on peut résoudre dans le sens des moindres carrés avec *SVD*. Puisque l'on dispose de plus de trois plans, on trouve le "meilleur" point correspondant à l'intersection de ces plans (il est très improbable qu'ils s'intersectent exactement au même point), une sorte d'intersection *moyenne*.

3.4.2 Polygone

Reconstruire un modèle sous forme d'un nuage de points ne nous est pas très utile en synthèse d'images. La structure 3D sera difficile à percevoir et il nous sera impossible d'utiliser ce modèle pour y appliquer des textures, en calculer l'illumination, etc. Il est donc nécessaire de disposer d'une des primitives les plus utilisées en infographie : le polygone. En effet, la plupart des modèles en trois dimensions utilisés en infographie sont composés de polygones. Le polygone est également la primitive géométrique la plus simple pour faire de la photométrie. Etant donné que la suite du projet vise à effectuer une reconstruction photométrique en plus d'une récupération de textures, il est important de pouvoir reconstruire des polygones plutôt que seulement des points.

En vision, il est plus facile d'extraire des primitives de bas niveau tels que des points ou des segments mais une étape supplémentaire, plus complexe, est nécessaire pour les réunir sous forme de facettes 3D et d'objets. Il existe également des méthodes [Cha93] (*edge-region cooperative segmentation*) qui essaient de former des chaînes de segments en 2D susceptibles de constituer la projection d'un polygone 3D de la scène. Une fois ces facettes 2D reconstituées, elles sont mises en correspondance de façon automatique. Les polygones 3D reconstruits sont ensuite soumis à des tests de planarité pour vérifier si les hypothèses relatives aux chaînes de segments 2D étaient justes. On peut voir immédiatement nombre de problèmes qui peuvent surgir tout au long de ce processus de reconstruction et qui peuvent forcer à tout recommencer plusieurs fois. Dans notre système, nous n'avons à faire face à aucun de ces problèmes puisque nous traçons nous-mêmes nos polygones sur les images et les mettons en correspondance explicitement.

Nous avons vu qu'un point dans une image génère deux équations de contraintes pour sa position 3D. Un polygone génère donc autant de contraintes qu'il possède de points. Etant donné que la transformation projective ne modifie pas l'incidence, il est possible de retrouver la structure du polygone en 3D en reliant les points 3D correspondant aux sommets du polygone sur l'image. Par contre, cela ne garantit pas la planarité des polygones de plus de trois sommets ainsi reconstruits.

3.4.3 Ligne

Dans beaucoup de scènes, il est fréquent que des faces d'objets ne soient pas entièrement visibles. De fait, il n'est pas possible de les recouvrir par un seul polygone. Il est donc utile de pouvoir placer et reconstruire des lignes que nous pourrons ensuite relier entre elles d'une image à l'autre pour former les faces. Par exemple, lors de la reconstruction d'une pièce dans un bâtiment, il est très ardu, avec une lentille normale, de pouvoir obtenir un mur complet sur une image. Il est plus facile de récupérer les

côtés du mur, sous forme de droites, à partir de plusieurs vues puis de les réunir en un seul polygone. De même, il existe nombre de segments dans une image qui pourraient être utilisés dans diverses contraintes. Il est ainsi intéressant d'ajouter une telle primitive. Les lignes de *Rekon* sont soit des droites, soit des demi-droites, soit des segments.

Une ligne peut se représenter de plusieurs façons. Par exemple, un point associé à un vecteur directeur ou un système de deux équations de plans dont l'intersection constitue la ligne en 3D. Ces représentations ne sont pas vraiment intégrables dans notre système car elles dépendent du point ou des plans choisis et ne sont donc pas homogènes. Nous avons donc choisi d'utiliser la *représentation de Plücker* d'une ligne,

$$\begin{bmatrix} 0 & P_{xy} & P_{xz} & P_{xh} \\ -P_{xy} & 0 & P_{yz} & P_{yh} \\ -P_{xz} & -P_{yz} & 0 & P_{zh} \\ -P_{xh} & -P_{yh} & -P_{zh} & 0 \end{bmatrix},$$

qui peut être intégrée directement dans nos systèmes d'équations linéaires. Le lecteur moins familier avec cette représentation peut se référer à l'annexe A et aux ouvrages qui y sont mentionnés.

Voyons comment utiliser cette représentation afin de calibrer une caméra et calculer la position d'une ligne en trois dimensions à partir de ses projections dans les images.

Soit une ligne 2D $l = \begin{bmatrix} a & b & c \end{bmatrix}^T$ passant par les points $p_1(x_1, y_1)$ et $p_2(x_2, y_2)$, et p un point quelconque de cette ligne. l est représentée par l'équation

$$déterminant(\vec{p_1p}, \vec{p_1p_2}) = 0 \quad \Rightarrow \quad \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x - x_1 & y - y_1 \end{vmatrix} = 0$$

et ainsi,

$$a = y_1 - y_2 \qquad b = x_2 - x_1 \qquad c = x_1y_2 - x_2y_1.$$

La ligne 3D L correspondant à cette projection l se trouve sur le plan Π (voir section 3.2) défini par

$$\Pi = \begin{bmatrix} a & b & \star & c \end{bmatrix} \mathbf{T} = \begin{bmatrix} aT_0 + bT_4 + cT_8 \\ aT_1 + bT_5 + cT_9 \\ aT_2 + bT_6 + cT_{10} \\ aT_3 + bT_7 + c \end{bmatrix}^T.$$

$L \in \Pi$ si, pour deux points[4] $P_1(P_{1x}, P_{1y}, P_{1z})$ et $P_2(P_{2x}, P_{2y}, P_{2z})$, $P_1 \neq P_2$, nous avons

$$\Pi \cdot P_1 = 0 \qquad \text{et} \qquad \Pi \cdot P_2 = 0.$$

[4] P_1 et P_2 ne sont pas les correspondants 3D de p_1 et p_2 mais n'importe quels points sur la ligne 3D.

Autrement dit, $L \in \Pi$ si

$$(aT_0 + bT_4 + cT_8)P_{1x} + (aT_1 + bT_5 + cT_9)P_{1y} + (aT_2 + bT_6 + cT_{10})P_{1z} = -(aT_3 + bT_7 + c), \tag{3.8}$$

et de même pour P_2,

$$(aT_0 + bT_4 + cT_8)P_{2x} + (aT_1 + bT_5 + cT_9)P_{2y} + (aT_2 + bT_6 + cT_{10})P_{2z} = -(aT_3 + bT_7 + c). \tag{3.9}$$

Ce qui nous donne deux équations par couple (ligne 3D ; projection de cette ligne dans l'image). Si nous possédons la position de la ligne en 3D sous forme d'une matrice de Plücker, nous pouvons en extraire deux points P_1 et P_2 tel qu'indiqué dans la section A.2. L'extraction des points p_1 et p_2 de la ligne 2D l, projection de L, nous permet de calculer les coefficients a, b et c. Il nous reste donc les onze paramètres de la matrice de transformation $\mathbf{T}$ comme inconnues. Si l'usager indique six lignes 3D et leurs projections sur une image, il est donc possible de calibrer cette image à l'aide des équations 3.8 et 3.9, car on obtient ainsi douze équations. Ces contraintes peuvent également être utilisées avec celles des points pour plus de flexibilité dans le calcul des caméras. Dans le système *Rekon* actuel, nous n'utilisons que les points pour la calibration des images. Les résultats obtenus sont pour l'instant satisfaisants et l'intégration des lignes dans la calibration se fera éventuellement si nécessaire.

Liu *et al.* [LHF90] présentent une manière de calculer une matrice de projection à partir de lignes exprimées en coordonnées de Plücker. Nous avons jugé qu'il n'était pas nécessaire d'implanter cette méthode (qui semble assez compliquée et comporte beaucoup de contraintes additionnelles) puisque les matrices de projection obtenues à l'aide des points seulement sont satisfaisantes. De plus, on voit souvent assez de points dans une image pour pouvoir facilement définir la matrice de projection avec une précision satisfaisante.

Afin de déterminer la position 3D d'une ligne, nous utilisons le fait qu'une ligne se trouve à l'intersection de deux plans (axiome 5) tel que montré sur la figure figure 3.4. Nous n'aurons donc besoin que de deux images.

Une ligne 3D L appartient à un plan $\Pi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$ si $L\Pi = [0]$, i.e. si

$$\begin{bmatrix} 0 & P_{xy} & P_{xz} & P_{xh} \\ -P_{xy} & 0 & P_{yz} & P_{yh} \\ -P_{xz} & -P_{yz} & 0 & P_{zh} \\ -P_{xh} & -P_{yh} & -P_{zh} & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

En isolant les coordonnées de Plücker, on obtient le système d'équations (en supposant la ligne nor-

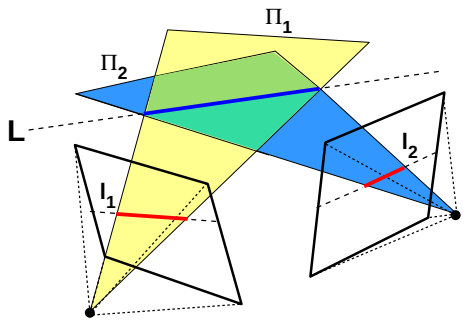


FIG. 3.4: Contrainte de correspondance pour une ligne.

malisée, i.e. les deux points qui la définissent sont normalisés et $h = 1$)

$$\begin{bmatrix} b & c & 0 & d & 0 & 0 \\ -a & 0 & c & 0 & d & 0 \\ 0 & -a & -b & 0 & 0 & d \\ 0 & 0 & 0 & -a & -b & -c \end{bmatrix} \begin{bmatrix} P_{xy} \\ P_{xz} \\ P_{yz} \\ P_{xh} \\ P_{yh} \\ P_{zh} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Cette contrainte nous fournit quatre équations pour le calcul de L. Il nous suffit comme prévu d'un autre plan Π' pour avoir suffisamment d'équations pour déterminer les six coordonnées Plücker de la ligne.

Si nous avons n projections de cette ligne, nous obtenons n plans 3D auxquels elle appartient et donc $4n$ contraintes pour le calcul des six paramètres de sa position. Il est clair que nous avons à résoudre un système surdéterminé et nous allons donc déterminer la "meilleure" ligne possible au sens des moindres carrés.

Une solution triviale mais indésirable correspond au vecteur nul. Afin d'éviter une telle solution, nous introduisons une contrainte additionnelle afin de forcer l'une des coordonnées de la ligne à varier. Ainsi, on donnera à P_x, P_y ou P_z une valeur arbitraire de 1. Pour être sûr que la coordonnée choisie varie réellement et donc que la ligne n'est pas parallèle à un axe ou plan de coordonnée, nous approximons son vecteur directeur à l'aide de deux des plans qui la définissent. En effet, le produit vectoriel de deux plans donne la direction de leur intersection (figure 3.5). Nous choisissons pour notre contrainte la coordonnée de ce vecteur direction qui a la plus grande valeur absolue.

3.5 Fonctionnement

Le système *Rekon* est un système interactif de reconstruction de scènes tridimensionnelles à partir d'images. Il allie les informations que lui donne un usager au sujet des images à un moteur de conver-

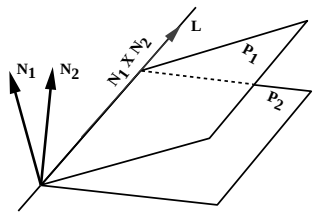


FIG. 3.5: Les normales de deux plans permettent de calculer la direction de leur ligne d'intersection.

gence qui utilise ces informations pour calibrer les images et trouver le "meilleur" modèle 3D qui satisfasse les contraintes spécifiées par l'usager. Un schéma de sa structure et de son fonctionnement est présenté à la figure 3.6.

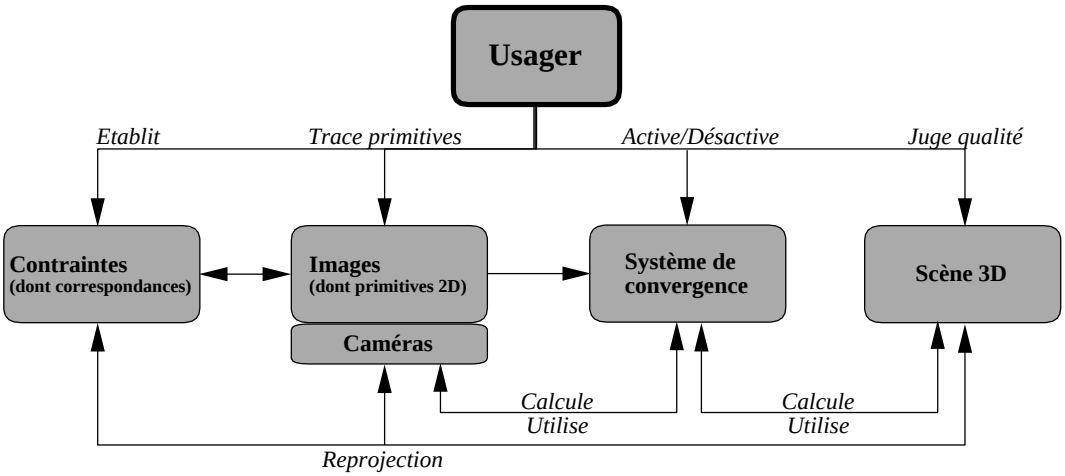


FIG. 3.6: Structure du système *Rekon.*

3.5.1 Structure du système

Rekon travaille principalement avec trois listes :

- Une liste d'images choisies par l'usager. A chaque image est associée une caméra, représentée par une matrice de transformation tel que vu précédemment, ainsi qu'une liste de primitives 2D dessinées sur l'image par l'usager.
- Une liste de contraintes géométriques entre les primitives 2D et/ou 3D.
- Une liste de primitives 3D reconstruites par le système, qui constitue la scène 3D.

Le modèle 3D est représenté à l'aide de points, lignes et polygones en trois dimensions. A chaque primitive 2D est associée une primitive 3D. Plusieurs primitives 2D peuvent être associées à la même primitive 3D mais pas l'inverse. Deux primitives 2D définies comme ayant des points communs sur l'image (l'usager peut tracer une nouvelle primitive en utilisant les points d'une autre) permet d'indi-

quer que ces primitives sont reliées dans l'espace 3D. C'est ainsi que l'on obtient la connectivité de nos primitives 3D.

3.5.2 Rôle de l'usager

L'intégration de l'usager dans le processus de reconstruction est ce qui distingue principalement notre système des systèmes de reconstruction de la vision qui tentent de tout faire automatiquement, la segmentation, la mise en correspondance, la calibration, la reconstruction (de géométrie et de textures), etc. Nous évitons ainsi tous les problèmes encore présents dans ces techniques, le compromis étant une perte d'automatisme pour une meilleure qualité de reconstruction. Il ne faut pas non plus perdre de vue que les modèles que nous cherchons à obtenir sont destinés à des applications infographiques et donc se doivent d'être plus exacts et visuellement "meilleurs" que des modèles utilisés par les applications de la vision par ordinateur.

L'usager commence par choisir les images avec lesquelles il désire travailler. Il pourra en ajouter ou en enlever en tout temps au cours de la reconstruction. Nous n'avons pas besoin d'effectuer une segmentation de nos images puisque l'usager trace lui-même les contours des primitives qu'il désire reconstruire, soit des points, des lignes ou des polygones.

Afin d'amorcer le système, il doit spécifier la position 3D relative (ou absolue si elle est connue) de primitives tracées (par exemple, six points). Ceci permet de calculer une première caméra tel que décrit précédemment. L'usager pourra ensuite indiquer des correspondances entre ces primitives et leurs équivalents sur d'autres images afin de calculer d'autres matrices de transformation (caméras). Une fois les caméras de deux images disponibles, il est possible de calculer la position des primitives de ces images mises en correspondance. Des itérations entre calibration et reconstruction permettent de faire *converger* le système, comme cela sera décrit plus loin.

L'usager peut également spécifier des contraintes entre les différentes primitives. Ces contraintes servent à raffiner le modèle obtenu et seront décrites plus en détails au chapitre 4. Il peut choisir de n'appliquer que certaines de ces contraintes lors de la reconstruction.

En tout temps, l'usager a la possibilité de demander un calcul des caméras ou un calcul des positions des primitives. Il peut ainsi faire converger la reconstruction jusqu'à ce qu'il soit satisfait du modèle obtenu. Pour juger de la qualité de la reconstruction et/ou de la calibration, il peut demander la reprojection du modèle reconstruit sur une image. Pour cela, il faut évidemment que la matrice de transformation associée à l'image soit disponible. Il dispose également du modèle 3D en reconstruction et peut s'en servir pour faciliter l'établissement de correspondances entre primitives 2D ou

pour observer les éventuels défauts et effectuer les corrections qui s'imposent. Le modèle 3D est visible dans une fenêtre qui permet de l'examiner dans n'importe quelle position par des rotations, translations et changements d'échelle.

3.5.3 Rôle du système

Le rôle du système se résume en deux mots : calibration et reconstruction. Un moteur de convergence alterne entre ces deux étapes en essayant d'améliorer les deux aspects, chacun ayant une influence directe sur l'autre. C'est ce qui fait de notre méthode une méthode *itérative*.

Ce sont les contraintes introduites par l'usager qui permettent de solutionner ces deux étapes. Tel que décrit plus tôt, nous exprimons ces contraintes comme un ensemble d'incidences sur des plans 3D. Nous pouvons donc les intégrer directement dans nos systèmes d'équations linéaires.

Calibration

Au niveau de la calibration, c'est à l'usager d'amorcer le système. Il donne manuellement la position relative d'un certain nombre de primitives du système (par exemple, six points ou six lignes). Les coordonnées de ces primitives dans l'espace 3D sont définies dans le repère euclidien de la scène. Ces positions permettent de calculer une première matrice de transformation (caméra). Si ces primitives sont également présentes dans une autre image, on pourra la calibrer (calculer sa caméra) par correspondance avec les primitives de la première image. Si les primitives ne sont pas toutes présentes dans une autre image, l'usager devra fournir de nouvelles positions. Une fois deux caméras calculées, la reconstruction peut commencer et les autres caméras pourront être calculées à l'aide des correspondances établies par l'usager entre les primitives 2D.

Ainsi, nous avons un ensemble de contraintes par image pour calculer sa matrice de projection. Ces contraintes sont celles qui relient les primitives 2D dessinées sur l'image à leur valeur 3D (disponible lorsqu'entrée par l'usager ou calculée par le système). Ce sont uniquement ces contraintes qui permettent de résoudre le système d'équations associé à l'image dont les inconnues sont les onze paramètres de la matrice de transformation utilisée pour obtenir l'image. D'autres contraintes pourraient être utilisées, mais dans les scènes reconstruites, nous n'en avons pas ressenti le besoin.

Reconstruction

Au moment de la reconstruction du modèle 3D on utilise toutes les contraintes du système, contrairement à la calibration qui n'utilise que les contraintes définies entre les primitives 2D d'une

image et leurs positions 3D. Les contraintes de position éventuellement alliées à celles décrites au chapitre 4 permettent d'obtenir les positions des primitives dont le nombre de contraintes disponibles est suffisant pour leur calcul. Les contraintes pour le calcul d'une primitive 3D sont exprimées sous la forme d'équations dont les inconnues sont la position de cette primitive. Les coefficients connus du système à résoudre proviennent des équations des plans auxquels appartient la primitive, plans dont le calcul a été décrit dans la section 3.4.

Le système possède un gestionnaire de contraintes qui s'occupe d'organiser, de propager et de gérer les contraintes. Il choisit les contraintes à appliquer lors du calcul d'une primitive. Il faut noter qu'on ne calcule pas toutes les primitives d'un seul coup, mais plutôt primitive par primitive, chaque primitive ayant un ensemble de contraintes s'appliquant au calcul de sa position.

Résolution des systèmes d'équations linéaires

Nous avons choisi d'exprimer nos contraintes par des équations linéaires en raison de leur simplicité et de leur facilité d'expression et de traitement. De plus, il existe plusieurs méthodes de résolution bien connues et robustes. A l'inverse, les méthodes d'optimisation non-linéaires (par exemple, la méthode de Levenberg-Marquardt) sont susceptibles de poser des problèmes dus aux extrema locaux et aux discontinuités dans l'espace de solutions. Elles ont souvent besoin de bonnes valeurs de départ et donc d'une approximation de la solution et des gradients des fonctions objectifs. Par contre, nous devons sacrifier les possibilités de contraintes que peuvent nous offrir l'utilisation d'équations non-linéaires, par exemple les notions de distance ou de planarité exacte, ainsi que les contraintes inhérentes à certains modèles de projection non-linéaires, telles la position du point focal, la distorsion radiale, la perpendicularité du plan image, etc. Les algorithmes linéaires sont également plus sensibles au bruit dans les données que les algorithmes non-linéaires, ce qui ne nous concerne pas vraiment puisque l'usager spécifie lui-même les contraintes. Zhang *et al.* [ZS97] utilisent une méthode linéaire pour obtenir une solution initiale pour sa calibration et sa reconstruction (d'ailleurs similaire à la nôtre) puis adopte une méthode non-linéaire pour raffiner les résultats, ce qui semble être une solution de compromis intéressante.

Nos systèmes d'équations ont la plupart du temps plus d'équations que d'inconnues, on dit qu'ils sont *surdéterminés*. On essaie donc de trouver la meilleure solution de "compromis", i.e. celle qui satisfait le mieux toutes les équations simultanément. Si cette notion de "satisfaction" est définie dans le sens des moindres carrés[5], alors le problème linéaire surdéterminé se réduit à un problème linéaire

[5]On veut minimiser la somme des différences élevées au carré entre les parties gauche et droite de l'équation $\mathbf{A}\mathbf{x} = \mathbf{b}$,

généralement solutionnable, que l'on appelle le problème linéaire des moindres carrés [PFTV92] : Etant donné que nous savons que nos équations ne seront pas satisfaites exactement dû aux erreurs de précision, nous allons essayer de minimiser la somme des erreurs au carré.

Il existe plusieurs méthodes de résolution possibles [PFTV92]. Nous avons choisi d'utiliser la *décomposition en valeurs singulières* (ou *SVD* pour *singular value decomposition*) qui est une méthode de choix pour résoudre la plupart des problèmes linéaires de moindres carrés. En effet, certaines méthodes de résolution de systèmes d'équations linéaires peuvent ne procurer aucune solution. Un tel état bloquerait ou dégraderait notre système et nous voulons donc éviter de nous retrouver dans une telle impasse. Par contre, *SVD* dans le cas d'un système surdéterminé du type $\mathbf{A}\mathbf{x} = \mathbf{b}$ donne *toujours* une solution qui sera la meilleure approximation à la solution réelle dans le sens des moindres carrés. *SVD* utilise un ensemble puissant de techniques. C'est un algorithme stable et efficace dont les défauts de comportement sont extrêmement rares et qui converge rapidement vers la solution. Nous l'utilisons dans notre système comme une "boîte noire" à laquelle on fournit la matrice $\mathbf{A}$ et le vecteur $\mathbf{b}$ et qui nous retourne le vecteur solution $\mathbf{x}$.

Nous utilisons donc *SVD* pour solutionner le problème de la récupération des paramètres de nos caméras et celui de la position de nos primitives 3D. On ne peut vraiment prouver la convergence vers un meilleur modèle puisque rien ne nous garantit que le modèle calculé sera plus "proche" du modèle réel. En effet, nous ne travaillons pas avec les positions 2D exactes des primitives mais plutôt avec des approximations données par l'usager par ses dessins. Par contre, dans toutes les situations où nous avons testé notre système, la solution a toujours convergé vers un état plus stable du système et une meilleure reconstruction du modèle.

Cette convergence est due à notre façon de calibrer et de reconstruire. On résout d'abord pour les paramètres des matrices de transformation qui correspondent le mieux aux projections indiquées (primitives 2D tracées sur les images) des primitives 3D dont la position est disponible. Une fois ces matrices calculées, on peut calculer les positions des primitives du système à l'aide des correspondances indiquées par l'usager. Ces positions permettent éventuellement de calculer de nouvelles matrices de projection qui, à leur tour, permettent de déterminer de nouvelles positions pour les primitives du système. Et ainsi de suite jusqu'à ce qu'aucune information additionnelle ne soit calculée lors d'une des deux étapes. A partir de ce moment, les itérations de résolution effectuées ne servent qu'à "améliorer" le modèle et les caméras récupérées. L'usager décidera à quel moment le processus devra être arrêté. Le système pourrait également le faire automatiquement dès qu'il constate que la position

---

soit $[\mathbf{A}\mathbf{x} - \mathbf{b}]^2$.

globale des points en 3D ne change pas de manière significative par rapport à l'itération précédente, par exemple, et que l'usager ne devrait donc percevoir aucun changement dans le modèle.

En résumé, lorsque l'on calcule les paramètres de la matrice de projection, on le fait à partir des "meilleures" positions des primitives calculées à l'étape précédente. Ces "meilleures" matrices devraient pouvoir permettre de re-calculer de "meilleures" positions pour les primitives 3D, et ainsi de suite. Nous plaçons le terme "meilleur" entre guillemets car nous ne prouvons pas que ce seront nécessairement des positions plus précises mais, dans tous les cas observés, c'est effectivement ce qui se produit.

Lors de notre reconstruction, nous allons atteindre un point où le système se stabilisera et où toute nouvelle itération de convergence n'aura qu'un effet minime sur la calibration et la reconstruction. Si le modèle ne nous satisfait pas et que nous avons effectué toute correspondance utile possible, il faut ajouter de l'information additionnelle par une nouvelle image ou sur le modèle lui-même et non plus seulement sur ses projections. C'est l'objet du prochain chapitre.

3.6 Résultats

Les résultats obtenus par l'application des principes de base décrits dans cette section sont satisfaisants. Par contre, nous avons observé quelques problèmes dans certaines scènes reconstruites. Ces problèmes sont des problèmes courants du domaine et plusieurs systèmes de reconstruction en accusent les effets.

Parmi les problèmes les plus fréquemment rencontrés, on peut citer

- des polygones non planaires ;
- des primitives coplanaires en 3D qui ne le sont plus causant des objets qui semblent "flotter" dans l'espace ;
- des primitives qui ne sont plus parallèles ou perpendiculaires entre elles. Ce genre d'anomalie est très vite remarqué par un observateur. De plus, plus les primitives sont éloignées des primitives qui ont servi à calibrer l'image, plus elles seront "inclinées" ;
- des polygones que l'on ne peut reconstruire car un ou plusieurs de leurs sommets ne sont visibles que sur une seule image, ce qui résulte en des objets incomplets ;
- des modèles qui reprojettent bien sur les images mais dont la qualité visuelle est moins satisfaisante. Ceci nous montre que le fait que les images soient bien calibrées ne conduit pas nécessairement à un modèle correct.

Les quatre premiers problèmes sont présents dans la reconstruction d'une scène que l'on peut observer à la figure 3.7 tandis que la figure 3.8 illustre le dernier.

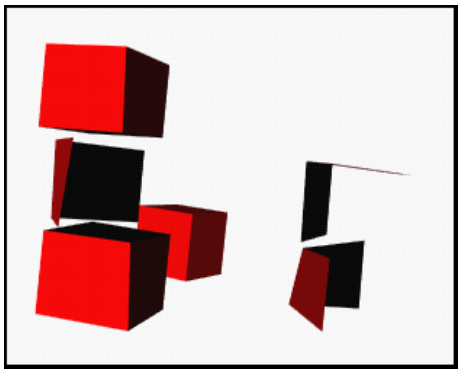


FIG. 3.7: Quelques problèmes de reconstruction.

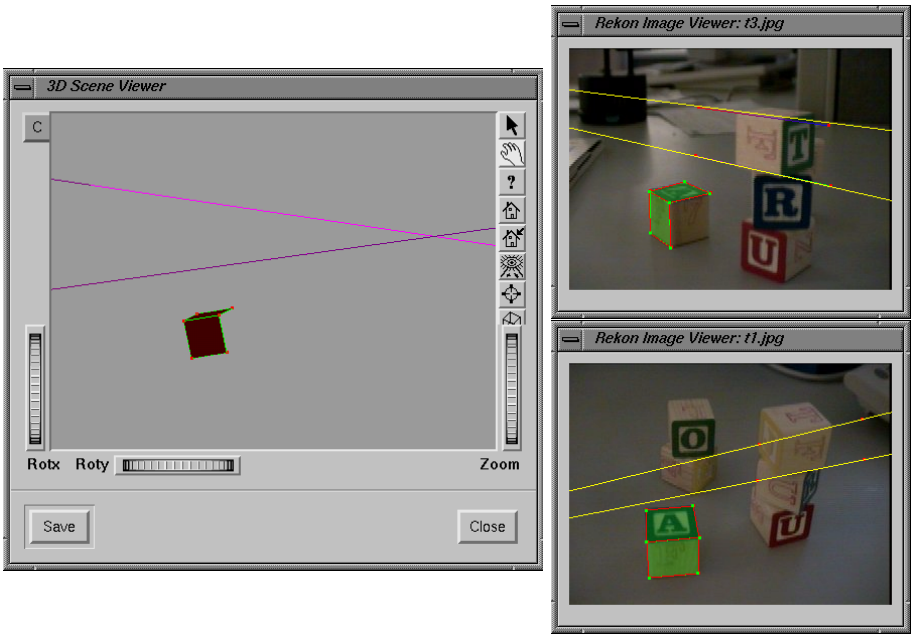


FIG. 3.8: Lignes dont la position 3D est incorrecte mais dont la projection sur l'image est exacte.

Ces défauts nuisent à la qualité visuelle du modèle et s'avèrent gênants pour l'observateur. Une solution est d'utiliser cet observateur afin de les corriger. En effet, celui-ci détecte plus ou moins immédiatement ces anomalies et est donc en mesure de les indiquer au système afin qu'il essaie de les rectifier. Le chapitre suivant décrit les contraintes additionnelles que l'usager peut spécifier entre des primitives afin de remédier aux problèmes observés.

Chapitre 4

Contraintes

> ***"[...] le beau rêve des choses inachevées où l'on se contente de souhaiter sans oser exiger, de promettre sans donner."***
>
> **— Stefan Zweig, *"Rêves oubliés"***

Si vous êtes en train de lire ceci au sein d'une ville ou à l'intérieur d'un bâtiment quelconque, prenez quelques instants pour regarder autour de vous. Que remarquez-vous ? La plupart des objets rigides qui nous entourent exhibent des caractéristiques inhérentes aux scènes façonnées par l'homme, telles la planarité, le parallélisme et la perpendicularité de leurs structures. Un nombre important d'objets est également constitué de surfaces de révolution et nous en discuterons brièvement en conclusion. Ceci découle évidemment du fait qu'il est plus facile et moins contraignant pour l'homme de construire des surfaces planes que des surfaces courbes, celles-ci étant plus présentes dans la nature.

Nous savons que la géométrie projective déforme ces caractéristiques. Il est donc plus difficile de les détecter automatiquement par des méthodes d'analyse d'images. Cependant, un observateur humain peut aisément les décrire dans la plupart des cas. Ce qu'il ne peut interpréter est la plupart du temps très ardu, voire impossible, à construire que ce soit réellement ou synthétiquement ; on n'a qu'à penser aux dessins d'Escher ou aux oeuvres d'architectes fantasques. Il nous semble donc naturel d'offrir à l'usager de notre système la possibilité de définir certaines de ses primitives comme planaires, coplanaires, perpendiculaires, parallèles, etc. Certains systèmes de vision vérifient la planarité des modèles reconstruits [Cha93] ; nous préférons l'imposer là où elle existe effectivement.

Lorsque l'usager dessine des primitives, celles-ci ne se trouvent pas exactement aux positions où elles devraient être pour deux raisons principales. Premièrement, de par la nature même des images qui constituent une version discrète "filtrée" et donc limitée du monde réel continu, rendant impossible un placement *exact* des primitives. Et deuxièmement car, l'erreur étant humaine, l'usager intro-

duit des erreurs de précision qui s'accumulent et finissent par se refléter de manière plus ou moins visible sur le modèle en reconstruction. Un moyen efficace pour améliorer ce positionnement consiste à utiliser des images de très haute qualité associées à un outil de *zoom* qui permettrait de disposer les primitives avec plus d'exactitude. Certaines techniques de vision permettent également d'ajuster automatiquement des primitives à des contours de l'image ou des textures lorsque l'usager indique leurs positions approximatives (par exemple, par l'utilisation de la transformée de Hough [IK88] pour détecter les segments). Le fait d'ajouter des contraintes entre les primitives dans l'espace 3D permet d'obtenir des améliorations notables qui corrigent les erreurs introduites par l'usager de manière plus rapide et efficace que l'ajustement manuel de ces primitives.

D'autre part, nous avons pu observer que plus on s'éloigne des primitives de calibration d'une image, i.e. les primitives 2D qui ont servi à calculer la matrice de projection de l'image, moins la reconstruction est satisfaisante. Les positions sont de moins en moins exactes au fur et à mesure que l'on s'écarte de ces primitives de calibration. Les polygones, par exemple, sont de plus en plus "inclinés" ou "étirés" par rapport à leur vraie position. Une solution est d'essayer, dans la mesure du possible, d'"entourer" la scène par les primitives de calibration. Une illustration de ce phénomène est présentée au chapitre 5.

Nous avons donc voulu vérifier si l'ajout de contraintes au niveau des primitives 3D pouvait permettre de corriger les problèmes observés. Ces contraintes se devaient d'être simples à exprimer et à utiliser. L'exigence principale afin de pouvoir les intégrer dans notre système était qu'elles soient exprimables sous forme d'équations linéaires. Toutefois, cela élimine plusieurs types de contraintes possibles, dont toutes celles faisant référence à des entités telles longueur ou angle. Une fois l'utilité des contraintes additionnelles démontrée, le système pourra être étendu à des contraintes non-linéaires.

Il est donc possible pour l'usager de choisir des primitives et de les relier par des contraintes de différentes natures. Nous avons classifié ces contraintes en trois catégories :

Contraintes de type 2D-3D Ce sont des contraintes qui relient les primitives 2D dessinées sur les images et les primitives 3D du modèle en reconstruction.

Contraintes de type 2D-2D Ce sont des contraintes qui relient uniquement les primitives 2D des images entre elles.

Contraintes de type 3D-3D Ce sont des contraintes qui expriment une relation existant entre deux primitives de l'espace.

Il est à noter que dans ce qui suit, nous utiliserons le qualificatif "valide" pour une primitive 3D pour signifier que cette primitive a été manuellement entrée par l'usager ou calculée par le système.

Une primitive invalide (ou indéfinie) sera donc une primitive dont au moins une des coordonnées est inconnue.

4.1 Contraintes 2D-3D

Une contrainte 2D-3D relie une primitive 2D se trouvant dans une image à une primitive 3D du modèle. La seule contrainte de ce genre est celle que nous avons décrite au chapitre 3 et qui concerne la relation entre une primitive 2D et sa position dans l'espace, i.e. la *contrainte de position*.

4.1.1 Position

Définition

Des contraintes de position relient *toutes* les primitives 2D à leur correspondante 3D dans l'espace. Ainsi, plusieurs primitives 2D mises en correspondance seront reliées à la même primitive 3D, qu'elle soit valide ou non, tandis que l'inverse n'est pas possible. Ce sont les contraintes utilisées lors de la phase de calibration.

Spécification et utilisation

Une contrainte de position est automatiquement créée lorsque l'usager ajoute une primitive 2D dans le système. Sa position 3D est donc indéfinie jusqu'à ce que l'usager la spécifie lui-même ou jusqu'à ce que le système dispose d'assez d'informations pour la calculer.

Ce sont les contraintes de position (plus précisément, les contraintes de position des points) qui nous permettent de calculer les paramètres de la matrice de projection tel que décrit au chapitre 3.

Lorsque l'on recherche la position d'un point, on veut calculer ses trois coordonnées spatiales. L'usager a la possibilité de les entrer manuellement (pour les besoins de la calibration par exemple). Pour un polygone, on recherche les positions de tous ses sommets et pour une ligne les six coordonnées de sa représentation de Plücker.

4.2 Contraintes 2D-2D

Les contraintes 2D-2D relient entre elles les primitives 2D se trouvant sur les images. Ce sont des contraintes utilisées lors de la phase de reconstruction.

4.2.1 Correspondance

Définition

Lorsque l'usager met des primitives 2D en correspondance d'une image à l'autre, il indique spécifiquement que ces primitives sont la projection de la même structure 3D. Ces primitives 2D se trouvent donc en *contrainte de correspondance*. Une contrainte de correspondance est constituée d'une liste de primitives 2D et de la primitive 3D qui leur est associée. Evidemment, il est impossible de mettre en correspondance deux primitives 2D appartenant à la même image.

Spécification et utilisation

La mise en correspondance de primitives 2D s'effectue par l'usager qui indique les primitives à relier. Les contraintes de position des primitives choisies sont alors modifiées en conséquence car ces primitives ont maintenant une position 3D commune. De même, les contraintes associées à chacune des primitives mises en correspondance sont fusionnées. C'est le gestionnaire de contraintes qui s'occupe de modifier la base de contraintes en conséquence lorsqu'une contrainte de correspondance est ajoutée dans le système.

Nous avons décrit précédemment comment une contrainte de correspondance permet d'obtenir la position 3D d'un point ou d'une ligne. Il suffit, pour chacune de ces primitives, d'avoir au moins deux projections de leur position dans l'espace 3D pour pouvoir calculer cette position au sens des moindres carrés à l'aide des contraintes géométriques décrites au chapitre 3. Il faut bien évidemment qu'au moins deux des matrices de transformation qui ont permis d'obtenir ces projections soient disponibles.

En ce qui concerne les polygones, deux polygones mis en correspondance provoquent la génération de correspondances entre tous leurs sommets. Dès que les caméras des images auxquelles appartiennent ces polygones sont disponibles, il est possible de calculer les positions 3D de leurs sommets et ainsi d'obtenir le polygone 3D correspondant.

4.3 Contraintes 3D-3D

Les contraintes 3D-3D expriment des relations entre des primitives dans l'espace 3D. Ces relations existent en trois dimensions mais sont plus difficiles à détecter dès qu'on effectue une projection perspective car elles perdent leurs propriétés euclidiennes. Ce sont des contraintes prises en compte lors de la phase de reconstruction et qui permettent d'améliorer le modèle obtenu uniquement avec

les contraintes de correspondance.

Certaines des contraintes nécessitent que des valeurs 3D pour certaines primitives aient été préalablement calculées afin de pouvoir les utiliser pour calculer d'autres grandeurs 3D telles des normales de plans. Il faut donc parfois déjà avoir effectué une itération de convergence avant de pouvoir obtenir des résultats par l'addition de contraintes.

De plus, nous allons voir que l'introduction de certaines contraintes permet d'obtenir la position de primitives 3D dont moins de deux projections sont disponibles ou pour lesquelles aucune contrainte de correspondance n'a été établie, ce qui constitue un avantage très intéressant.

4.3.1 Planarité

Définition

Une contrainte de planarité exprime le fait que tous les points d'un polygone doivent se retrouver sur un même plan. Pour exprimer cette contrainte, nous associons tout simplement une normale à chaque polygone 3D de notre scène. Cette normale définit l'orientation du plan sur lequel le polygone devrait se trouver.

Spécification et utilisation

Cette contrainte peut être spécifiée de deux manières différentes. On applique à un polygone soit une *contrainte de normale*, soit une *contrainte de planarité de polygone*.

Contrainte de normale

L'usager entre lui-même la normale d'un polygone ou celle-ci est calculée automatiquement dès que l'usager a entré manuellement la position d'au moins trois sommets du polygone. Nous avons également inclus la possibilité de pouvoir enlever ou modifier une normale à un polygone (celle-ci sera également enlevée ou modifiée dans tous les polygones qui lui sont parallèles).

Lorsque la normale $N = \begin{bmatrix} a & b & c \end{bmatrix}^T$ d'un polygone est disponible, il nous reste à calculer le terme indépendant des coordonnées d'un plan, i.e. le coefficient d de l'équation du plan $ax + by + cz + d = 0$. Pour cela, nous utilisons tous les points valides $P_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ du polygone afin de calculer un d_{moy} qui représente la moyenne de tous les d_i tels que

$$d_i = -(ax_i + by_i + bz_i).$$

Si les points du polygone ont été entrés manuellement par l'usager, d_{moy} devrait correspondre au d

"exact" de chacun des points. En effet, si l'usager les a spécifiés correctement, il devrait les avoir placés sur un même plan.

Contrainte de planarité de polygone

Lorsqu'aucune normale n'est disponible pour le polygone, on calcule le "plan moyen" pour ce polygone, i.e. le plan le plus près, au sens des moindres carrés, de tous les sommets valides du polygone.

Pour calculer ce plan $\Pi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$, on construit un système d'équations linéaires de la forme $\mathbf{A}\mathbf{x} = \mathbf{b}$ où $\mathbf{A}$ contient les coordonnées homogènes normalisées des points valides du polygone, $\mathbf{x}$ les coefficients inconnus du plan et $\mathbf{b}$ est un vecteur nul. Ce système reflète le fait que tous les sommets du polygone devraient appartenir à un même plan et donc en vérifier l'équation. Ainsi,

$$P \begin{pmatrix} P_x & P_y & P_z & 1 \end{pmatrix} \in \Pi \begin{pmatrix} a & b & c & d \end{pmatrix} \quad \text{si} \quad P \cdot \Pi = aP_x + bP_y + cP_z + d = 0. \tag{4.1}$$

Afin d'éviter la solution triviale $\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, un des éléments de $\mathbf{x}$ doit être normalisé en lui assignant une valeur fixe de 1. On ne peut choisir cet élément arbitrairement car le plan cherché pourrait être parallèle à un ou deux axes de coordonnées et donc posséder un ou deux coefficients nuls. Afin d'éviter un mauvais choix, on approxime la normale du polygone. Pour cela, on choisit trois de ses sommets P_1, P_2 et P_3 afin de former le produit vectoriel $P_1P_2 \times P_1P_3$ qui permet d'obtenir un vecteur perpendiculaire au plan contenant les trois points. Ce vecteur constitue une approximation de la normale réelle du polygone puisque les polygones reconstruits sont en général proches de leur position exacte. L'élément normalisé choisi sera celui qui correspond au coefficient de la normale calculée dont la valeur absolue est la plus élevée. La résolution du système d'équations se fera donc pour les trois autres coefficients du plan et nous permettra de trouver le "meilleur" plan de support du polygone au sens des moindres carrés.

Une fois le plan de support Π du polygone calculé, d'une manière ou de l'autre, on peut introduire une nouvelle contrainte à la position d'un point P de ce polygone. Ainsi,

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = -d \tag{4.2}$$

exprime le fait que $P \in \Pi$ si $P \cdot \Pi = 0$.

Le sens de la normale n'a pas d'importance dans ce calcul puisqu'on ne se sert de cette normale que pour définir l'orientation du plan de support du polygone.

Lorsque l'usager choisit d'utiliser les contraintes de normale, le système prend en compte dans son calcul toutes les normales entrées par l'usager ou calculées à partir des positions qu'il a entrées. Lorsqu'il choisit d'utiliser les contraintes de planarité de polygone, le système calcule automatiquement la "meilleure" normale de chaque polygone (à l'aide de l'équation 4.1) et la considère dans le calcul de la position de chacun de ses sommets.

Résultats

Il est presque impossible que tous les points calculés d'un polygone 3D de plus de trois sommets se retrouvent exactement sur le même plan. Une contrainte de planarité nous semblait donc nécessaire pour corriger ce défaut. Les résultats obtenus avec l'introduction de cette contrainte sont très appréciables comme on peut le constater sur la figure 4.1.

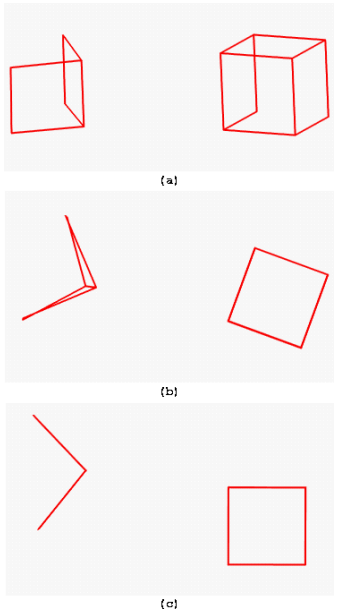


FIG. 4.1: Résultat de l'application de la contrainte de planarité de polygone. (a) Un cube entré manuellement et deux polygones, calculés sans contraintes de planarité. (b) Vue de haut ; les deux polygones ne sont pas planaires. (c) Les polygones se sont replacés après l'application de la contrainte de planarité de polygone.

Il faut cependant noter que ce n'est pas parce qu'on a une contrainte de planarité sur un polygone que ses sommets se retrouveront exactement sur le même plan. Cette contrainte ne fera que s'ajouter à celles composant le système d'équations formé pour le calcul de la position d'un point. La résolution de ce système par moindres carrés donnera à la contrainte autant d'importance qu'aux autres (sauf en cas de contraintes pondérées, comme ceci sera abordé au chapitre 6). De plus, chaque sommet d'un polygone est calculé indépendamment des autres, selon les contraintes qui s'y appliquent. La

planarité du polygone n'est donc pas *forcée* mais *influencée*. En pratique, on a mesuré des distances points-plan de support de l'ordre de 0.0001 lorsque les points 3D de la scène sont normalisés entre 0 et 1 et que seules les contraintes de planarité et de coplanarité sont utilisées, ce qui est très satisfaisant. Par contre, cela dépend également des autres contraintes qui concernent les primitives, la distance au plan pouvant augmenter lorsque les primitives sont plus contraintes. Le résultat obtenu au niveau de la distance point-plan de support est sûrement dû au fait qu'il existe plusieurs solutions possibles pour que la position d'un point satisfasse seulement ses contraintes de position. En effet, on a vu plus tôt que la résolution par moindres carrés minimisait la distance d'un point à toutes les droites (ou plans) 3D qui le définissaient (section 2.1.2). Il est possible qu'il existe plusieurs positions du point qui minimisent cette distance. L'ajout de la contrainte de planarité permet de choisir la position la plus proche du plan de support calculé.

De plus, la présence d'une contrainte de planarité pour un polygone permet de calculer la position de points qui n'ont pas été mis en correspondance (par exemple, s'ils ne sont visibles que dans une seule image). En effet, il faut trois équations pour trouver les trois coordonnées d'un point. Nous avons vu qu'une contrainte de position en génère deux (équations 3.4 et 3.5). L'ajout d'une contrainte de planarité pour le polygone auquel appartient le point permet d'obtenir la troisième équation (équation 4.2). On peut ainsi obtenir la position du point dans l'espace 3D.

4.3.2 Coplanarité

Définition

Une contrainte de coplanarité exprime le fait que plusieurs primitives se retrouvent sur le même plan, par exemple des affiches sur un mur ou la base d'objets se trouvant sur une table. Similairement à la contrainte précédente mais en plus général, cette contrainte nous permet de calculer un plan moyen pour l'ensemble des points formé par les primitives (points, lignes, polygones) définies comme étant coplanaires et de rapprocher les primitives de ce plan.

Spécification et utilisation

L'usager indique les primitives 3D coplanaires dans la scène en sélectionnant leurs primitives 2D correspondantes dans les images. Lors de la reconstruction de géométrie, le système calcule, pour chaque ensemble de primitives coplanaires, le plan "moyen" (i.e. le plus proche) des points provenant des différentes primitives (points, sommets des polygones et couples de points extraits des lignes). Ce plan est calculé de la même manière que pour une contrainte de planarité de polygone (voir section

4.3.1). Une contrainte de la même forme que celle de planarité (équation 4.2) est ajoutée au calcul de la position 3D de chacun des points appartenant au plan moyen, i.e. de chacun des points liés par la contrainte de coplanarité.

Cette contrainte s'avère très utile pour indiquer les relations de *contact* entre parties du modèle. Par exemple, si on veut pouvoir spécifier qu'un objet se trouve *sur* un autre, on peut simplement définir les points de contact de l'objet du dessus comme étant coplanaires avec la/les primitives définissant le haut de l'autre objet. On rapprochera ainsi les deux objets l'un de l'autre jusqu'à ce qu'ils semblent se toucher dans le modèle reconstruit. Pour la plupart des objets dans les scènes réelles, il existe presque toujours une face que l'on ne voit pas (les objets ne flottent d'habitude pas en l'air). On peut deviner cette face et la dessiner approximativement sur les images, puis indiquer une contrainte de coplanarité entre cette facette et l'objet sur lequel elle repose. On peut également contraindre les sommets visibles de sa face cachée à être coplanaires avec la face visible de l'autre objet.

Lorsque l'on définit une nouvelle relation de coplanarité entre un polygone et une autre primitive, on vérifie si le polygone se trouve déjà dans une liste de coplanarité. Si c'est le cas, on ajoute à cette liste la deuxième primitive de la relation. En effet, étant donné qu'il n'y a qu'un seul plan qui puisse passer par un polygone, un polygone ne peut appartenir qu'à un seul ensemble de primitives coplanaires. Il n'en est pas de même pour un point ou une ligne puisqu'ils peuvent appartenir à une infinité de plans.

Résultats

Cette contrainte fournit de nettes améliorations au modèle reconstruit comme on peut le voir aux figures 4.2 et 4.3. On peut aussi y remarquer l'apparition de nouveaux points 3D dont la position a pu être calculée grâce à l'ajout de la contrainte supplémentaire de coplanarité. Il est important de noter qu'ici aussi nous n'obtiendrons pas de coplanarité exacte dû à la méthode de résolution par moindres carrés. Les primitives se rapprocheront d'un plan moyen tout en tentant également de satisfaire les autres contraintes auxquelles elles sont soumises.

4.3.3 Parallélisme

Définition

Une contrainte de parallélisme exprime le fait que des primitives sont parallèles dans la scène. Cette caractéristique se retrouve très souvent autour de nous et il est important de pouvoir l'exploiter dans notre système. Deux droites parallèles en 3D ne le seront plus une fois projetées en perspective.

(a)

(b)

FIG. 4.2: Influence de la contrainte de coplanarité. (a) Primitives “flottantes”. (b) Primitives réunies par l’application de la contrainte de coplanarité entre les points des primitives du haut et le polygone du bas.

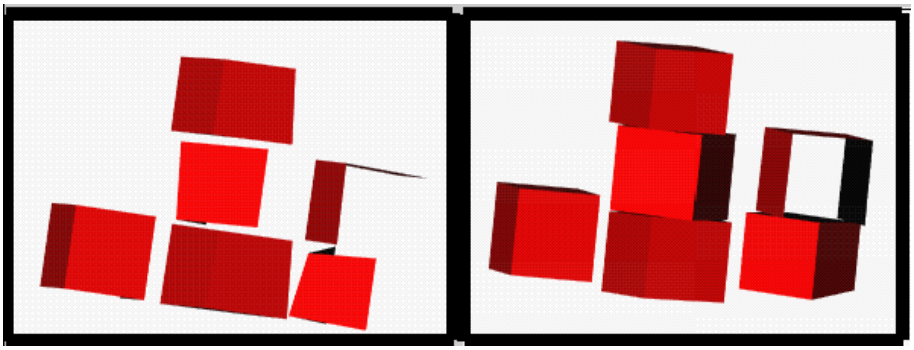


FIG. 4.3: Influence de la contrainte de coplanarité. (a) Primitives “flottantes” (b) Primitives corrigées.

Inversement, deux droites parallèles sur une image ne le sont pas forcément dans l'espace 3D. Il est ainsi difficile de détecter automatiquement cette caractéristique (par exemple, en utilisant les points de fuite des droites de l'image). Il est préférable que l'usager l'indique explicitement car il utilise la sémantique de la scène et peut ainsi donner plus d'information, et souvent plus facilement, qu'un algorithme de détection automatique.

Spécification et utilisation

Parallélisme de polygones

THÉORÈME 1 (CONDITION DE PARALLÉLISME DE DEUX PLANS)

Si les plans

$$a_1x + b_1y + c_1z + d_1 = 0 \qquad \textit{et} \qquad a_2x + b_2y + c_2z + d_2 = 0$$

sont parallèles, les vecteurs normaux $N_1 = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}^T$ *et* $N_2 = \begin{bmatrix} a_2 & b_2 & c_2 \end{bmatrix}^T$ *sont colinéaires*[1] *(et inversement).*

L'usager n'a qu'à spécifier les polygones parallèles dans la scène en sélectionnant leurs primitives 2D correspondantes sur les images. En utilisant les positions 3D de ces polygones, calculées lors d'une première itération de convergence, le système peut calculer ou récupérer les normales d'une liste de polygones parallèles. Il peut ainsi calculer une normale *moyenne* $N_{moy} = \begin{bmatrix} a & b & c \end{bmatrix}^T$, où

$$N_{moy} = \frac{\sum_i N_i}{n},$$

avec N_i la normale du polygone i et n le nombre de polygones de la liste de polygones parallèles dont la normale est disponible. Cette normale moyenne sera ensuite attribuée à tous les polygones de la liste. Evidemment, si l'un des polygones possède une normale entrée par l'usager ou calculée à partir de points entrés par l'usager, celle-ci est tout simplement propagée à tous les autres polygones parallèles.

Il est à noter qu'il faut faire attention ici à l'orientation (sens) des normales qui entrent dans le calcul de la normale moyenne (voir les normales N_2 et N_3 de la figure 4.4). En effet, deux plans parallèles peuvent avoir des normales de sens contraire, bien que de même direction. Faire la moyenne de telles normales introduiraient alors des erreurs énormes puisque l'on obtiendrait un vecteur orienté de façon tout à fait différente. Il nous faut donc choisir arbitrairement une des deux orientations de normales possibles (par exemple, celle du premier polygone de la liste) et renverser celles de sens

[1] *Par définition, des vecteurs portés par des droites parallèles sont dits* colinéaires.

contraire. Afin de déterminer les normales de sens contraire, nous formons le produit scalaire des deux normales. Si celui-ci est inférieur à 0, l'angle formé par les deux normales est obtus ($> 90^o$) et les deux normales sont considérées de sens contraire. Cette hypothèse peut s'avérer incorrecte dans le cas où les approximations que l'on utilise sont fausses, mais la reconstruction est alors tellement mauvaise qu'il est de toute façon risqué de vouloir y ajouter des contraintes. Ceci n'est encore jamais arrivé lors de nos tests.

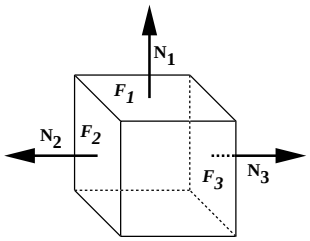


FIG. 4.4: Diverses normales sur une forme simple.

Une fois cette normale moyenne N_{moy} obtenue, on peut ajouter une contrainte au calcul de chacun des sommets des polygones. On voudrait que la normale du plan de support du polygone auquel appartient chaque sommet se rapproche le plus possible de la normale moyenne calculée. On peut donc ajouter une contrainte de normale au calcul de la position d'un sommet du polygone, tel que décrit dans la section 4.3.1 et exprimé par l'équation 4.2.

Parallélisme de lignes

Dans le plan, deux lignes sont parallèles lorsqu'elles ne s'intersectent pas. Cette définition n'est plus valide dans l'espace puisqu'il est possible que deux lignes 3D ne s'intersectent pas et ne soient pas pour autant parallèles (*skew lines*). Deux lignes sont parallèles en 3D si elles ont la même direction et se trouvent donc sur un même plan.

DÉFINITION 1 (PARALLÉLISME DE LIGNES 3D)
Deux lignes 3D sont parallèles lorsqu'elles sont coplanaires et ne s'intersectent pas.

L'indication de lignes parallèles se fait de manière similaire aux polygones en sélectionnant les lignes 2D correspondant aux lignes 3D qui sont parallèles. Nous avons choisi deux contraintes parmi celles possibles pour représenter cette relation.

Rappelons tout d'abord que pour calculer la position d'une ligne 3D, nous cherchons à retrouver les six éléments du vecteur représentant les coordonnées de Plücker de la ligne, $L = \begin{bmatrix} P_{xy} & P_{xz} & P_{yz} & P_{xh} & P_{yh} & P_{zh} \end{bmatrix}^T$ (voir section 3.4.3). La direction de L est donnée par le vecteur

$D = \begin{bmatrix} P_{xh} & P_{yh} & P_{zh} \end{bmatrix}^T$ et la normale au plan Π_{orig} passant par la ligne et l'origine est donnée par $N_{orig} = \begin{bmatrix} P_{yz} & P_{zx} & P_{xy} \end{bmatrix}^T$ (voir l'annexe A pour plus de détails). Nous avons donc évidemment $D \perp \Pi_{orig}$ puisque la direction d'une ligne est perpendiculaire à la normale de tout plan passant par cette ligne.

Soient deux lignes parallèles L_1 et L_2, et donc leurs directions D_1 et D_2 sont égales. Il passe un seul plan Π_1 par L_1 et l'origine (que nous appellerons le plan *ligne-origine* de L_1). Puisque L_2 a la même direction que L_1, il existe un plan parallèle à Π_1 passant par L_2. La direction D_2 de L_2 est donc elle aussi perpendiculaire à la normale N_1 de Π_1 (voir la figure 4.5). Dans notre première méthode, nous allons imposer à la direction de toute ligne L d'être perpendiculaire à toutes les normales des plans ligne-origine des lignes qui lui sont parallèles.

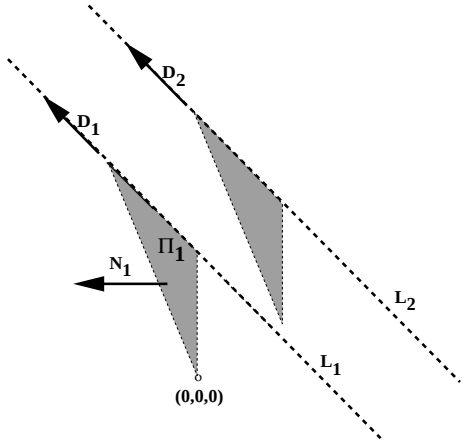


FIG. 4.5: Construction pour illustrer le parallélisme de lignes 3D.

Soit une ligne L de direction $D = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T$. Soit un ensemble de lignes L_i parallèles à L. Si la normale au plan ligne-origine de L_i est définie comme $N_i = \begin{bmatrix} a_i & b_i & c_i \end{bmatrix}^T$, nous pouvons ajouter, pour chaque ligne L_i parallèle à L, une équation au système destiné à calculer la position de L :

$$\begin{bmatrix} 0 & 0 & 0 & a_i & b_i & c_i \end{bmatrix} \begin{bmatrix} P_{xy} \\ P_{xz} \\ P_{yz} \\ P_x \\ P_y \\ P_z \end{bmatrix} = 0,$$

qui représente le fait que $N_i \perp D$, pour toutes les lignes L_i. Etant donné que nous utilisons une technique de résolution par moindres carrés, ces équations devraient permettre de contraindre la position de la ligne en tenant compte des directions de toutes les lignes qui lui sont parallèles.

Pour notre seconde méthode, nous utilisons une approche similaire au parallélisme de polygones. Le système tente de calculer une direction *moyenne* $D_{moy} = \begin{bmatrix} D_x & D_y & D_z \end{bmatrix}^T$ pour toutes les lignes d'un ensemble de lignes parallèles où

$$D_{moy} = \frac{\sum_i D_i}{n}$$

avec D_i la direction "normalisée" de la ligne L_i et n le nombre de lignes parallèles à L.

Pour chaque ligne L_i de l'ensemble des lignes parallèles, on a $D_{moy} \perp N^i_{orig}$ et donc

$$\begin{bmatrix} D_x & D_y & D_z \end{bmatrix} \begin{bmatrix} P_{yz} \\ P_{zx} \\ P_{xy} \end{bmatrix} = 0.$$

On peut donc ajouter au calcul de chacune des lignes de cet ensemble l'équation

$$\begin{bmatrix} D_z & -D_y & D_x & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{xy} \\ P_{xz} \\ P_{yz} \\ P_x \\ P_y \\ P_z \end{bmatrix} = 0.$$

Propagation

Une contrainte de parallélisme doit être propagée à travers le système car c'est une propriété transitive :

PROPRIÉTÉ 1

Soient trois plans Π_1, Π_2 et Π_3. Si $\Pi_1 \parallel \Pi_2$ et $\Pi_2 \parallel \Pi_3$, alors par transitivité, $\Pi_1 \parallel \Pi_3$.

Ainsi, si l'on a défini deux listes distinctes de primitives parallèles et que l'on ajoute une contrainte de parallélisme entre deux primitives appartenant respectivement à chacune des listes, il faut fusionner les listes correspondantes.

Résultats

Parallélisme de polygones

La figure 4.6 montre l'amélioration obtenue par l'ajout de contraintes de parallélisme entre les diverses primitives d'une scène.

Sur toutes les scènes testées, nous avons pu nous rendre compte que la contrainte de parallélisme est essentielle pour corriger les défauts qui surviennent lorsque l'on n'utilise que des contraintes de correspondance.

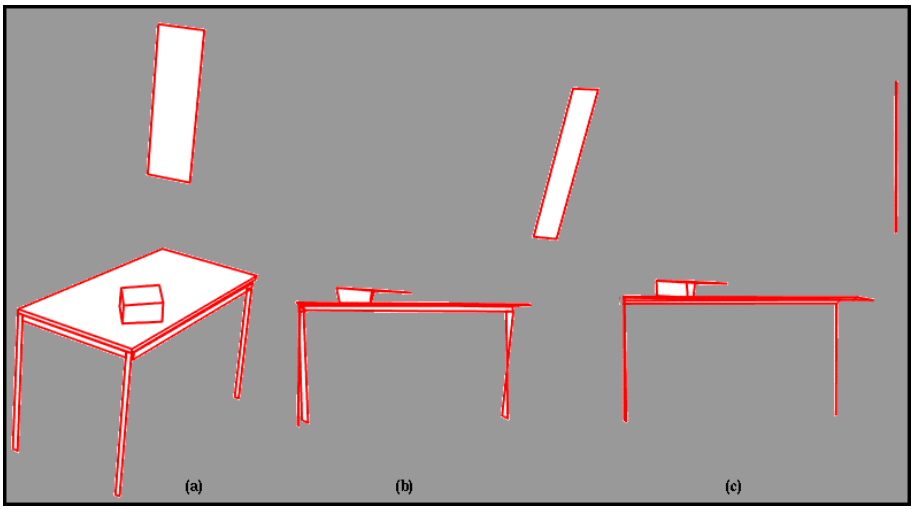


FIG. 4.6: Résultat obtenu par l'application de la contrainte de parallélisme de polygones. (a) Vue d'une scène reconstruite. (b) Vue de côté de la même scène avant l'application de contraintes. On remarque les défauts au niveau des pattes de la table, de la boîte sur la table et du cadre sur le mur. (c) Résultat après l'application de la contrainte de parallélisme.

Parallélisme de lignes

Les deux méthodes de calcul d'une contrainte de parallélisme de lignes n'ont pas été comparées formellement mais donnent visuellement des résultats similaires pour les exemples testés. Par contre, la première méthode ajoute au calcul d'une ligne autant d'équations qu'elle a de lignes qui lui sont parallèle tandis que la deuxième méthode ajoute une équation au calcul de chacune des lignes d'une liste de lignes parallèles. Il nous semble donc que la deuxième méthode permet de donner aux contraintes de parallélisme un poids égal aux autres contraintes et est ainsi plus correcte. C'est celle que nous utilisons.

La figure 4.7 (a) montre deux lignes affichant un problème de parallélisme qui sera corrigé par l'application d'une contrainte de parallélisme entre les deux lignes. On en voit le résultat à la figure 4.7 (b).

Le parallélisme appliqué aux segments est une fonctionnalité qu'il reste à implanter mais qui s'avèrera très utile pour corriger les défauts que l'on peut encore remarquer dans certaines scènes au niveau des côtés des polygones. Cette extension sera simple à ajouter puisque les deux sommets définissant un segment peuvent également définir la ligne qui passe par le segment. Il suffit alors simplement d'appliquer à cette ligne les contraintes de parallélisme décrites précédemment et la position des points (et donc du segment) sera modifiée en conséquence.

La parallélisme entre un polygone et une ligne constitue également un ajout intéressant et facile à implanter. Il ne s'agit que de définir la direction de la ligne comme étant perpendiculaire à la normale

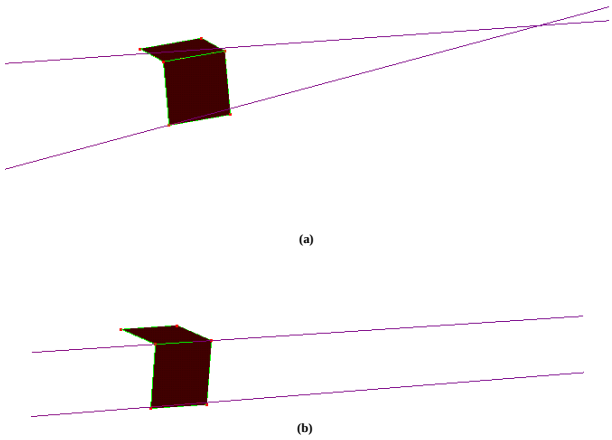


FIG. 4.7: Résultat obtenu par l'application de la contrainte de parallélisme de lignes. (a) Deux lignes qui devraient être parallèles mais ne le sont pas. (b) Les lignes de la figure (a) après ajout d'une contrainte de parallélisme entre les deux lignes.

du polygone.

4.3.4 Perpendicularité

Définition

Une contrainte de perpendicularité exprime le fait que deux primitives sont perpendiculaires. On peut retrouver cette caractéristique aussi fréquemment que le parallélisme dans les scènes architecturales.

Nous avons pu observer que plus on s'éloignait dans l'espace des primitives de calibration, plus les polygones devenaient inclinés comme on peut le remarquer sur l'image à la figure 4.8 où la primitive ayant servi à calibrer l'image est le petit cube à droite. Cette contrainte s'est donc avérée nécessaire afin de corriger ce problème. La contrainte de parallélisme appliquée aux côtés des boîtes a permis de corriger une partie du problème. Toutefois, les boîtes continuaient d'être inclinées, même si leurs côtés étaient parallèles deux à deux.

Spécification et utilisation

Tout comme pour le parallélisme, l'usager indique une relation de perpendicularité en sélectionnant les primitives 2D correspondant aux primitives 3D concernées. Des couples de primitives perpendiculaires sont alors insérés dans le système. Le gestionnaire de contraintes s'occupe de propager cette contrainte de perpendicularité à travers le système. En effet, toutes les primitives

parallèles à l'une des deux primitives de la relation sont également perpendiculaires à l'autre. Le gestionnaire va donc insérer autant de couples de primitives perpendiculaires qu'il y a de primitives dans les deux listes de parallélisme associées aux primitives mises en contrainte de perpendicularité initialement.

Perpendicularité de polygones

THÉORÈME 2 (CONDITION DE PERPENDICULARITÉ DE DEUX PLANS)

Si les plans

$$a_1x + b_1y + c_1z + d_1 = 0 \qquad \textit{et} \qquad a_2x + b_2y + c_2z + d_2 = 0$$

sont perpendiculaires, les vecteurs normaux $N_1 = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}^T$ *et* $N_2 = \begin{bmatrix} a_2 & b_2 & c_2 \end{bmatrix}^T$ *le sont aussi (et inversement). C'est pourquoi la condition nécessaire et suffisante de perpendicularité est*

$$a_1a_2 + b_1b_2 + c_1c_2 = 0.$$

Ceci est illustré à la figure 4.4 pour les faces F_1 *et* F_2 *du cube (après propagation de la contrainte, on aura également* $F_1 \perp F_3$ *puisque* F_1 *est parallèle à* F_3*).*

Soit Π_1 le plan de support d'un des deux polygones d'une relation de perpendicularité. Π_1 est orthogonal à l'autre plan Π_2 de la relation ainsi qu'à l'ensemble S de tous les polygones parallèles à Π_2, et inversement. Si l'on connaît la normale N_1 de Π_1, on peut ajouter une contrainte simple au calcul de la normale moyenne N_{moy} de S (voir section 4.3.3) puisqu'on a $N_1 \perp N_{moy}$ selon le théorème 2 et donc $N_1 \cdot N_{moy} = 0$.

Cette contrainte peut également être utilisée avec les contraintes de *planarité de polygone* (section 4.3.1) et de *coplanarité* (section 4.3.2) en ajoutant une équation au calcul du plan moyen $\Pi = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$. Si on a un polygone perpendiculaire à un polygone de ce plan moyen, sa normale $N' = \begin{bmatrix} a' & b' & c' \end{bmatrix}^T$ est également orthogonale à la normale du plan moyen $N = \begin{bmatrix} a & b & c \end{bmatrix}^T$ et ainsi

$$N \cdot N' = aa' + bb' + cc' = 0.$$

On peut donc ajouter au calcul de Π l'équation

$$\begin{bmatrix} a' & b' & c' & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0.$$

Perpendicularité de lignes

La méthode utilisée pour intégrer la perpendicularité de lignes est similaire à celle utilisée pour les polygones. En effet, deux lignes perpendiculaires ont des directions perpendiculaires. On peut donc utiliser les directions de chacune des lignes pour contraindre un peu plus la direction moyenne calculée pour les listes de lignes parallèles à chacunes de lignes de la relation d'orthogonalité.

Résultats

Les résultats obtenus par l'ajout de ce type de contrainte ont été très satisfaisants. En à peine quelques itérations de convergence (de l'ordre d'une dizaine), toutes les primitives inclinées se redressent comme on peut l'observer sur l'image de la figure 4.9.

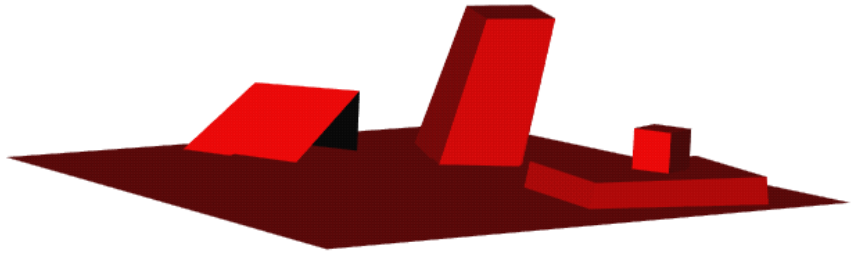


FIG. 4.8: Inclinaison des primitives éloignées de la primitive de calibration (le petit cube à droite).

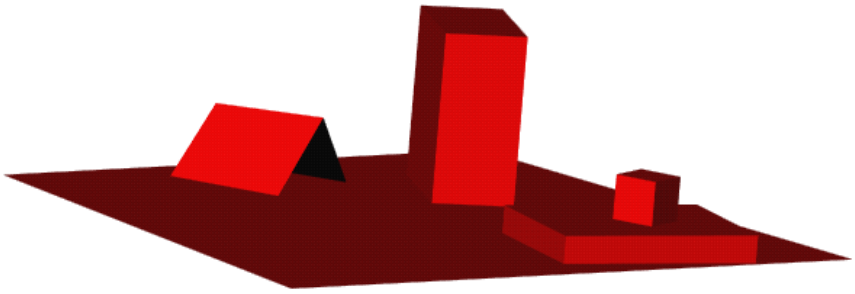


FIG. 4.9: Primitives inclinées corrigées par l'activation de la contrainte de perpendicularité.

4.3.5 Intersection de lignes

Si deux lignes 3D sont sécantes, cela implique deux propriétés

1. Les lignes sont coplanaires ;
2. Les lignes ont un point commun.

Pour calculer la position dans l'espace de ce point d'intersection, deux approches sont possibles.

- Si l'usager indique les deux lignes 2D, ainsi que leur point 2D d'intersection sur les images nous obtenons deux plans pour le point (les deux plans orthogonaux qui le définissent) et un plan par ligne, pour un total de quatre plans. Le point cherché appartient à tous ces plans et nous avons donc assez de contraintes pour pouvoir en calculer la position 3D. Il est important que le point 2D ne soit pas pris sur la même image que les deux lignes 2D puisqu'il y aura alors redondance d'informations, les quatre plans s'intersectant alors en une ligne 3D et non un point. Cette contrainte pourrait aussi se généraliser à l'*incidence* d'un point sur une ligne. En choisissant, *dans deux images différentes*, une ligne 2D et un point 2D sur cette ligne, on obtient trois plans pour calculer la position du point.
- Si le point d'intersection n'est pas visible, l'usager n'indique que les deux lignes. Chaque ligne donne un plan : celui qui passe par la ligne 2D et sa position 3D (qui a donc besoin d'être connue). Afin d'obtenir le plan manquant nous pouvons projeter l'une de ces deux lignes sur une autre image en utilisant la matrice de projection correspondante afin d'obtenir une ligne 2D sur l'image. Nous obtenons donc un troisième plan passant par la ligne 3D. Une fois les nouveaux plans dérivés, on peut calculer leur point d'intersection et ainsi obtenir le point d'intersection des lignes en 3D.

Ces deux approches supposent bien sûr que les matrices de transformation des images ainsi que les positions 3D des lignes sont disponibles afin de pouvoir calculer les plans nécessaires.

Cette contrainte n'a pu être implémentée en raison de contraintes de temps, mais cela peut se faire relativement rapidement. C'est une contrainte qui sera très utile pour former des polygones à partir de droites tracées dans des images différentes et pour faire se rejoindre des droites/segments. Par exemple, la reconstruction de grands murs pourrait être difficile si l'on doit absolument dessiner un polygone qui le recouvre entièrement et qu'aucune image ne permet de voir le mur en entier. Par contre, reconstruire ses côtés à partir de plusieurs images puis les réunir en un polygone une fois qu'ils sont tous disponibles, permettrait d'obtenir la position 3D du mur.

4.4 Extensions

Les résultats obtenus par l'ajout de toutes ces contraintes ont produit une nette amélioration sur tous les modèles testés. Il serait donc intéressant de continuer dans cette voie et d'ajouter d'autres contraintes pour offrir plus de possibilités à l'usager quant à l'amélioration de son modèle.

Il est important de remarquer que l'usager constate normalement assez vite ce qui est incorrect dans son modèle, par exemple deux murs d'un bâtiment qui ne sont pas perpendiculaires ou une table non planaire, il peut donc agir immédiatement en appliquant la contrainte appropriée. Disposer de contraintes additionnelles serait disposer de plus d'outils de "réparation" de modèles et amènerait une plus grande flexibilité et de meilleurs résultats.

Parmi les contraintes que nous jugeons comme pouvant avoir une influence sur la qualité d'un modèle se trouvent :

- Des *contraintes d'angle* pour lesquelles on spécifie l'angle entre deux structures.
- Des *contraintes de longueur* qui indiqueraient que deux primitives sont de longueur égale.
- Des *contraintes de symétrie* qui nous permettraient de ne reconstruire qu'une partie d'un objet symétrique et de laisser le système bâtir le reste une fois la contrainte spécifiée. C'est une des contraintes importantes dans *Façade*. Egalement, comme le font remarquer Weik et Grau [WG96], les humains sont très sensibles aux violations de relations géométriques attendues. Ils considèrent donc une contrainte de symétrie comme étant très importante pour obtenir des reconstructions "subjectivement" satisfaisantes.
- Des *contraintes de similarité* qui permettraient d'indiquer des objets identiques mais à des positions différentes, par exemple des fenêtres dans un bâtiment. Cela permettrait de réduire de beaucoup le temps de modélisation d'éléments répétitifs d'une scène, souvent présents dans les constructions de l'homme.

Des *primitives de plus haut niveau* que celles actuellement disponibles pourraient également être introduites mais leur but serait surtout de faciliter la tâche de l'usager plutôt que d'améliorer la reconstruction en tant que tel. Elles contiendraient implicitement un ensemble de contraintes. Par exemple, une primitive *cube* permettrait de réunir six primitives polygones contraintes par des relations de parallélisme, de perpendicularité et de longueur. De telles primitives pourraient être organisées dans des hiérarchies, comme dans *Façade*. *Façade* exploite extensivement et avec succès l'utilisation de ces primitives. Elles permettent également de réduire efficacement le nombre de contraintes en un ensemble non redondant.

A un autre niveau, l'utilisation de poids ou de facteurs d'incertitude associés à chaque *type* de contrainte ainsi qu'à une contrainte en particulier permettrait de mieux contrôler l'effet des contraintes. Dans certaines scènes, on peut juger un certain type de contrainte comme étant plus important globalement qu'un autre. De même, pour une certaine partie du modèle, on voudrait pouvoir indiquer qu'une contrainte spécifique est plus ou moins importante. Ainsi, la contrainte de position d'une primitive éloignée du point de vue ou d'un polygone vu d'un angle rasant devrait être moins importante qu'une contrainte de position d'une primitive proche de l'observateur ou d'un polygone vu de face. Une série de critères serait à déterminer et le système pourrait même suggérer des poids, par exemple en fonction de la surface qu'une primitive occupe sur l'image.

Chapitre 5

Résultats

Car tout ce que je raconte, je l'ai vu ; et si j'ai pu me tromper en le voyant, bien certainement je ne vous trompe point en vous le disant.

—Lettre à Stendhal

Nous présentons ici les résultats obtenus par notre système pour divers types de scènes, tant au niveau de la calibration que de la reconstruction. Certains résultats spécifiques aux contraintes utilisées ont déjà pu être observés au chapitre 4 et ont clairement démontré l'utilité si ce n'est la nécessité de ces contraintes dans l'amélioration des modèles 3D reconstruits.

5.1 Reconstruction des caméras

Dans cette section, nous évaluons notre méthode de calibration. Pour ce faire, nous ne pouvons utiliser une caméra synthétique et comparer la matrice correspondante avec celle que nous retrouvons. En effet, leurs divers paramètres peuvent être tout à fait différents mais aboutir quand même à la même projection. La précision d'un coefficient d'une matrice de projection n'a pas un lien direct avec la qualité de son approximation de la vraie caméra. Plusieurs valeurs totalement différentes peuvent aboutir à des projections similaires. On n'a qu'à penser à la diminution d'un facteur de *zoom* compensé par une translation vers la scène.

Nous évaluons donc principalement la qualité de notre calibration en reprojetant le modèle reconstruit à travers les matrices de projection calculées sur les images correspondantes. Cette reprojection se superpose aux primitives tracées sur les images et nous permet aussi de vérifier la précision du modèle reconstruit.

5.1.1 Importance de la répartition des primitives de calibration

Les figures 5.1 et 5.2 montrent la calibration de la même image. Dans la première, les six points utilisés pour la calibration ont été choisis dans le coin supérieur gauche de l'image (encerclés) tandis que dans la seconde, ils sont bien répartis à travers l'image (indiqués par des flèches). Les reprojections du modèle sur les images de droite à travers les matrices calculées avec ces six points parlent d'elles-mêmes. Il est à noter que bien que la matrice de projection de la première image soit de toute évidence erronée, la reprojection des points qui ont servi à la calculer est correcte, ce qui est tout à fait normal puisqu'on calcule une matrice qui doit transformer des points 3D donnés en des points 2D aux positions indiquées.

Ainsi, pour obtenir une bonne calibration d'une image, il est *essentiel* que les primitives choisies soient bien réparties à travers *l'image*. Ceci découle du fait que la matrice ne fait qu'approximer le processus qui a permis d'obtenir l'image. En ne prenant que des points isolés dans un coin de l'image, on utilise un échantillon non représentatif des points de l'image et donc de la transformation que leurs équivalents 3D ont subie pour arriver à ces positions.

La solution de calibration consiste à minimiser l'erreur globale d'un système d'équations dans lequel chaque équation concerne la transformation effectuée, à travers une matrice, sur un point 3D pour le faire correspondre avec un point 2D. La solution sera donc celle qui fera correspondre le mieux ces points 3D transformés aux points 2D indiqués sur l'image. Si tous les points 2D se retrouvent dans une petite région de l'image (et/ou de l'espace), il se peut (et il est fort probable) qu'il y ait plusieurs matrices de transformations qui arrivent au même résultat *pour cette région de l'image*. Par contre, si les points sont bien répartis à travers l'image, le nombre de solutions pour la matrice diminue. Notre calibration est ainsi plus "contrainte".

De plus, les erreurs d'imprécision (dues au traçage des primitives par un usager) s'accumulent dans une région de l'image et ont des répercussions négatives sur la qualité de la matrice plus importantes que si ces erreurs étaient distribuées dans l'image. On demande au système de faire correspondre une petite partie du modèle à un ensemble de primitives 2D ; c'est ce qu'il fait mais une erreur sur une zone restreinte se trouve amplifiée dans le reste. Il est donc normal que l'approximation de la transformation réelle à l'échelle de l'image entière soit moins bonne, voire complètement fausse.

De même, si on s'efforce d'"entourer" la *scène 3D* par les primitives de calibration *fixes* (celles dont on entre les positions manuellement), on évitera le problème des primitives qui s'inclinent lorsqu'elles se retrouvent loin des primitives de calibration (figure 4.8). Malheureusement, cela est souvent difficile car il est plus facile d'utiliser un objet de la scène dont on mesure ou estime les dimen-

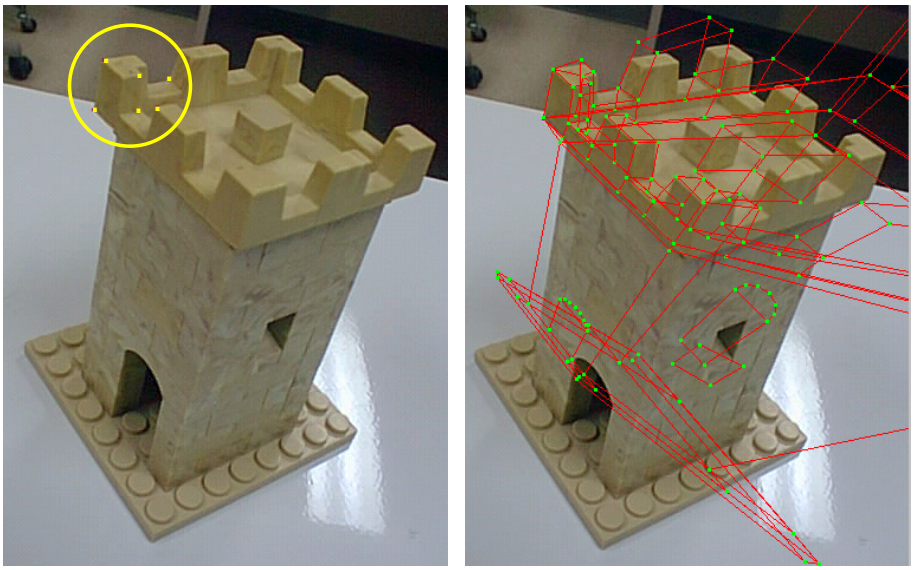


FIG. 5.1: Une image calibrée avec des points mal répartis.

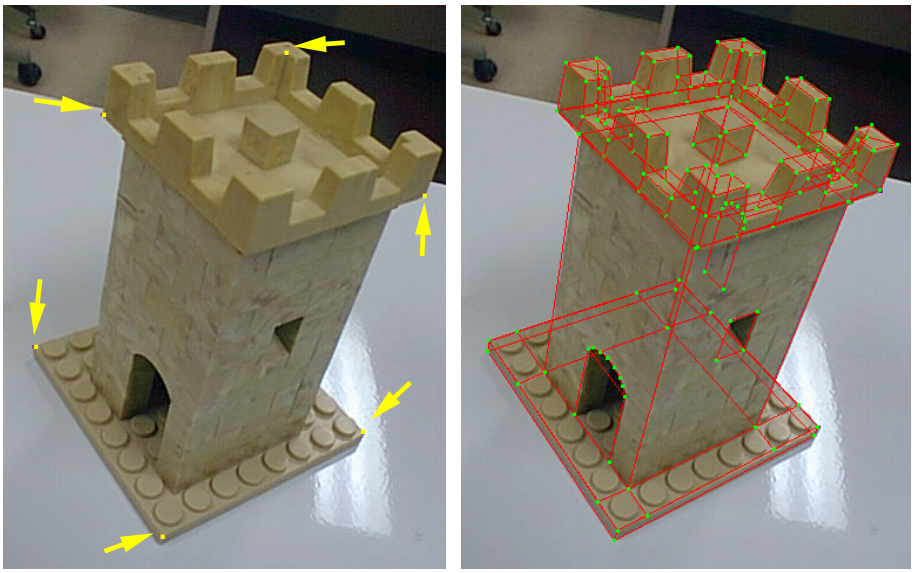


FIG. 5.2: Une image calibrée avec des points bien répartis.

sions dans son repère local plutôt que de devoir mesurer des primitives espacées dans le repère de la scène. De plus, il faut s'assurer que les primitives de calibration apparaissent dans au moins une autre image et de façon bien répartie. Evidemment, si on ne reconstruit qu'un seul objet, le problème est plus facilement soluble.

Ainsi, la *répartition*[1] image et spatiale des primitives de calibration a une grande influence sur le résultat de cette calibration. De plus, leur *configuration*[2] spatiale est aussi très importante. Il est connu que la calibration avec des points nécessite l'utilisation de points non coplanaires. Sur l'image de droite de la figure 5.3, on peut observer le résultat d'une calibration effectuée avec les six points coplanaires indiqués par des flèches dans l'image de gauche. Plusieurs autres configurations risquent de poser problème mais les caractériser représente un domaine de recherche encore actif aujourd'hui. Il faut donc retenir que si notre calibration ne nous satisfait pas, il peut être utile de vérifier la configuration spatiale des primitives employées.

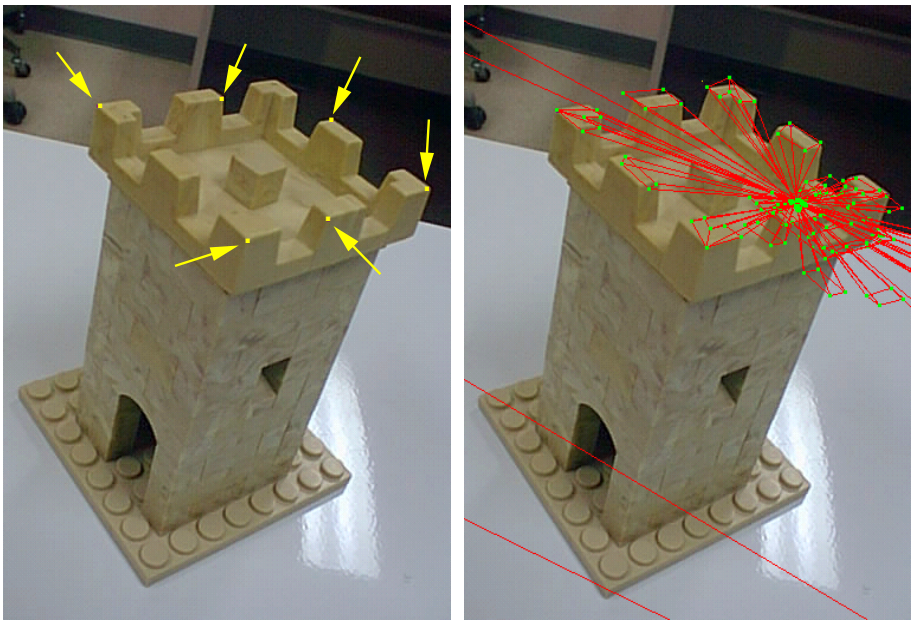


FIG. 5.3: Une image calibrée avec des points coplanaires.

[1]Par "répartition" des primitives, on entend leurs positions dans l'image ou dans l'espace selon le cas.
[2]Par "configuration" des primitives, on entend leur disposition les unes par rapport aux autres.

5.1.2 Remarques

Reprojection du modèle

Si l'on étudie la distance globale, en pixels, entre la reprojection du modèle 3D sur une image et les primitives 2D correspondantes, on peut remarquer que lorsque l'on active les contraintes 3D-3D afin d'améliorer le modèle, cette distance augmente dans la plupart des cas. En effet, lorsque les contraintes entrent en jeu, elles peuvent déformer le modèle pour corriger ses défauts afin qu'il soit plus conforme (visuellement parlant) à la réalité et donc plus susceptible d'être considéré satisfaisant par un observateur. Les systèmes d'équations servant à calculer la position des primitives ne contiendront plus seulement des contraintes de correspondance mais aussi les autres contraintes, chaque contrainte ayant le même poids. L'importance des contraintes de position en sera donc diminuée et la position calculée ne sera plus celle qui permettra au système de la reprojeter le plus exactement possible sur son équivalent tracé sur l'image, mais plutôt la position qui satisfera le mieux toutes les contraintes. Il est donc normal que la reprojection soit moins satisfaisante. De plus, la position de la primitive 2D n'est pas exacte alors qu'elle constitue un gros facteur dans le calcul de la distance d'"erreur". Les contraintes compensent donc pour l'imprécision de l'interaction de l'usager ainsi que pour celle affectant les primitives se trouvant loin des primitives de calibration fixes.

Par exemple, dans la scène de la tour, nous avons observé une distance maximale de 0.173 pixels avant l'activation des contraintes, avec une distance moyenne de 0.039 pixels entre le modèle reprojeté et les primitives tracées. Après activation des contraintes 3D-3D, la distance maximale passe à 0.166 pixels et la distance moyenne à 0.047 pixels.

Au sujet des primitives utilisées pour la calibration

En pratique, pour calibrer nos images, nous ne nous sommes servis que de points (incluant les sommets des polygones). Nous avons jugé que la position des lignes n'était pas toujours assez fiable pour risquer de dégrader notre calibration par leur utilisation, et par le même fait le modèle. De plus, les résultats obtenus avec seulement des points sont très satisfaisants. De même, d'autres types de contraintes (autres que celles de position) auraient pu être utilisées dans le calcul de la matrice. Pour des raisons de temps et de complexité d'implantation par rapport à l'incertitude de l'apport d'amélioration que nous aurions pu en retirer, nous avons préféré ne pas nous y attarder.

Il faut noter que l'imprécision de la primitive utilisée dans la calibration peut avoir une influence sur son résultat. Ainsi, si l'on place un point 2D sur l'image et que l'incertitude sur le calcul de sa profondeur est élevée (par exemple, si le point est caché ou s'il appartient à une primitive qui ne

reprojette que dans peu de pixels), la calibration utilisant ce point sera moins satisfaisante.

5.2 Reconstruction de géométrie

5.2.1 Scènes testées

Nous présentons dans cette section les résultats obtenus pour quelques scènes avec des statistiques sur leur reconstruction. Il est important de noter que nous utilisons une interface simple et que notre système n'est pas optimisé. Certains modèles ainsi que les images utilisées pour leur création sont disponibles sur la page web associée au projet (*http ://www.iro.umontreal.ca/labs/infographie/papers*).

Scène de synthèse

Nous avons utilisé une scène de synthèse simple afin de vérifier certains comportements du système puisque nous disposons alors des positions 3D exactes des primitives utilisées.

Cette scène contient sept objets cubiques formant un total de trente-trois polygones. Nous disposons de cinq images de résolution 500×400 de cette scène, prises des points de vue indiqués sur la figure 5.4 par les cônes gris. La configuration des points de vue a été estimée en utilisant les plans passant par tous les (ou un nombre suffisant de) points d'une même image (voir la figure 3.3 pour un rappel des deux plans passant par un point de l'image). La position de la caméra correspond approximativement à l'intersection de ces plans tandis que la direction de vue a été prise comme étant l'intersection des deux plans passant par le point central de l'image.

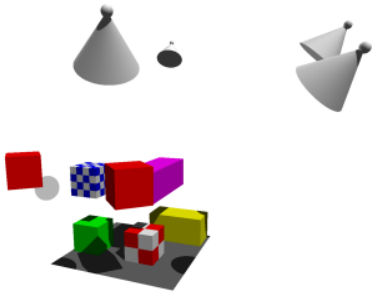


FIG. 5.4: Positions des cinq caméras ayant servi à obtenir les images.

Les coordonnées 3D de six points du cube en damier sur le sol ont été entrées pour calibrer une première image. Les autres images ont été calibrées par correspondances. La scène a pris environ une heure à reconstruire, avec les contraintes. Le modèle résultant est montré à la figure 5.5

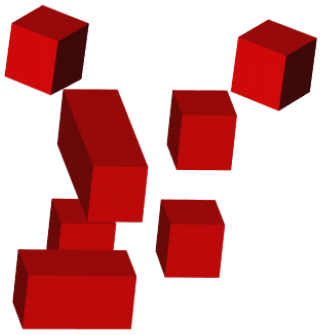


FIG. 5.5: Une vue synthétique du modèle reconstruit.

Les petits cubes

Cette petite scène a permis d'illustrer l'utilité des contraintes lorsque les images sont de moindre qualité. Nous disposons de quatre images de résolution 320×240 dont seulement une seule montre l'arrière des cubes. Nous pouvons observer ces images à la figure 5.6. La primitive de calibration est le cube dont la face supérieure affiche un "A". Trois des images (toutes sauf celle en bas à gauche) montrent des primitives 2D tracées par l'usager dont deux qui permettent d'observer la reprojection du modèle à travers la matrice de projection récupérée pour l'image (en haut à gauche et en bas à droite).

Dû à la résolution des images et à la taille du cube de calibration, les primitives plus éloignées du modèle reconstruit affichent des défauts évidents (image de gauche dans la figure 5.7) tels que des polygones non planaires, non parallèles ou non perpendiculaires. De plus, plusieurs cubes sont incomplets car un ou plusieurs des sommets de leurs faces arrières n'étaient visibles que sur une seule image et donc n'avaient pu être mis en correspondance. Ces défauts ont été corrigés par l'ajout de contraintes entre les diverses primitives 3D. Ces contraintes ont même permis d'obtenir la position des primitives visibles dans une seule image comme on peut le voir sur les images du centre et de droite à la figure 5.7 (par exemple, le cube en bas à gauche de la scène).

La figure 5.8 montre une nouvelle vue synthétique du modèle reconstruit des petits cubes. Les textures ont été extraites des images et apposées sur le modèle.

Les autres modèles

Quelques-uns des modèles reconstruits incluent une tour jouet de cent dix polygones reconstruite à partir de dix images en six heures (figure 5.9), une scène de bureau de trente polygones reconstruite à partir de six images en trois heures (figure 5.10) et une cafetière d'une centaine de polygones

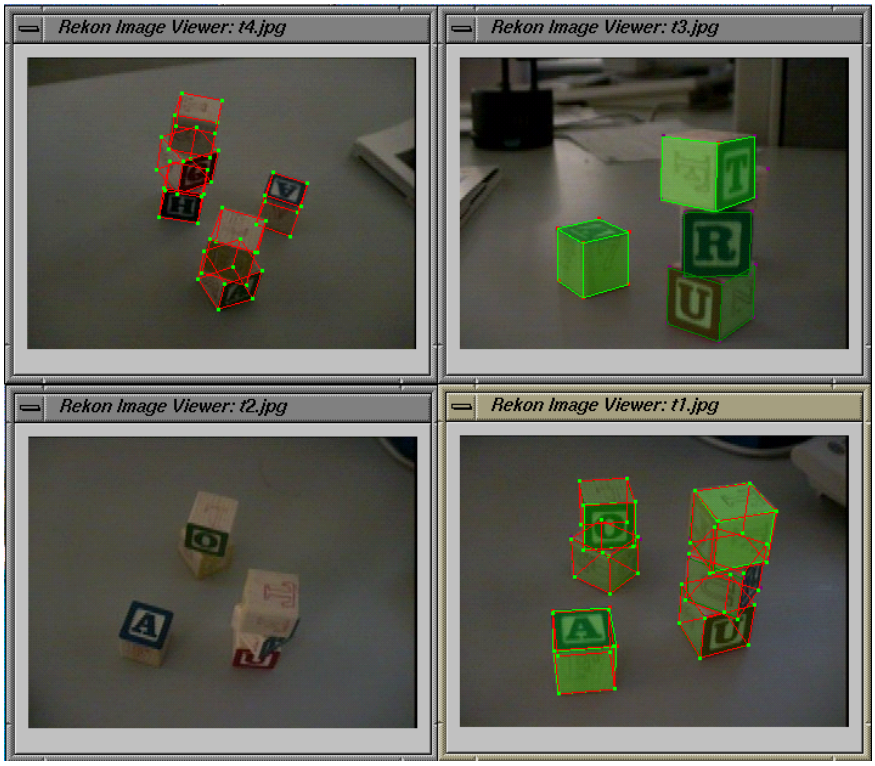


FIG. 5.6: Les quatre images des petits cubes, avec des primitives tracées ou reprojetées. L'image en haut à gauche et celle en bas à droite permettent d'observer la reprojection du modèle. Toutes excepté celle en bas à gauche montrent les primitives 2D tracées par l'usager.

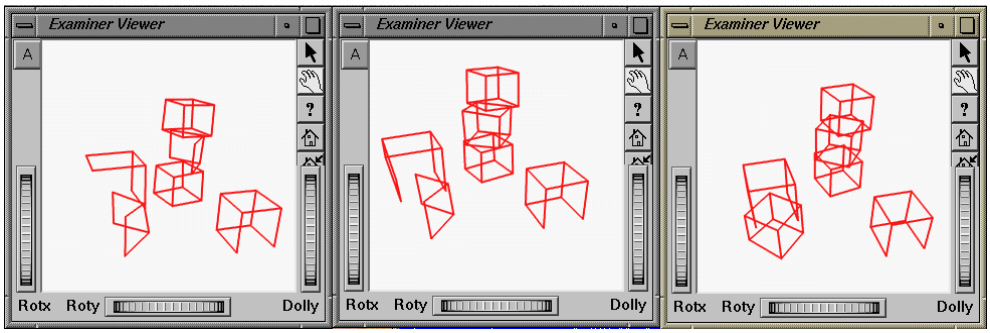


FIG. 5.7: Amélioration progressive de la reconstruction des petits cubes. Les contraintes 3D ont permis d'obtenir la position de primitives qui n'avaient pu être calculées avec les contraintes de correspondance uniquement, par exemple les cubes dans le coin inférieur gauche.

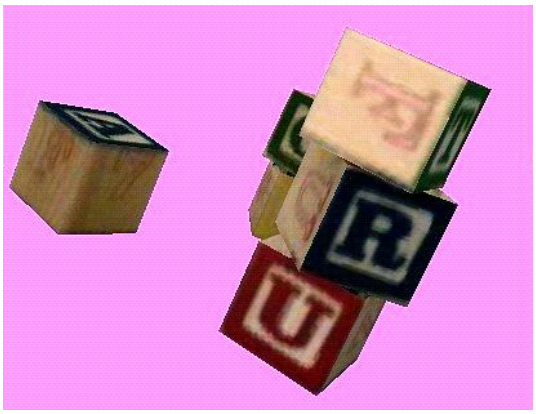


FIG. 5.8: Une vue synthétique du modèle reconstruit des petits cubes, avec des textures également extraites des images.

reconstruite à partir de neuf images en cinq heures (figure 5.11). Chacune des figures comprend une des images employées dans la reconstruction (avec le modèle en reprojection par-dessus) ainsi qu'une ou plusieurs images générées de points de vue synthétiques arbitraires à l'aide du modèle reconstruit.

5.2.2 Reconstruction des primitives

Nous avons pu observer que la reconstruction satisfaisante d'une primitive dépend de quelques facteurs importants :

- le choix des points de vue utilisés pour reconstruire cette primitive ;
- le niveau de complexité (nombre de primitives et nombre de contraintes) de la scène ;
- la qualité du tracé de la primitive 2D sur l'image qui dépend elle-même de l'usager, de la résolution des images et de la visibilité de la primitive (cachée ou qui projette dans un petit nombre de pixels par rapport à sa surface réelle) ;
- l'angle duquel la primitive est observée.

Choix des points de vue

Il est connu que la stéréo employant des caméras espacées produit des résultats beaucoup plus stables et précis que lorsque les caméras sont proches. Ceci est dû au fait que les rayons que l'on intersecte pour obtenir les positions des points ne sont pas parallèles et la profondeur calculée est ainsi beaucoup plus précise que lorsque les directions des rayons sont presque parallèles. La figure 5.12 permet de comprendre pourquoi l'imprécision sur la profondeur est plus élevée lorsque les points de vue sont rapprochés. En effet, sur les deux figures présentées, l'incertitude de la position du point 2D est

FIG. 5.9: Une image d'une tour jouet et trois images générées de points de vue arbitraires à partir de sa reconstruction. On a reprojeté le modèle obtenu sur l'image originale avec la matrice de transformation calculée pour cette image. Les textures ont également été extraites des photos.

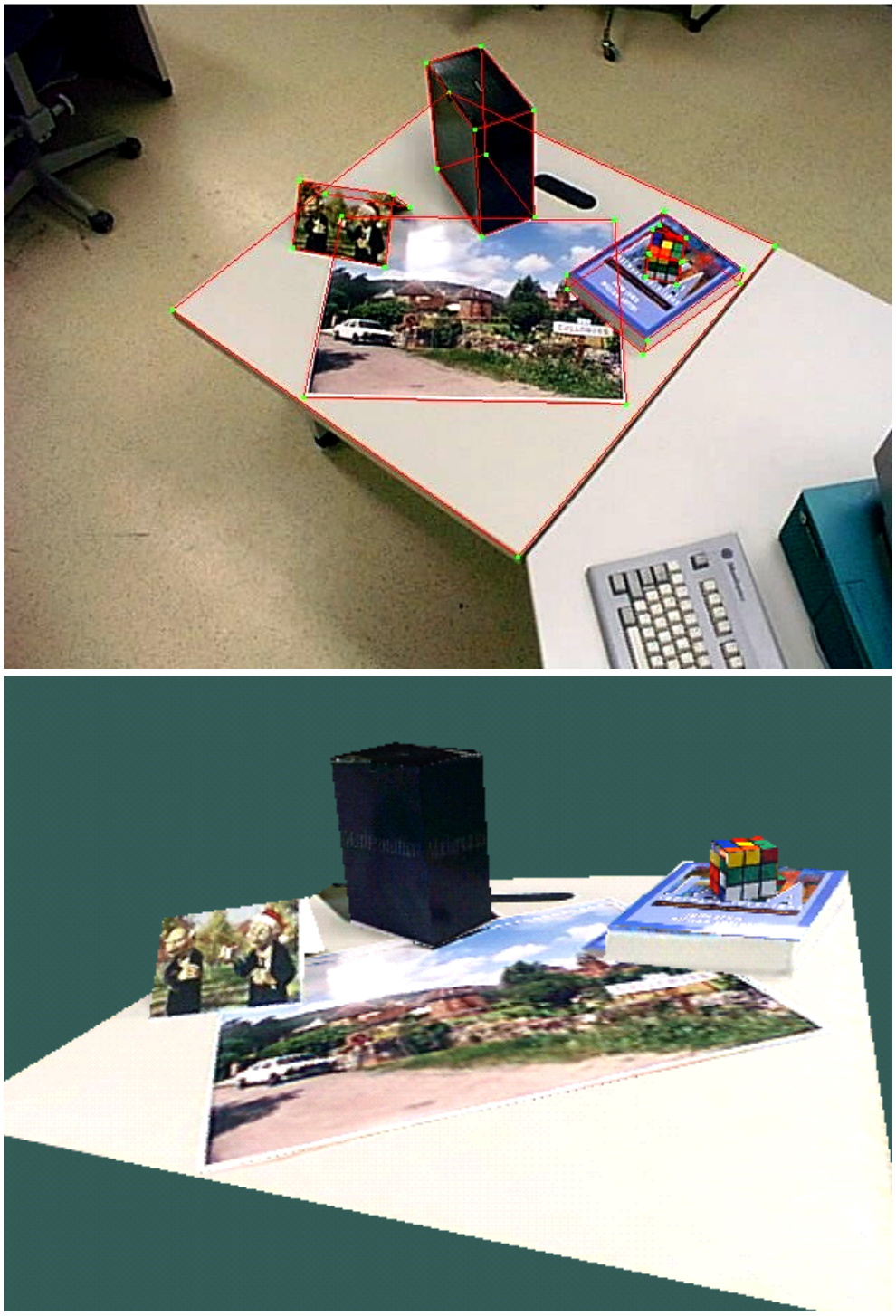
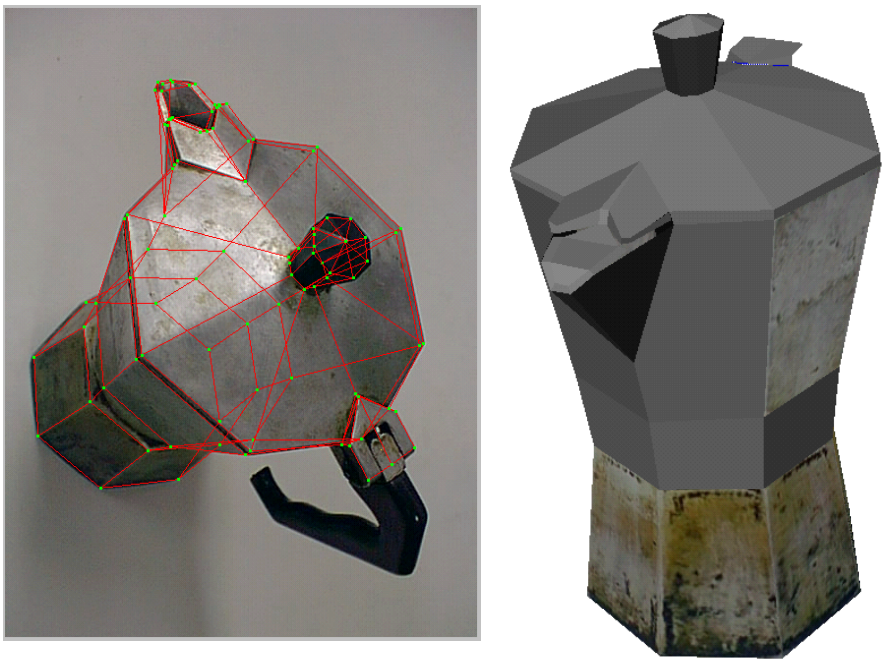


FIG. 5.10: Une image d'un bureau et une vue synthétique du modèle généré, avec les textures extraites des images originales.



(a) (b)

FIG. 5.11: (a) Une des images ayant servi à reconstruire la cafetière avec le modèle reprojeté. (b) Une vue du modèle reconstruit avec quelques textures extraites des images.

la même, tel qu'indiqué. La zone grisée représente la région 3D dans laquelle peut se trouver le point. On observe que cette région est bien plus étendue lorsque les points de vue sont rapprochés (image (a)) que lorsqu'ils sont éloignés (image (b)). Ainsi, *Rekon* produit de bien meilleurs résultats lorsque les points de vue utilisés sont très différents. Par le même fait, le processus d'acquisition d'images en est simplifié car nous avons moins de contraintes à respecter. Moins d'images sont nécessaires ce qui permet de traiter plus facilement des environnements plus étendus. Ainsi, les primitives reconstruites à partir de points de vue éloignés sont mieux reconstruites que celles reconstruites uniquement à partir de points de vue rapprochés.

La figure 5.13 montre deux images des petits cubes avec une primitive 2D ligne tracée dans chacune. Le cube comportant un "A" est celui qui a servi à la calibration des images et il n'y a aucune autre primitive dans la scène. Les deux images ont été prises de points de vue rapprochés. La figure 5.14 montre le résultat de reconstruction obtenu après la mise en correspondance des deux lignes. La position de la ligne n'est pas très satisfaisante. Pour l'exemple de la figure 5.15, on a remplacé une des images par une autre prise d'un point de vue opposé à la première. Le résultat montré à la figure 5.16 permet de voir nettement l'amélioration obtenue.

Par contre, l'ajout d'une contrainte (dans ce cas précis) à la scène de la figure 5.13 permet de corriger aussitôt la position de la ligne. La figure 5.17 montre le résultat obtenu après l'ajout

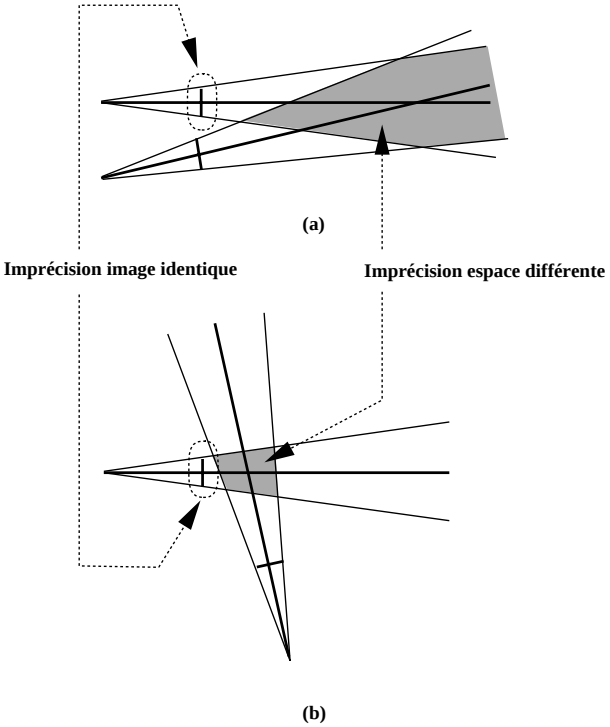


FIG. 5.12: Effet du choix des points de vue. (a) Deux points de vue rapprochés et l'imprécision en espace 3D résultant pour le calcul de la position d'un point. (b) Deux points de vue éloignés et l'imprécision en espace 3D résultant pour le calcul du même point.

d'une contrainte de coplanarité entre la ligne et la face supérieure du cube de calibration. D'autres contraintes auraient pu permettre d'arriver au même résultat. Évidemment, les améliorations obtenues dépendent beaucoup du modèle, des scènes 2D et des caméras disponibles, du nombre de contraintes dans le système, etc. On ne peut énoncer de règle générale en ce qui concerne l'amélioration potentielle d'un modèle étant donné le nombre de facteurs entrant en jeu dans sa reconstruction. Tout au plus avons nous constaté que, dans la plupart des cas, les contraintes semblent compenser pour la restriction de position des points de vue.

Sur la figure 5.18, on peut voir deux lignes tracées sur deux des images de la scène de synthèse, prises de points de vue rapprochés. Une vue de la ligne reconstruite se trouve à la figure 5.19, sur laquelle est également indiquée la position de la ligne "théorique". On peut ainsi observer l'erreur sur la position de la ligne reconstruite. Sur la figure 5.20, on a tracé les deux lignes sur deux images prises de points de vue très différents. La position calculée est beaucoup plus correcte, comme on peut le voir sur la figure 5.21.

Cet exemple permet de constater que même si la scène comporte plus de primitives et de contraintes et que les matrices de projection sont bien calculées, l'influence des points de vue se fait encore sentir. L'exemple de la figure 5.22 montre la reconstruction correcte d'une ligne obtenue après

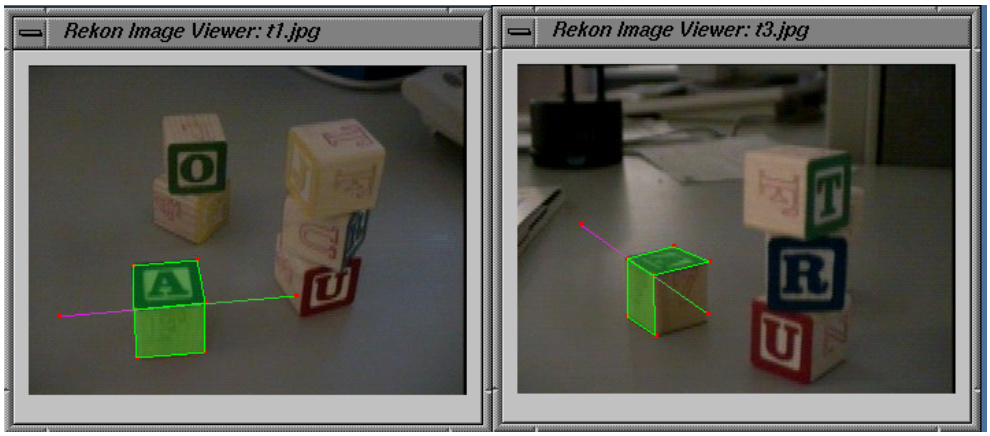


FIG. 5.13: Deux images provenant de points de vue proches, sur lesquelles on a tracé une ligne 2D.

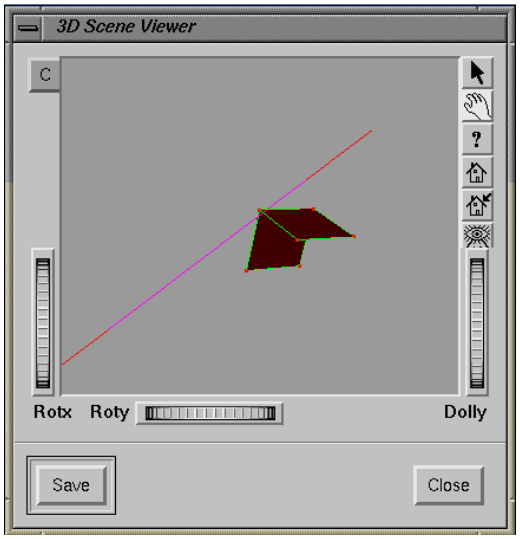


FIG. 5.14: La ligne 3D reconstruite à l'aide d'une contrainte de correspondance entre les lignes 2D des images de la figure 5.13. Sa position est incorrecte.

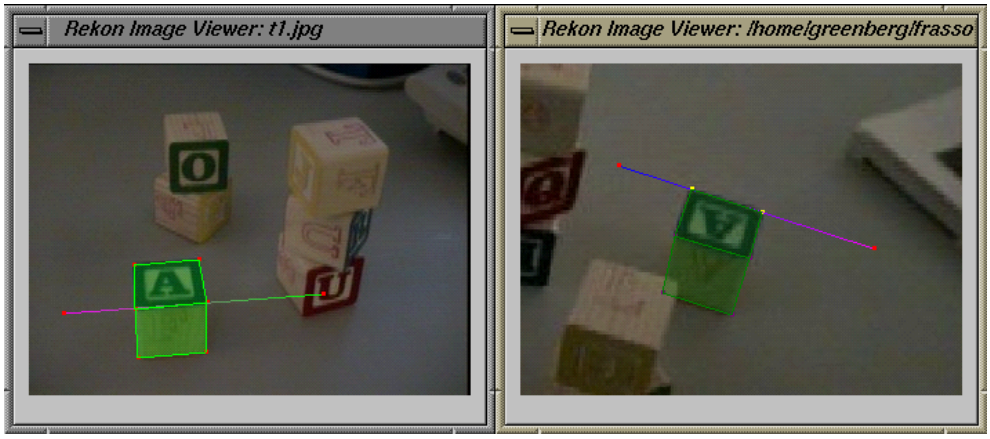


FIG. 5.15: Deux images prises de points de vue éloignés, sur lesquelles on a tracé une ligne 2D.

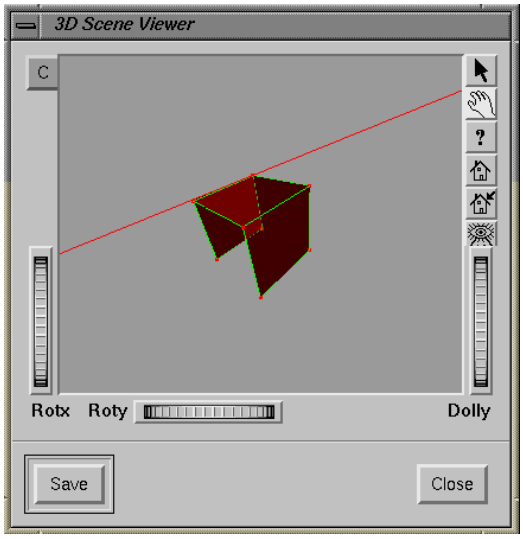


FIG. 5.16: La ligne 3D reconstruite à l'aide d'une contrainte de correspondance entre les lignes 2D des images de la figure 5.15. La position de la ligne est bien meilleure qu'à la figure 5.14.

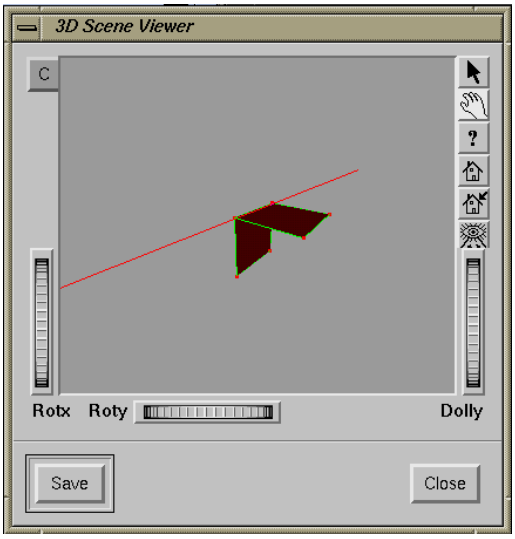


FIG. 5.17: La ligne 3D de la figure 5.14 corrigée après l'ajout d'une contrainte de coplanarité entre la ligne et la face supérieure du cube.

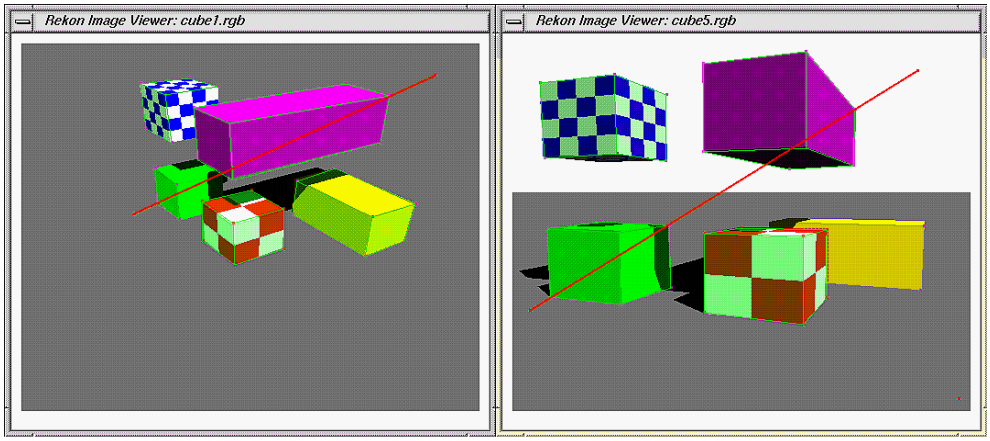


FIG. 5.18: Deux images provenant de points de vue proches, sur lesquelles on a tracé une ligne 2D.

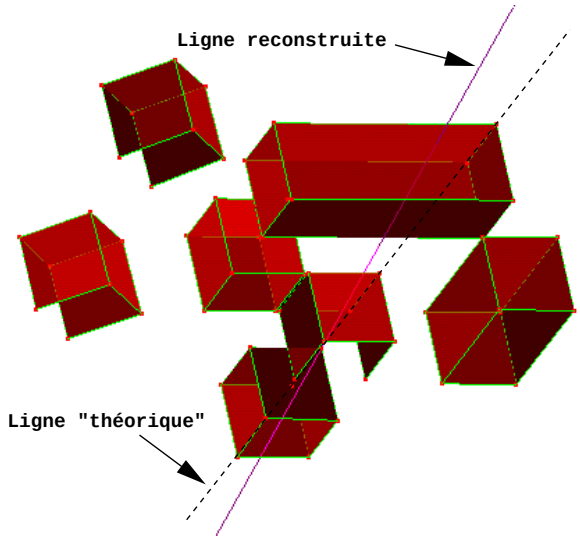


FIG. 5.19: La ligne 3D reconstruite avec les primitives de la figure 5.18.

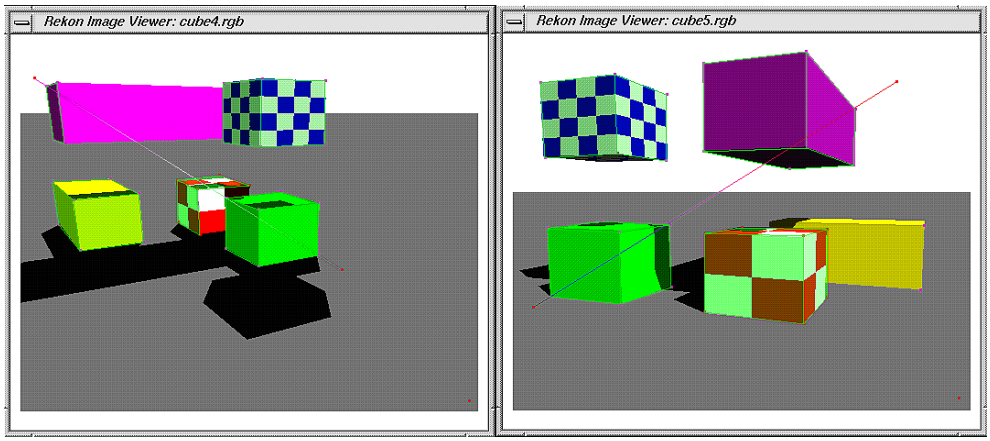


FIG. 5.20: Deux images prises de points de vue éloignés, sur lesquelles on a tracé une ligne 2D.

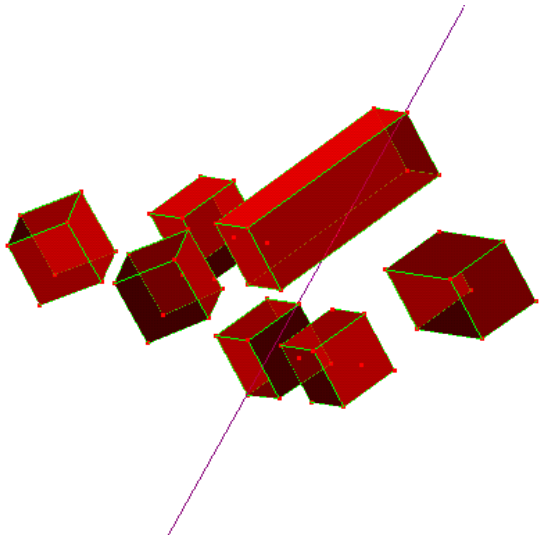


FIG. 5.21: La ligne 3D reconstruite avec les primitives de la figure 5.20.

une seule itération sur un modèle complexe. Ainsi, plus le système est stable, plus la reconstruction des primitives sera rapide à converger.

Il est intéressant de remarquer que pour les exemples pour lesquels la position 3D de la ligne était incorrecte, la reprojection de la ligne à travers la matrice de projection calculée correspondait très bien à la primitive tracée.

Après avoir effectué les mêmes tests pour un point, nous avons pu nous rendre compte que la reconstruction des points semble moins dépendre des points de vue que les lignes. Ceux-ci sont souvent plus contraints que les lignes dû à leur appartenance à un polygone, lui-même éventuellement soumis à d'autres contraintes comme la planarité de ses sommets, le parallélisme, la perpendicularité, etc. De plus, lorsque nous reconstruisons un point, nous calculons ses trois coordonnées alors que pour une ligne, ce sont les six coordonnées de la matrice de Plücker de la ligne que nous déterminons. Ces six coordonnées n'ont peut-être pas toutes la même influence dans la reconstruction de la ligne et il serait intéressant d'étudier cet aspect plus en détail.

Reconstruction des objets incomplets

Les contraintes de normale, de planarité de polygone et de coplanarité permettent de reconstruire la position de primitives sans que celles-ci aient été mises en correspondance. C'est un avantage très intéressant car cela diminue le nombre de primitives à tracer et le nombre de vues nécessaires à la reconstruction d'un modèle.

En effet, il faut trois plans pour trouver la position 3D d'un point. Sa contrainte de position

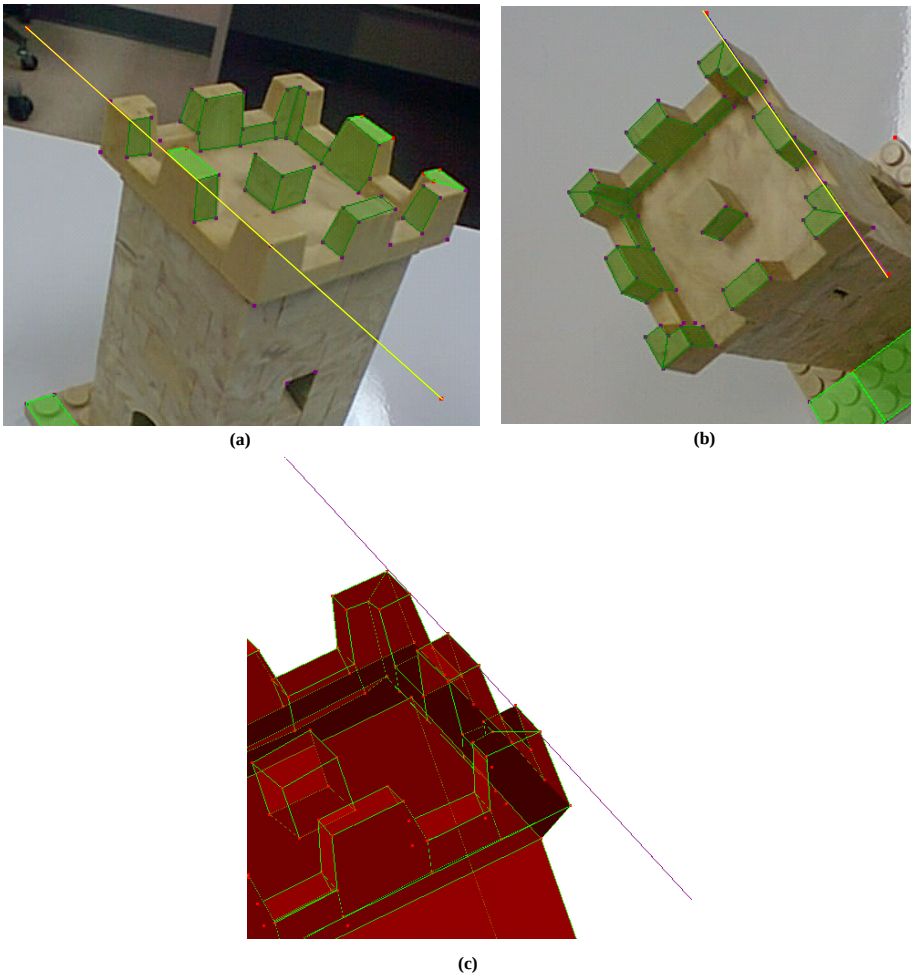


FIG. 5.22: (a) et (b) : Deux primitives lignes mises en correspondance sur un modèle complexe. (c) La primitive 3D résultante après une seule itération.

en fournit deux. Les quatre types de contraintes mentionnées ci-haut permettent d'ajouter le plan manquant au calcul.

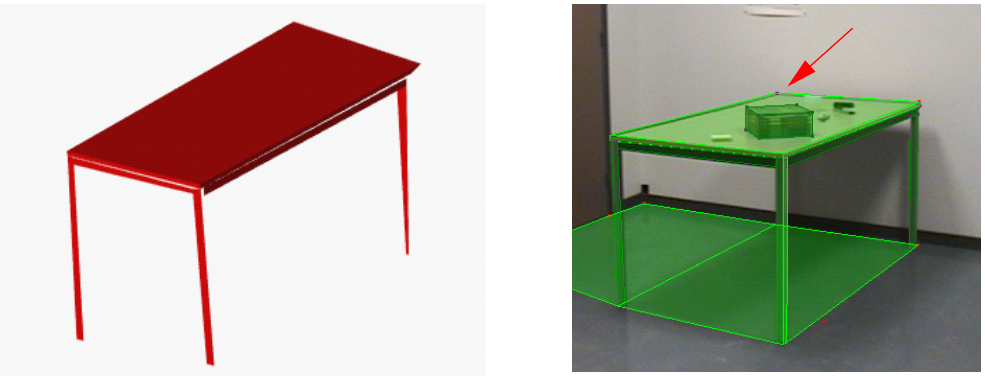
De même pour les lignes, une contrainte de position de ligne génère un plan. Il n'en manque qu'un pour pouvoir déterminer la position de la ligne dans l'espace. Une contrainte impliquant la ligne fournit un tel plan.

Imprécision au niveau des primitives 2D

Il est important de noter que la reconstruction de primitives vues à un angle rasant est beaucoup moins précise que si les primitives sont vues de face. Les coordonnées dans le repère de l'image sont obtenues en indiquant les points à l'écran. La précision des mesures est donc au pixel près et même si on peut agrandir (*zoomer*) arbitrairement la zone pointée, l'information du pixel (sa couleur) est indivisible et il devient difficile d'estimer exactement la position des segments et des coins à l'intérieur d'un pixel. Cela peut beaucoup faire varier la reconstruction d'un point si celui-ci est éloigné dans l'image et qu'une grande partie d'un polygone projette dans ce pixel. L'aire (en pixels) couverte par la projection d'un objet (angle solide entre une surface et sa reprojection) a donc une grande importance. Des variations de l'ordre du pixel dans le placement de la primitive peuvent avoir une grande influence sur sa reconstruction dans l'espace si cette projection ne recouvre que quelques pixels. Il devrait être possible de déterminer des mesures d'évaluation qui prendraient en compte des mesures comme la surface couverte par la projection d'un objet, etc. afin de pouvoir donner un poids approprié à la contrainte correspondante.

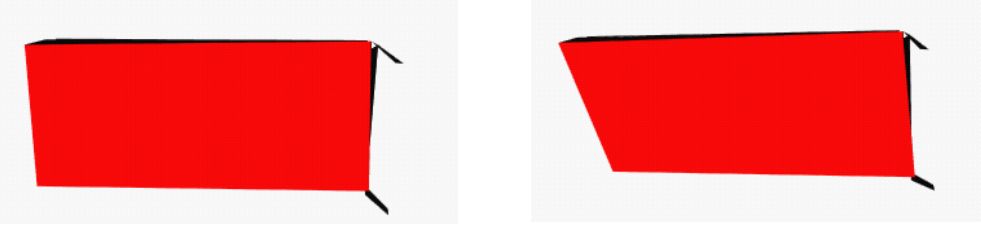
Sur la figure 5.23, on peut voir l'image (à droite) d'une table dont le dessus est vu d'un angle rasant. Sa reconstruction est présentée à gauche. Sur la figure 5.24 (a), on peut voir la table du dessus. Le point indiqué par une flèche dans l'image 5.23 (b) est celui qui est le plus susceptible d'être affecté par une imprécision au niveau du tracé des primitives. Nous avons déplacé ce point 2D de 4 pixels vers la gauche et de 1 pixel vers le haut. Le résultat obtenu dans la reconstruction après quelques itérations est présenté sur l'image 5.24 (b). On voit que la position de la primitive a été assez affectée. Le même test effectué avec un sommet plus proche de l'observateur n'a permis d'observer qu'un très léger déplacement de la primitive 3D correspondant au sommet déplacé.

Nous pouvons estimer la position d'une primitive cachée (comme le sommet d'un polygone, par exemple) en suivant les deux arêtes visibles correspondantes. Cela est très pratique car nous pouvons ainsi tracer plus de primitives que celles effectivement visibles mais nous nous exposons ainsi au risque de commettre une erreur d'imprécision plus importante sur cette primitive que sur les autres.



(a) (b)

FIG. 5.23: (a) Reconstruction d'une table (b) Une des images originales, avec le modèle reprojeté.



(a) (b)

FIG. 5.24: (a) Table vue de haut. (b) Résultat après un léger déplacement d'un sommet du polygone 2D.

Ce risque est diminué lorsque l'on soumet la primitive à des contraintes supplémentaires.

5.3 Convergence

Rappelons qu'une *itération de convergence* est le processus qui calcule les matrices de projection à l'aide des positions 3D valides puis calcule de nouvelles positions à l'aide des matrices ainsi disponibles.

Pour toutes les scènes testées, chaque itération a pris moins d'une seconde sans les contraintes et moins de deux secondes avec les contraintes activées. Le tableau 5.1 permet de voir le temps d'une itération pour diverses scènes, avec et sans les différentes contraintes. Le nombre de points et de polygones de chaque scène est indiqué ainsi que le nombre de primitives impliquées pour chaque type de contrainte, si cela s'applique.

Afin d'évaluer la précision et la convergence de notre système, nous avons utilisé la scène synthétique décrite au début de la section 5.2.1. Nous avons suivi la variation de position 3D des trois points indiqués sur la figure 5.25 (a) de coordonnées respectives (-2,3,0), (0,2,-2) et (1,2,-2). Les trois courbes de la figure 5.25 (b) représentent la distance en coordonnées du monde entre la position

contraintes/scène	cubes	synthèse	bureau	tour	cafetière
nb points/polygones	47/24	57/72	38/29	169/110	97/94
sans	0.06	0.16	0.07	0.71	0.27
+normale	0.08	0.25	0.10	0.93	0.38
+planarité	0.14	0.48	0.19	1.42	0.80
+parallélisme (nb)	0.21 (10)	0.50 (2)	0.21 (5)	1.43 (0)	0.82 (0)
+perpendicularité (nb)	0.54 (77)	0.61 (0)	0.39 (53)	1.43 (0)	0.82 (0)
+coplanarité(pts/polys)	0.68 (23/2)	0.87 (24/0)	0.50 (11/2)	1.89 (10/20)	0.84 (0/2)

TAB. 5.1: Durée, en secondes, du calcul d'une itération pour diverses scènes avec ajout progressif des différentes sortes de contraintes.

réelle des trois points et celle des points reconstruits.

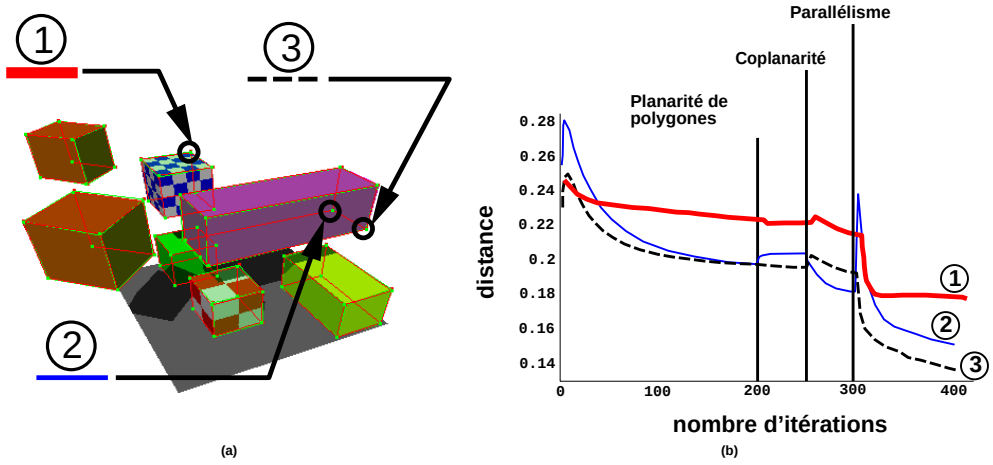


FIG. 5.25: (a) Trois points de la scène synthétique, avec le modèle reprojeté. (b) Distance entre la position réelle et calculée de trois points de la scène.

Nous avons d'abord effectué deux cent itérations en trente secondes sans aucune contrainte autre que les correspondances entre points. Nous avons ensuite activé une à une les contraintes de planarité de polygone, de coplanarité, puis de parallélisme. Cinquante itérations séparent l'introduction de chaque type de contrainte.

Nous pouvons observer que les trois courbes montrent une diminution globale de la distance, et donc une amélioration de la reconstruction. Chaque décroissance de la courbe est suivie d'une stabilisation. Ce plateau indique que le système a atteint un équilibre entre le calcul des caméras et celui des positions 3D et qu'aucune des deux étapes n'apporte de modification significative à l'autre. L'introduction des contraintes permet d'améliorer la reconstruction, en particulier la contrainte de parallélisme dans cette scène où c'est une caractéristique prépondérante. En effet, quelques itérations après l'ajout de la contrainte, on observe une chute importante des trois courbes dès que la contrainte est activée. Nous pouvons remarquer que l'introduction d'une contrainte peut parfois causer une brève augmentation de la distance entre le point réel et son équivalent reconstruit. Ceci est dû au fait que,

parfois, l'ajout d'une nouvelle contrainte peut perturber le système qui avait trouvé son équilibre et cette perturbation affecte en premier les éléments les moins contraints (comme le deuxième point par exemple). Dans la plupart des cas observés, le système retourne rapidement à un état stable et amélioré.

Tel que mentionné plus tôt, la convergence du système n'a pas été prouvée, mais elle a été constatée pour tous nos tests. De plus, les modèles semblent converger assez rapidement. Par contre, en cas d'erreur de l'usager (par exemple une mauvaise correspondance, un polygone tracé à l'envers ou une erreur dans l'application d'une contrainte), le système peut *diverger*. En effet, l'erreur entraîne le calcul de mauvaises matrices de projections qui, à leur tour, entraînent le calcul de mauvaises positions et ainsi de suite. Pour un modèle peu contraint ou assez simple, l'effet de l'erreur sera très rapidement détecté par l'usager car le modèle se dégradera très vite. Par contre, si le modèle est très contraint, l'effet de l'erreur sera subtil et souvent pas immédiatement perçu par l'usager. Celui-ci remarquera que la reconstruction ne semble pas s'améliorer et il devra recommencer sa reconstruction dans la plupart des cas, ne sachant quand ni où cette erreur s'est introduite.

5.4 Robustesse

Notre système semble relativement robuste, essentiellement dû à l'utilisation des contraintes. Les modèles convergent rapidement vers un résultat satisfaisant et l'effet de l'introduction des contraintes est la plupart du temps aussitôt visible au niveau des primitives concernées. Cependant, si le modèle est complexe et déjà très contraint, l'effet de la contrainte prendra un peu plus de temps (d'itérations) à se faire sentir. De plus, peu d'images suffisent pour obtenir des modèles corrects, et ce même si la qualité des images n'est pas très bonne.

On a pu observer que lorsque l'on commet une erreur au niveau de la position 2D d'un point dans une image, le système reste assez stable. Si la reconstruction est relativement simple et donc le système est peu contraint, le modèle reflètera cette erreur de manière évidente. Par contre, si le modèle est plus complexe et le système doit manipuler beaucoup d'équations, l'erreur sera "absorbée" par les autres contraintes et ne sera presque pas visible. L'ajout de contraintes sur un modèle géométrique simple permet de compenser largement pour les erreurs de placement de primitives, même lorsque celles-ci sont délibérément énormes. De plus, lorsque le modèle s'est déformé par l'introduction d'une telle erreur, l'activation des contraintes permet d'obtenir en quelques itérations un modèle plus satisfaisant. Nous avons introduit une erreur de placement de point (d'environ 50 pixels) sur une des images des petits cubes (figure 5.26). La figure 5.27 (a) montre une image de la scène reconstruite

avant l'erreur. Les figures 5.27 (b) et (c) permettent d'observer les répercussions de l'erreur lorsque les contraintes sont activées et lorsqu'elles ne le sont pas, une vingtaine d'itérations après l'introduction de l'erreur. Il est clair dans cet exemple que les contraintes compensent largement pour les erreurs d'imprécision de l'usager.

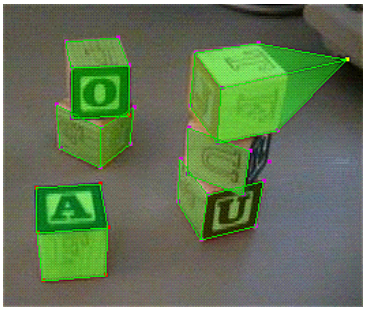


FIG. 5.26: Erreur de placement exagérée d'une primitive sur une image.

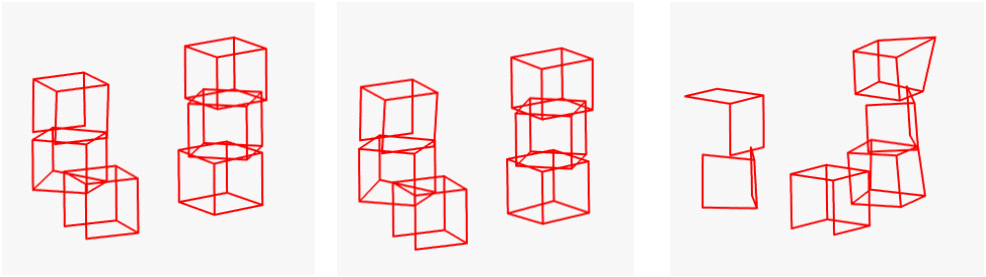


FIG. 5.27: Répercussions de l'erreur sur le modèle. (a) Modèle avant l'erreur. (b) Modèle après l'erreur, contraintes activées (c) Modèle après l'erreur, contraintes désactivées.

Une erreur de correspondance ou de contrainte, quant à elle, aura des répercussions plus importantes car c'est alors la sémantique du modèle qui est affectée. Le modèle se détériore alors plus ou moins vite selon sa complexité, et l'activation des contraintes ne pourra rien y faire, au contraire cela peut accélérer la dégradation du modèle.

Chapitre 6

Conclusion

“Knowledge is a rare thing –
you gain by giving it away.”

—Unknown

6.1 Contributions

Nous avons proposé une méthode de reconstruction interactive et incrémentale, comprenant une technique de calibration adaptée aux besoins de cette reconstruction, à partir de différentes vues d’une scène réelle ou synthétique. La récupération des caméras et de la géométrie s’effectue grâce à un ensemble de primitives 2D tracées par un usager sur des images et reliées par divers types de contraintes à l’aide d’un système simple à utiliser.

Tel que le fait remarquer Carlsson [Car95], le problème de la reconstruction et celui de la calibration sont intimement liés et ne devraient pas être résolus en deux étapes séparées. Dans notre système, chacune de ces deux étapes influe énormément sur l’autre. En itérant entre elles, nous pouvons obtenir des résultats satisfaisants sur les deux plans.

La méthode de calibration que nous employons ne nous donne pas les paramètres de la (ou des) caméra qui a permis d’obtenir les images mais seulement une matrice de transformation qui approxime cette caméra. Nous ne supposons aucun modèle de caméra en particulier lors du calcul des coefficients de cette matrice, évitant ainsi l’extraction de paramètres usuels de caméras. En effet, c’est un processus possiblement instable selon les expériences passées et que nous n’avons pas tenté d’aborder dans le cadre de cette recherche. L’utilisation de ces matrices pour reprojeter le modèle reconstruit sur les images correspondantes a permis de montrer que notre méthode semble efficace pour approximer les caméras réelles. Une condition importante à respecter pour obtenir une approximation

de caméra satisfaisante pour une image est que les primitives utilisées soient bien réparties à travers l'image et si possible dans l'espace. De plus, leur configuration spatiale est également influente (par exemple, on ne peut calibrer une image avec des points coplanaires).

La récupération de géométrie s'effectue en utilisant les correspondances que l'usager indique entre les primitives 2D qu'il a lui-même tracées sur les images. Il peut observer le modèle 3D obtenu et juger de sa qualité en le reprojetant sur les images originales ou en utilisant un "visualisateur de scènes" qui lui permet de l'examiner dans n'importe quelle position, taille et orientation. S'il remarque des défauts (planarité, parallélisme ou perpendicularité non respectés, objets flottants, primitives manquantes, etc.), il peut tenter de les corriger en ajoutant de nouvelles images, d'autres primitives 2D ou des contraintes entre les primitives 3D concernées. Pour cela, il n'a qu'à sélectionner leurs homologues 2D sur les images. La nette amélioration observée sur les modèles après l'addition de contraintes justifie largement leur emploi. L'utilisation de ces contraintes permet d'obtenir des modèles au moins équivalents à ceux obtenus avec les systèmes actuels. Il est important de noter que les défauts que nous corrigeons avec l'ajout des contraintes représentent des anomalies, souvent sémantiques, que les observateurs remarquent très vite et qui nuisent fortement à la qualité du modèle. Il est donc important de les rectifier afin d'obtenir des modèles plus réalistes.

Les diverses contraintes utilisées pour le calcul des deux étapes de reconstruction sont exclusivement exprimées sous forme d'équations linéaires. Elles découlent de principes simples des géométries affine et projective. L'algorithme de résolution utilisé (la décomposition en valeurs singulières) pour les systèmes de contraintes générés permet d'obtenir la "meilleure" solution au sens des moindres carrés. C'est un algorithme efficace, fiable et robuste qui nous permet d'obtenir une bonne stabilité du système. Une itération de convergence (calcul des caméras suivi de celui de la géométrie) prend un temps négligeable par rapport au temps de modélisation. Typiquement, on n'a pas observé plus de deux secondes par itération, incluant tout le calcul des contraintes. De plus, il n'est pas nécessaire d'itérer longuement pour obtenir des résultats satisfaisants. Nous avons pu observer que le système converge rapidement vers un état stable.

L'usager tient une place prépondérante au sein de notre système et il est intégré à tous les niveaux du processus. C'est grâce à lui que tout est possible. Il trace les primitives 2D, indique les coordonnées 3D des primitives de calibration, établit les correspondances et les autres contraintes entre les primitives, lance le calcul des caméras ou du modèle, etc. Son intégration dans le système lui permet d'avoir un bon contrôle sur tout le processus de reconstruction et, considérant la qualité graphique désirée des modèles à reconstruire, d'obtenir ainsi de meilleurs résultats. Le prix à payer

est un temps d'interaction avec le système parfois non négligeable.

Nous avons voulu utiliser l'usager pour parer aux erreurs typiques des algorithmes automatiques provenant du domaine de la vision tels que la segmentation, la mise en correspondance, le choix des primitives à extraire, etc. L'usager *sait* ce qu'il veut modéliser lors d'une reconstruction et avec quelle précision il aimerait le faire. Il peut distinguer facilement les objets qui l'intéressent dans des images. Il juge également quand un modèle reconstruit le satisfait selon l'utilisation qu'il veut en faire, ou qu'une texture serait suffisante pour simuler les détails manquants, et peut donc décider à tout moment de continuer la convergence du système, d'ajouter des contraintes ou d'arrêter le tout. Il est responsable de beaucoup d'aspects, mais a également le contrôle sur beaucoup d'aspects.

Notre approche sépare le travail de reconstruction en deux parties. Une plus facile (et moins risquée !) pour un utilisateur, c'est-à-dire la détermination des contours des projections du modèle dans les images, les correspondances entre ces contours ainsi que diverses relations géométriques au sein du modèle lui-même. Il n'a pas besoin de travailler en 3D puisque toute son interaction s'effectue sur les images. La deuxième partie est plus facile pour un ordinateur. Il s'agit de la résolution des systèmes d'équations générées par les diverses contraintes introduites par l'utilisateur. Même si l'interaction peut sembler fastidieuse (surtout si l'on commet des erreurs qui peuvent s'avérer fatales), la qualité graphique des modèles obtenus et le fait que cette interaction puisse être réduite de plusieurs façons, comme cela sera vu plus loin, compensent.

Un avantage intéressant est que l'on puisse utiliser une grande diversité d'images de qualité variable. Les résultats obtenus avec des images de qualité moyenne sont tout de même très satisfaisants. De plus, nous ne sommes pas contraints à des déplacements de caméras réduits comme plusieurs autres systèmes qui effectuent les correspondances automatiquement. La récupération de la structure 3D d'une scène est très sensible au bruit dans les images lorsque le déplacement entre les positions de caméras disponibles est petit et nous évitons donc ce problème. De plus, la récupération de la profondeur est beaucoup plus précise lorsque les images proviennent de points de vue éloignés. Nous sommes ainsi en mesure de traiter de plus grands environnements avec moins d'images que les autres systèmes.

Mais . . .

La qualité de la reconstruction et de la calibration dépend en partie de la précision des primitives 2D tracées par l'usager. L'augmentation de cette précision par l'utilisation d'images de plus haute résolution ou d'outils d'aide entraînera une amélioration certaine des résultats. La présence de

contraintes compense beaucoup cette dépendance à la précision.

La nécessité d'avoir des mesures 3D ou de pouvoir les estimer (présence d'une primitive cubique par exemple) pour être en mesure de calibrer deux images peut constituer un inconvénient. Si ces mesures ne sont pas disponibles, nous avons pu observer qu'une estimation donnait quand même de bons résultats, les contraintes aidant. La reconstruction se fait alors à un facteur d'échelle près pour les dimensions estimées.

6.2 Améliorations

Les modèles que nous avons obtenus jusque là sont très satisfaisants. Par contre, nous nous sommes limités à la reconstruction de "mondes" polyédriques. Même si cela nous permet quand même de reconstruire un large éventail d'objets, il serait intéressant de pouvoir traiter les surfaces courbes. Borshukov [Bor97] présente une méthode pour intégrer la reconstruction de surfaces de révolution dans un système de reconstruction à partir d'images, en l'occurrence le système *Façade*. Les idées qu'il expose pourraient être intégrées dans notre système car les surfaces de révolution constituent quand même une grande partie des surfaces courbes (artificielles) qui nous entourent. Dans le cas de surfaces courbes arbitraires, plusieurs travaux ont été réalisés en vision mais les problèmes rencontrés sont souvent très importants et empêchent l'obtention de modèles satisfaisants pour des applications en synthèse d'images. Une adaptation de leurs méthodes pour notre système interactif pourrait éventuellement permettre d'éviter de tels problèmes.

Au niveau du système *Rekon* actuel, nous pensons qu'il bénéficierait grandement d'améliorations qui permettraient d'alléger la tâche de l'usager et de lui donner un meilleur contrôle sur tout le processus afin d'obtenir de meilleurs modèles. La première modification à considérer étant une amélioration de l'interface, pour l'instant pas aussi "conviviale" que souhaitée.

6.2.1 Réduction du temps d'interaction

L'inconvénient premier d'un système interactif de reconstruction est justement cette interactivité. Devoir dessiner chaque primitive à reconstruire dans deux images en moyenne, puis les mettre en correspondance une à une est une tâche qui peut s'avérer très fastidieuse lorsque le modèle est complexe. Il serait donc intéressant de pouvoir ajouter un certain degré d'automatisme au système. Il est toutefois essentiel que cet automatisme ne soit pas *risqué*, c'est-à-dire qu'il y ait peu de risques qu'une erreur survienne et que si cela se produit, l'usager puisse facilement y remédier, par exemple en annulant l'action fautive.

Outils d'aide à l'interaction

Le domaine de la vision a procuré beaucoup d'algorithmes intéressants en analyse d'images. Certains pourraient être intégrés directement dans *Rekon* ou adaptés pour y inclure une aide de l'usager afin de minimiser le risque d'erreur. Un outil de *segmentation assistée* permettrait d'accélérer le temps passé à dessiner les primitives. L'usager n'aurait qu'à spécifier grossièrement la position du contour de la primitive (par exemple, avec un outil du genre "pinceau" qui permettrait de tracer un large trait sur les contours de la primitive) et le système s'occuperait de placer avec le plus d'exactitude possible une primitive 2D sur ce contour. Une autre possibilité d'accélération serait de tracer automatiquement des primitives d'après leur reprojection sur une image. L'usager n'aurait alors qu'à modifier le placement de la primitive si besoin est et même ne plus avoir à spécifier les correspondances sous-jacentes. Cet outil pourrait également être combiné avec le précédent. De telles fonctionnalités pourraient permettre d'améliorer la précision du tracé et par le fait même la calibration et la reconstruction.

Au niveau de la mise en correspondance, le système pourrait indiquer la position des lignes épipolaires correspondant à un point d'une image dans les autres images. Cela réduit l'espace de recherche du problème puisque les points correspondants sur les autres images devraient se trouver sur cette ligne. C'est une des fonctionnalités du système *REALISE*. Lorsque l'usager place un point, les lignes épipolaires apparaissent sur les autres images et l'usager n'a qu'à faire glisser un point le long de la ligne à la bonne position. Cela suppose évidemment que les caméras sont bien calculées. Cela n'est vraiment utile que si un point est visible dans une image et pas dans une autre. On a ainsi moins de risque de se tromper en plaçant le point "à l'aveuglette".

Dans certaines situations, l'utilisation de mosaïques pourrait également réduire le temps de modélisation par la création automatique de correspondances lorsque l'usager trace des primitives sur la mosaïque (formée par la superposition des images disponibles). Cependant, leur construction est encore sujette à beaucoup trop de restrictions pour envisager sérieusement l'intégration des techniques actuellement proposées.

Récupération des détails

La récupération des petits détails entraîne un temps de modélisation non négligeable si on décide de ne pas les simuler par une texture. Dans *Façade*, le modèle calculé est utilisé pour guider un algorithme stéréo chargé de récupérer les détails (*model-based stereo*). De même, les progrès récents dans le domaine de la reconstruction stéréoscopique [RC98] permettraient de mettre en correspondance automatiquement plusieurs images prises sous des angles de vues arbitraires. Si on dispose d'une sur-

face polygonale approximative et d'un intervalle de tolérance fournis par l'usager, il serait possible de reconstruire une surface non polygonale se situant à l'intérieur de ce "volume de tolérance" par rapport à la surface approximative, sous forme d'une carte de déplacements (*displacement map*). On pourrait envisager l'intégration de telles techniques afin de récupérer les surfaces exactes à partir de surfaces approximatives.

Utilisation de primitives de plus haut niveau

L'ajout de primitives 2D ou 3D de plus haut niveau, telles des rectangles ou des cubes, serait une extension intéressante au système. De telles primitives seraient constituées de primitives de base ainsi que de contraintes. Par exemple, un cube est un ensemble de six polygones dont les arêtes ont une longueur fixe et reliés par des contraintes de parallélisme et de perpendicularité. L'usager n'aurait plus à spécifier toutes ces contraintes, la primitive les contiendrait implicitement. De telles primitives pourraient être organisées hiérarchiquement, comme les *blocs* de *Façade*. Debevec *et al.* [DTM96] démontrent que cette façon de faire est plus commode et plus efficace que de manipuler des primitives de bas niveau et que la modélisation à l'aide de blocs augmente la robustesse du système, tout en réduisant le nombre de paramètres à retrouver.

Le domaine de l'intelligence artificielle pourrait apporter des techniques afin d'accélérer le processus de modélisation par l'usager. Le système pourrait apprendre la structure générale de certains objets communs pour un certain type de scène, par exemple des étagères, des tables, des fenêtres, des portes, etc. Une étagère est composée d'un caisson normalement rectangulaire et d'un certain nombre de planches disposées horizontalement. L'usager pourrait seulement indiquer la position approximative du caisson et, à partir de ses connaissances, le système pourrait retrouver la position des planches, y dessiner les primitives nécessaires et ainsi faciliter la tâche de l'usager. On introduirait ainsi des primitives "intelligentes" de haut niveau.

6.2.2 Augmentation du contrôle

Plusieurs aspects du système pourraient être améliorés de façon à fournir à l'usager un meilleur contrôle durant tout le processus de reconstruction.

Au niveau des contraintes

Plusieurs autres contraintes pourraient être ajoutées au système telles des contraintes de symétrie, de longueur, d'angle, etc. Ces possibilités ont précédemment été décrites à la section 4.4. Elles

sont destinées à satisfaire les attentes des observateurs concernant des propriétés géométriques qu'ils espèrent naturellement retrouver dans les scènes reconstruites et dont l'absence est souvent immédiatement détectée. L'ajout de telles contraintes impliquerait le traitement d'équations non-linéaires par notre système. C'est un aspect important qu'il faudra bien considérer, étant donné les problèmes reliés à l'utilisation de tels types d'équations.

La manipulation des contraintes pourrait être améliorée de plusieurs façons. Pour vérifier l'effet d'une contrainte sur une partie seulement du modèle, il serait intéressant de pouvoir activer et désactiver *localement* une contrainte. Pour l'instant, lorsqu'un type de contrainte est activé durant la reconstruction, ce sont toutes les contraintes de ce type qui entrent en jeu dans le calcul du modèle entier. En activant localement une contrainte, par exemple une contrainte de perpendicularité entre deux polygones, on serait à même de mieux juger son effet.

Nous avons déjà abordé dans la section 4.4 l'ajout de poids associés à un certain type de contrainte ou à une contrainte en particulier. Ce serait une fonctionnalité très utile et nous pensons que les résultats que nous obtiendrions seraient de beaucoup améliorés. La méthode de résolution que nous utilisons (décomposition en valeurs singulières) permet facilement d'ajouter un vecteur de poids pour pondérer chaque équation du système à résoudre.

L'utilisation d'un *visualisateur* de contraintes permettrait de gérer efficacement de telles fonctionnalités. Ainsi, on pourrait sélectionner une primitive et afficher les contraintes qui lui sont associées. Il serait alors possible de modifier leurs poids, de les activer ou de les désactiver, et d'évaluer l'effet produit sur le modèle.

Le but premier de ces améliorations est évidemment de donner le plus de contrôle possible à l'usager quant à la spécification et la manipulation des contraintes qu'il introduit. On ne voudrait pas qu'il devienne "victime" de ses propres contraintes et qu'il perde le contrôle sur la reconstruction du modèle. La manière dont Gleicher [Gle94] traite et manipule les contraintes serait intéressante à étudier pour ces améliorations.

Au niveau du modèle

Lorsque le système effectue une itération de convergence, il recalcule toutes les caméras et toute la géométrie de la scène. Si celle-ci contient beaucoup d'objets et que certains étaient déjà reconstruits de façon satisfaisante, on perd du temps à les recalculer puisqu'ils ne devraient normalement pas être modifiés de façon significative. Il serait donc intéressant de pouvoir spécifier que l'on est satisfait d'une partie du modèle (par exemple, en sélectionnant les primitives qui en font partie) et que l'on

ne désire plus y faire de changement. Le système ne la considèrerait alors plus dans le calcul de la géométrie mais seulement dans le calcul des caméras. De même, on pourrait spécifier que l'on est satisfait de la récupération d'une caméra en évaluant la projection du modèle sur l'image concernée. Le système n'aurait donc plus à calculer cette caméra à chaque itération mais s'en servirait pour recalculer la géométrie. Cette extension résulterait en une accélération des calculs (puisqu'on en élimine une partie) et une meilleure stabilité du système puisque ces parties contraintes à l'immobilité continuent à agir sur la reconstruction des autres. On parlerait alors encore plus de reconstruction *incrémentale*.

Une autre amélioration facile à implémenter et potentiellement utile serait la possibilité de pouvoir utiliser des primitives dont la position calculée servirait au calcul des matrices de projection mais qui ne feraient pas partie du modèle reconstruit. Par exemple, des ombres ou des parties de textures qui ne constituent pas vraiment des objets 3D mais dont on voit les projections dans plusieurs images.

Au niveau du système en général

L'introduction d'une erreur de correspondance par l'usager lors de la modélisation constitue un problème majeur et peut entraîner des conséquences désastreuses sur la géométrie et la calibration. Si le modèle est peu complexe, on pourra déceler cette erreur presque immédiatement. Par contre, dès que le modèle devient de plus en plus contraint, une telle erreur mettra plus de temps à devenir perceptible et il sera alors trop tard pour déterminer son origine sans devoir vérifier toutes les correspondances une à une. Des erreurs au niveau des contraintes pourraient également avoir des effets néfastes. Par exemple, la spécification d'une normale de sens erroné influencerait le "retournement" du polygone. Si celui-ci possède plusieurs contraintes, on observerait une légère déviation qui pourrait avoir des origines diverses et ainsi confondre l'usager. Par contre, s'il est plus "libre" (moins contraint), la cause du problème serait facilement identifiable. Une opération *annuler* à répétition ne serait pas forcément utile si l'on ne connaît pas exactement l'origine du problème.

Il est important de noter qu'une erreur affecte la géométrie du modèle. Cette géométrie est utilisée pour le calcul des matrices, qui sont à leur tour utilisées pour recalculer la géométrie et ainsi de suite. En cas d'erreur sémantique, le modèle va donc invariablement se dégrader dû à ce mécanisme itératif. On ne convergerait donc plus mais on pourrait plutôt diverger peu à peu. Plus le système sera contraint, moins cette dégradation sera visible, la contrainte fautive ayant moins d'influence sur le résultat parmi une centaine d'autres contraintes que parmi une dizaine. De plus, notre méthode de résolution actuelle accorde une importance égale à chaque contrainte d'où le risque de ne pas détecter immédiatement une erreur lorsque celle-ci s'introduit dans un système déjà très contraint.

La détection automatique ou assistée des erreurs serait donc une fonctionnalité d'utilité non négligeable. Plusieurs possibilités s'offrent à nous mais aucune n'est réellement satisfaisante car cette détection constitue un problème assez ardu dans le cadre d'un système tel que le nôtre.

Nous pourrions donner la possibilité à l'usager de retourner à l'état précédant l'introduction d'une nouvelle contrainte s'il s'aperçoit après quelques itérations que quelque chose ne va pas. Nous avons vu dans le chapitre précédent que l'introduction d'une nouvelle contrainte pouvait temporairement perturber le système, c'est pour cela qu'il est nécessaire d'itérer quelques fois avant de pouvoir juger si une contrainte a eu ou non un réel effet négatif sur la reconstruction. Si le système est très contraint et que l'erreur passe inaperçue, cette fonctionnalité ne serait pas très utile.

Afin de visualiser l'influence d'une contrainte (par exemple, la dernière introduite), on pourrait accorder un gros poids à cette contrainte et ainsi mieux observer son effet sur le modèle et sa reprojection. Un mécanisme qui attribuerait un tel poids tour à tour à chacune des contraintes du système pourrait constituer un moyen de vérifier si une contrainte en particulier a un effet douteux ou néfaste sur la reconstruction. Ce serait ici encore à l'usager de constater l'effet de l'augmentation de pondération d'une contrainte, un changement drastique dans le modèle (position des primitives ou reprojection largement modifiées[1]) n'étant pas nécessairement une indication de la présence d'une erreur mais pouvant être un résultat normal ou volontaire d'une action de l'usager. C'est principalement cette dernière raison qui fait qu'il n'existe, à notre avis, aucune solution *automatique* facile pour détecter des erreurs dans un tel système. Le mieux que l'on puisse faire est d'utiliser des indicateurs d'erreurs potentielles et de laisser l'usager juger s'il y a matière à s'inquiéter. Numériquement, un indicateur potentiel pourrait être la détection de la contrainte "la moins satisfaite". Afin de trouver la meilleure solution, *SVD* effectue une minimisation de la somme des erreurs des contraintes. La contrainte la moins satisfaite est celle dont l'erreur impliquée par la solution choisie est la plus élevée. Cette contrainte a plus de risques que les autres d'être fautive puisque la solution choisie convient le plus à toutes les autres contraintes sauf à celle-là.

La méthode la plus sûre pour détecter les erreurs d'interaction reste donc la vérification manuelle. L'ajout de fonctionnalités pour faciliter cette tâche serait sûrement très appréciée, par exemple le visualisateur de contraintes ou les indicateurs mentionnés plus haut. *Rekon* indique pour l'instant le sens dans lequel ont été tracé les polygones et le sommet de départ utilisé ainsi que les primitives correspondant à une primitive sélectionnée.

[1] Le système peut évaluer la reprojection automatiquement en calculant la distance entre la reprojection des primitives 3D et leur équivalent 2D.

Incertitude et précision des modèles

Un critère important qu'un système de reconstruction devrait être capable de considérer est la précision à laquelle un usager désire qu'un modèle soit reconstruit. Pour cela, on devrait être capable de caractériser les erreurs possibles pour la position d'un point reconstruit (par exemple). On pourrait estimer l'incertitude à laquelle un usager place un point sur les images selon plusieurs critères comme son habileté, la précision des images, l'aire (en pixels) sur laquelle se projette une primitive, etc. Ces critères serviraient à générer une zone d'incertitude autour du point qui, répercutée dans l'espace, amènerait à définir un espace d'incertitude pour la position 3D du point. Cet espace pourrait être une mesure de la précision du modèle et servirait ainsi à définir une limite sur l'erreur maximale du modèle. Zhang et Schenk [ZS97] caractérisent et manipulent cette notion d'incertitude sous la forme de matrices de covariance qu'ils essaient de maintenir tout au long de la reconstruction/calibration. Ils font remarquer que l'erreur sur les points reconstruits n'est pas la même dans toutes les directions (par exemple, il y a normalement une erreur plus importante en profondeur que dans les directions latérales) et qu'elles sont différentes d'un point à l'autre.

6.3 Extensions

Le fait de pouvoir récupérer la géométrie d'un objet à partir d'images de cet objet nous donne la possibilité de créer un modèle réaliste de l'objet. Ce type d'objet peut être utilisé dans maintes applications infographiques telles les environnements virtuels immersifs, les effets spéciaux, les jeux vidéo, etc.

Si l'on veut pouvoir créer de tels modèles réalistes (photoréalistes), il faudrait pouvoir récupérer correctement les textures du monde réel car celles générées par ordinateur sont limitées et souvent détectables par l'observateur. *Rekon* offre cette possibilité [Oui98] grâce à la récupération de caméras et de géométrie qu'il effectue. Pour un polygone 3D reconstruit, il extrait les textures se trouvant dans les primitives 2D correspondantes sur les images. L'usager peut être intégré dans le processus et aider à corriger les anomalies éventuelles de récupération de texture (par exemple, un mauvais alignement des textures extraites). Le fait de posséder le modèle géométrique permet de pouvoir traiter les occlusions lors de l'extraction de textures et de n'utiliser que les portions de texture visibles sur l'image considérée. Un algorithme de combinaison de couleurs est utilisé pour tenter d'obtenir la "meilleure" texture possible et d'en corriger les défauts (spécularités, mauvais alignement, écrasement par la perspective, etc.).

L'étape suivante serait la récupération de l'illumination et des propriétés (réflectance, transparence, détails de surface (*bump map*), etc.) des objets de la scène. On pourrait ainsi changer ces propriétés, enlever ou ajouter des sources lumineuses, ajouter ou enlever des objets tout en obtenant une apparence et des ombres correctes, etc. Plusieurs équipes de recherche travaillent actuellement sur ces problèmes (voir les travaux de Shashua [Sha92] pour plus d'informations sur le sujet, ou plus récemment, ceux de Sato *et al.* [SWI97] et Zhang [Zha98]). L'usager pourrait là aussi être intégré dans le processus pour indiquer les positions des sources de lumière, les ombres ou les spécularités et ainsi aider le système à récupérer l'illumination et les diverses propriétés des objets.

6.4 Applications

Un système de reconstruction tel que le nôtre peut avoir plusieurs applications différentes. Tout d'abord, le modèle géométrique résultant d'une reconstruction à partir d'images peut être, tout comme tout autre modèle créé par des procédés plus traditionnels, utilisé directement par des logiciels qui permettront à des artistes d'y appliquer des textures et d'en varier les propriétés et l'illumination, ou même d'y ajouter des détails à l'aide d'un modeleur, une fois la structure générale récupérée. Par exemple, si des architectes veulent proposer à un client des extensions à sa maison, ils peuvent récupérer le modèle de la maison à partir de photographies puis ajouter leurs idées à la main.

Lorsque l'on sera capable d'extraire correctement toutes les caractéristiques d'une scène à partir d'images de cette scène (géométrie, textures, illumination, propriétés des objets, etc.), on disposera de tous les outils nécessaires à l'obtention de modèles réalistes. Les applications de réalité augmentée, très exigeantes au niveau de la qualité des modèles utilisés car l'observateur ne devrait pas pouvoir distinguer entre les objets réels et les objets de synthèse, y verraient une solution à leurs problèmes. Par contre, tous les aspects de la reconstruction à partir d'images mentionnés constituent actuellement un champ actif de recherche et nous sommes encore loin d'obtenir facilement des modèles à partir d'images qui peuvent réellement confondre un observateur averti.

Il existe également plusieurs applications potentielles pour un tel système dans le domaine de la robotique et de l'intelligence artificielle. En robotique par exemple, on pourrait indiquer à un robot un endroit d'une image à atteindre ou un objet, visible sur une image, à aller chercher. Le robot possédant le modèle de la scène et les relations entre images et scène devrait pouvoir se rendre à l'endroit désigné sans problèmes.

Il devrait même être possible de modifier interactivement la scène et d'obtenir les images résultant de ces modifications. Ainsi, si quelqu'un désire bouger des objets qu'il voit sur une image, il peut

indiquer sur les images la ou les primitives 2D correspondant à l'objet 3D ainsi que l'endroit où il voudrait le placer. Le système connaît le modèle 3D relié à ces primitives et peut donc déplacer l'objet dans la scène reconstruite puis produire de nouvelles images, d'un point de vue identique ou différent de celui des images originales. Des algorithmes de remplissage (*filling algorithms*) seraient éventuellement nécessaires pour combler les vides éventuels laissés par le déplacement de l'objet. Debevec [Deb98] présente un algorithme de remplissage en espace objet qui utilise une interpolation des couleurs similaires à celles des polygones entourant un "trou" pour le remplir.

Les possibilités d'emploi pour un tel système sont multiples et nous ne pouvons en donner ici une liste exhaustive. Le fait de posséder des images d'une scène, du modèle 3D de cette scène et des relations entre les images et le modèle offre une multitude de possibilités intéressantes pour plusieurs domaines.

FIG. 6.1: Sans commentaires.

Annexe A

Coordonnées de Plücker

Une ligne en deux dimensions s'exprime à l'aide d'une équation de ligne de la forme $ax+by+c=0$. En trois dimensions, un plan s'exprime avec une relation similaire : $ax+by+cz+d=0$. Malheureusement, il n'existe pas d'équivalent aussi simple pour une ligne en trois dimensions.

Cette annexe présente une représentation de ligne utilisée en géométrie projective : la représentation d'une ligne en coordonnées de Plückers. Divers manuels de géométrie projective en expliquent les principes [Han93][SK52][God] et le lecteur intéressé y est référé pour plus de détails.

A.1 Définition

On peut voir une ligne en 3D de deux façons. Une ligne en 3D est soit formée en connectant deux points de l'espace ou en intersectant deux plans. Ces deux représentations sont équivalentes (duales).

Soit L une ligne de l'espace. Elle peut être représentée par n'importe quel couple de points (non confondus) lui appartenant, $P_1 = \begin{bmatrix} x_1 & y_1 & z_1 & h_1 \end{bmatrix}^T$ et $P_2 = \begin{bmatrix} x_2 & y_2 & z_2 & h_2 \end{bmatrix}^T$. Tout point P de la ligne peut s'exprimer comme une combinaison linéaire de ces deux points [Han93],

$$P = t_1 P_1 + t_2 P_2. \tag{A.1}$$

Cette représentation est dépendante du choix des points P_1 et P_2 alors qu'une ligne pourrait être indifféremment représentée par n'importe quel autre couple de points lui étant incidents.

Si nous posons

$$P_{\alpha\beta} = \alpha_1\beta_2 - \alpha_2\beta_1 \qquad \text{où} \qquad \alpha, \beta \in \{x, y, z, h\}, \tag{A.2}$$

nous obtenons un ensemble de seize nombres qui peuvent être considérés comme des coordonnées pour la ligne L. Il est clair que de ces seize nombres, les quatre de la forme $P_{\alpha\alpha}$ sont nuls.

De plus, nous voyons également que $P_{\alpha\beta} = -P_{\beta\alpha}$, ce qui réduit le nombre de coefficients indépendants à six. Nous arrivons donc à un ensemble de six coordonnées pour la ligne $L = \begin{bmatrix} P_{xy} & P_{xz} & P_{yz} & P_{xh} & P_{yh} & P_{zh} \end{bmatrix}^T$.

L'équation A.2 peut également s'exprimer sous forme matricielle par une matrice anti-symétrique[1] 4×4

$$\begin{aligned} L &= P_1 P_2^T - P_2 P_1^T \\ &= \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ h_1 \end{bmatrix} \begin{bmatrix} x_2 & y_2 & z_2 & h_2 \end{bmatrix} - \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ h_2 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 & h_1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & x_1y_2 - x_2y_1 & x_1z_2 - x_2z_1 & x_1h_2 - x_2h_1 \\ x_2y_1 - x_1y_2 & 0 & y_1z_2 - y_2z_1 & y_1h_2 - y_2h_1 \\ x_2z_1 - x_1z_2 & y_2z_1 - y_1z_2 & 0 & z_1h_2 - z_2h_1 \\ x_2h_1 - x_1h_2 & y_2h_1 - y_1h_2 & z_2h_1 - z_1h_2 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & P_{xy} & P_{xz} & P_{xh} \\ -P_{xy} & 0 & P_{yz} & P_{yh} \\ -P_{xz} & -P_{yz} & 0 & P_{zh} \\ -P_{xh} & -P_{yh} & -P_{zh} & 0 \end{bmatrix}. \end{aligned}$$

qui constitue la *matrice de Plücker* de la ligne L.

Il est à noter que L est une matrice nulle si les deux points qui ont servi à la construire sont confondus.

La propriété fondamentale de ces coordonnées est que leur ratio est indépendant du choix des deux points P_1 et P_2 sur L. Elles forment un ensemble de coordonnées unique de L. En effet, si nous avons deux autres points de cette ligne P_1' et P_2' tels que :

$$P_1' = fP_1 + gP_2$$
$$P_2' = mP_1 + nP_2$$

par l'équation A.1.

Chaque coordonnée α_i' des nouveaux points est reliée aux coordonnées α_i des points P_1 et P_2 par

$$\alpha_1' = f\alpha_1 + g\alpha_2$$
$$\alpha_2' = m\alpha_1 + n\alpha_2$$

avec $\alpha_i' \in \{x_i', y_i', z_i', h_i'\}$, l'ensemble des coordonnées de P_i'.

[1] La transposée d'une matrice anti-symétrique est égale à sa négation.

Ainsi,

$$\begin{aligned} P'_{\alpha\beta} &= \alpha'_1\beta'_2 - \alpha'_2\beta'_1 \qquad \text{par la relation de Plücker A.2} \\ &= (f\alpha_1 + g\alpha_2)(m\beta_1 + n\beta_2) - (m\alpha_1 + n\alpha_2)(f\beta_1 + g\beta_2) \\ &= fn\alpha_1\beta_2 + mg\alpha_2\beta_1 - mg\alpha_1\beta_2 - fn\alpha_2\beta_1 \\ &= (fn - mg)\alpha_1\beta_2 + (mg - fn)\alpha_2\beta_1 \\ &= (fn - mg)(\alpha_1\beta_2 - \alpha_2\beta_1) \\ &= kP_{\alpha\beta} \end{aligned}$$

où $\alpha', \beta' \in \{x', y', z', h'\}$ et k est un coefficient de proportionnalité.

Les coordonnées de Plücker de L sont donc définies à un facteur de proportionnalité près et L est de fait une représentation *homogène* d'une ligne dans l'espace. Ainsi, deux matrices de Plücker qui diffèrent par un facteur scalaire représentent la même ligne.

Etant donné que les six coordonnées sont homogènes, seulement cinq sont indépendantes. La relation qui existe entre les six coordonnées de toute ligne est décrite par l'équation

$$P_{xh}P_{yz} + P_{yh}P_{zx} + P_{zh}P_{xy} = 0.$$

Cette représentation est appelée la *forme-point* d'une ligne [Han93] ou les coordonnées *radiales* [God] de la ligne. Il existe un dual à cette représentation. Au lieu de considérer la ligne comme passant par deux points, on peut la considérer comme étant l'intersection de deux plans. Le même genre de raisonnement aboutit à la *forme-plan* d'une ligne ou les coordonnées *axiales* de la ligne,

$$\begin{bmatrix} 0 & Q_{ab} & Q_{ac} & Q_{ad} \\ -Q_{ab} & 0 & Q_{bc} & Q_{bd} \\ -Q_{ac} & -Q_{bc} & 0 & Q_{cd} \\ -Q_{ad} & -Q_{bd} & -Q_{cd} & 0 \end{bmatrix},$$

où les coordonnées sont reliées par

$$Q_{ad}Q_{bc} + Q_{bd}Q_{ca} + Q_{cd}Q_{ab} = 0.$$

Il est possible de passer d'une représentation à l'autre par une simple manipulation algébrique.

A.2 Propriétés

La dérivation des propriétés suivantes se retrouve dans le cours de géométrie projective de Hanrahan [Han93]. Nous n'en ferons pas les démonstrations ici.

- Chaque rangée et chaque colonne de la matrice de Plücker représente un point en coordonnées

homogènes sur cette ligne. Les trois premières sont les intersections de la ligne avec chacun des plans de coordonnées tandis que la dernière représente l'intersection avec le plan à l'infini ($h = 0$). Ainsi, pour construire une ligne dans l'espace à partir de sa représentation de Plücker, on peut extraire directement deux points de la matrice.

- Dans la représentation à partir de plans d'une ligne, les rangées représentent des plans passant par la ligne. La quatrième rangée représente le plan passant par la ligne et l'origine.

- Le vecteur directeur V_1 de la ligne se retrouve dans la matrice de Plücker : $V_1 = \begin{bmatrix} P_{xh} & P_{yh} & P_{zh} \end{bmatrix}^T$. Lorsque les deux points P_1 et P_2 sont normalisés, $h = 1$ et V_1 correspond donc à $\begin{bmatrix} P_{2x} - P_{1x} & P_{2y} - P_{1y} & P_{2z} - P_{1z} \end{bmatrix}^T$, la définition du vecteur directeur de la ligne passant par P_1 et P_2.

- Le vecteur V_2 normal au plan passant par la ligne et l'origine est défini par $V_2 = \begin{bmatrix} P_{yz} & P_{zx} & P_{xy} \end{bmatrix}^T$.

- Evidemment, on a $V_1 \perp V_2$.

- Le point d'intersection P d'une ligne L et d'un plan Π se retrouve en multipliant la forme-point de L par les coordonnées du plan, $P = L\Pi$. Cela peut aussi être utilisé pour déterminer si L est incidente au plan Π. Dans ce cas, $L\Pi = 0$.

- De même, on peut calculer le plan passant à travers une ligne L et un point P en multiplant la forme-plan de L par P. Si P appartient à la ligne, $PL = 0$.

- L'intersection de deux lignes de l'espace, si elle existe, peut se trouver en multipliant la forme-point d'une des lignes par la forme-plan de l'autre.

Bibliographie

[ASP66] American Society of Photogrammetry. *Manual of Photogrammetry*. Editor-in-chief : Morris M. Thompson, 1966.

[Bor97] George D. Borshukov. *New Algorithms for Modeling and Rendering Architecture from Photographs*. Thèse de doctorat, University of California, Berkeley, 1997.

[Bou] http ://www.inria.fr/robotvis/personnel/sbougnou/Meta3DViewer/EpipolarGeo.html

[BR97] Sylvain Bougnoux et Luc Robert. « TotalCalib : a Fast and Reliable System for Off-line Calibration of Images Sequences ». In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 1997.

[Car95] Stefan Carlsson. « Duality of Reconstruction and Positioning from Projective Views ». Rapport technique CVAP175, Computational Vision and Active Perception Laboratory, Royal Institute of Technology, Stockholm, Suède, avril 1995.

[Cha93] H. Chabbi. « Checking 3D planar Surfaces using projective Geometry ». In *Proceedings of the* 8^{th} *Scandinavian Conference on Image Analysis*, 1993.

[CT97] George Chou et Seth Teller. « Multi-Image Correspondence using Geometric and Structural Constraints ». In *1997 Image Understanding Workshop*, 1997.

[Deb] http ://www.cs.berkeley.edu/ debevec/Campanile/

[Deb98] Paul E. Debevec. « Efficient View-Dependant Image-Based Rendering with Projective Texture-Mapping ». In Georges Drettakis et Nelson Max, éditeurs. *Eurographics Rendering Workshop 1998*, page 105–116, New York City, NY, juin 1998. Eurographics, Springer Wien.

[DTM96] Paul E. Debevec, Camillo J. Taylor et Jitendra Malik. « Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-Based Approach ». In Holly Rushmeier, éditeur. *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 11–20. ACM SIGGRAPH, Addison Wesley, août 1996.

[Fau92] Olivier Faugeras. « What Can Be Seen in Three Dimensions with an Uncalibrated Stereo Rig ? ». In Giulio Sandini, éditeur. *Proceedings of Computer Vision (ECCV '92)*, volume 588, pages 563–578, Berlin, Allemagne, mai 1992. Springer.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision — A Geometric Viewpoint.* MIT Press, 1993.

[Fau95] Olivier Faugeras. « Stratification of 3-D vision : Projective, Affine, and Metric Representations ». *Journal of The Optical Society of America A*, volume 12, numéro 3, pages 465–484, mars 1995.

[FHM+93] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin et C. Proy. « Real time correlation-based stereo : algorithm, implementatinos and applications ». Rapport technique 2013, Institut National De Recherche en Informatique et en Automatique (INRIA), 06902 Sophia Antipolis, France, 1993.

[FLM92] O. D. Faugeras Q.-T. Luong et S. J. Maybank. « Camera Self–Calibration : Theory and Experiments ». In Giulio Sandini, éditeur. *Proceedings of Computer Vision (ECCV '92)*, volume 588 de *LNCS*, pages 321–334, Berlin, Germany, mai 1992. Springer.

[FLR+95] Olivier Faugeras, Stéphane Laveau, Luc Robert, Gabriella Csurka et Cyril Zeller. « 3-D Reconstruction of Urban Scenes from Sequences of Images ». Rapport technique 2572, INRIA, Sophia-Antipolis, France, 1995.

[FT86] Olivier Faugeras et Georges Toscani. « The Calibration Problem for Stereo ». In *Proceedings of CVPR '86*, pages 15–20, 1986.

[FvDFH90] James Foley, Andries van Dam, Steven Feiner et John Hughes. *Computer Graphics : Principles and Practice*. Addison-Wesley, 2ème édition, 1990.

[Gan84] Sundaram Ganapathy. « Decomposition of Transformation Matrices for Robot Vision ». *Pattern Recognition Letters*, volume 2, numéro 6, pages 401–412, décembre 1984.

[Gle94] Michael Gleicher. *A Differential Approach to Graphical Manipulation*. Thèse de doctorat, Carnegie Mellon University, 1994.

[God] Lucien Godeaux. *Leçons de géométrie analytique à trois dimensions*. Universite de Liege. Cours de la faculte des sciences. Quatrième édition.

[Han93] Pat Hanrahan. « 2D and 3D Projective Geometry ». Université de Princeton, Notes de cours (CS403), chapitres 12 et 13 , 1993.

[Han96] Klaus Hanke. « Accuracy Study Project of Eos Systems' PhotoModeler ». Rapport technique, University of Innsbruck, Austria, 1996.

[Har97] Richard I. Hartley. « In Defense of the Eight-Point Algorithm ». *IEEE Trans. on Pattern Analysis Machine Intelligence*, volume 19, numéro 6, pages 580–593, juin 1997.

[HB94] Donald Hearn et M. Pauline Baker. *Computer Graphics*. Prentice Hall, deuxième édition, 1994.

[IK88] J. Illingworth et J. Kittler. « A Turvey of the Hough Transform ». *Computer Vision, Graphics, and Image Processing*, volume 44, pages 87–116, 1988.

[KHM95] Craig Kolb, Pat Hanrahan et Don Mitchell. « A Realistic Camera Model for Computer Graphics ». In Robert Cook, éditeur. *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 317–324. ACM SIGGRAPH, Addison Wesley, août 1995.

[Kru13] E. Kruppa. « Zur Ermittlung eines Objecktes aus zwei Perspektiven mit innerer Orientierung ». *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw., Kl. Abt. IIa*, volume 122, pages 1939–1948, 1913.

[Lbar96] F. Leymarie, A. de la Fortelle, J.J. Koenderink, A.M.L. Kappers, M. Stavridi, B. van Ginneken, S. Muller, K. Krake, O. Faugeras, L. Robert, S. Laveau et C. Zeller. « REALISE : Reconstruction of REALity from Image Sequences ». In *International Conference on Image Processing*, pages 651–654, Lausanne (Switzerland), septembre 1996. IEEE Signal Processing Society.

[LF93] Quang-Tuan Luong et Olivier Faugeras. « Self-Calibration of a Stereo Rig from Unknown Camera Motions and Point Correspondences ». Rapport technique 2014, INRIA Sophia-Antipolis, France, Juillet 1993.

[LGG95] C.-E. Liedtke, Oliver Grau et S. Growe. « Use of Explicit Knowledge for the Reconstruction of 3-D Object Geometry ». In *6th International Conference CAIP'95 Computer Analysis of Images and Patterns*, septembre 1995.

[LH81] H. C. Longuet-Higgins. « A Computer Algorithm for Reconstructing a Scene from Two Projections ». *Nature*, volume 293, pages 133–135, 1981.

[LHF90] Y. Liu, T.S. Huang et Olivier Faugeras. « Determination of Camera Location from 2-D to 3-D Line and Point Correspondences ». *IEEE Trans. on Pattern Analysis Machine Intelligence*, volume 12, numéro 1, pages 28–37, janvier 1990.

[Mar82] David Marr. *Vision*. W. H. Freeman and Company, San Francisco, 1982.

[MYTG94] Vannary Meas-Yedid, Jean-Philippe Tarel et André Gagalowicz. « Calibration métrique faible et construction interactive de modèles 3D de scènes ». In *Congrès Reconnaissance des Formes et Intelligence Artificielle*, Paris, France, 1994. AFCET.

[MZ92] Joseph Mundy et Andrew Zisserman, éditeurs. *Geometric Invariance in Computer Vision*. MIT Press, Cambridge, Massachusetts, 1992.

[Oui98] Mathieu Ouimet. « Extraction et unification de textures en provenance d'images ». Mémoire de maîtrise, Département I.R.O., Université de Montréal, 1998.

[PFTV92] W. H. Press, B. P. Flannery, S. A. Teukolsky et W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, deuxième édition, 1992.

[Pho] http ://www.photomodeler.com

[POF98] Pierre Poulin, Mathieu Ouimet et Marie-Claude Frasson. « Interactively Modeling with Photogrammetry ». In Georges Drettakis et Nelson Max, éditeurs. *Eurographics Rendering Workshop 1998*, page 93–104, New York City, NY, juin 1998. Eurographics, Springer Wien.

[RA90] David F. Rogers et J. Alan Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, deuxième édition, 1990.

[RC98] Sebastien Roy et Ingemar J. Cox. « A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem ». In *Int. Conf. on Computer Vision (ICCV'98)*, pages 492–499, janvier 1998.

[Rob96] Luc Robert. « Camera Calibration without Feature Extraction ». *Computer Vision and Image Understanding*, volume 63, numéro 2, pages 314–325, mars 1996.

[Sha92] Amnon Shashua. *Geometry and Photometry in 3D Visual Recognition*. Thèse de doctorat, Massachusetts Institute of Technology, 1992.

[SK52] J.G. Semple et G. T. Kneebone. *Algebraic Projective Geometry*. Oxford Science Publication, 1952.

[SK94] R. Szeliski et S. B. Kang. « Recovering 3D shape and motion from image streams using non-linear least squares ». *Journal of Visual Communication and Image Representation*, volume 5, numéro 1, pages 10–28, 1994.

[Str87] Thomas M. Strat. « Recovering the Camera Parameters from a Transformation Matrix ». In Martin A. Fischler and Oscar Firschein, éditeurs, *Readings in Computer Vision : Issues, Problems, Principles, and Paradigms*, pages 93–100. Morgan Kaufman, 1987.

[Str94] André Streilein. « Towards Automation in Architectural Photogrammetry : CAD-Based 3D-Feature Extraction ». *ISPRS Journal of Photogrammetry and Remote Sensing*, volume 49, numéro 5, pages 4–15, octobre 1994.

[Sut63] Ivan E. Sutherland. « Sketchpad : A Man-Machine Graphical Communication System ». In *Proceedings AFIPS Spring Joint Computer Conference*, volume 23, pages 329–346, Detroit, Michigan, mai 1963.

[SWI97] Y. Sato, M.D. Wheeler et K. Ikeuchi. « Object Shape and Reflectance Modeling from Observation ». In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 379–387. ACM SIGGRAPH, Addison Wesley, août 1997.

[Sze93] R. Szeliski. « Robust shape recovery from occluding contours using a linear smoother ». In *Image Understanding Workshop*, pages 939–950. Defense Advanced Research Projects Agency, Software and Intelligent Systems Office, avril 1993.

[Vyg90] Marc Vygoski. *Aide-Mémoire de mathématiques supérieures*. Éditions de Moscou, 1990.

[WG96] S. Weik et Oliver Grau. « Recovering 3-D Object Geometry using a Generic Constraint Description ». In *ISPRS96 - 18th Congress of the International Society for Photogrammetry and Remote Sensing*, juillet 1996.

[Zha98] Zhengyou Zhang. « Modeling Geometric Structure and Illumination Variation of a Scene from Real Images ». In *Sixth International Conference on Computer Vision, Bombay, Inde*. IEEE Computer Society Press, Janvier 1998.

[ZS97] Zhengyou Zhang et Veit Schenk. « Self-Maintaining Camera Calibration Over Time ». In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'97)*, pages 231–236, Porto Rico, June 1997.