3D Modeling From Images Using Geometric Constraints

Marta Wilczkowiak

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|__|__|__|__|__|__|__|__|__|__|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité:
Imagerie Vision et Robotique

Ecole Doctoral:
Mathématiques, Sciences et technologies de l'information, Informatique

présentée et soutenue publiquement
par

Marta Wilczkowiak

le 27 Avril 2004

3D Modelling From Images Using Geometric Constraints

Directeurs de thèse: Edmond Boyer, Peter Sturm et Radu Horaud.

JURY

Roger Mohr	Président
Roberto Cipolla	Rapporteur
Michel Dhome	Rapporteur
Edmond Boyer	Examinateur
Peter Sturm	Examinateur
Gilles Trombettoni	Examinateur

Thèse préparée au laboratoire GRAVIR - IMAG au sein du projet MOVI
INRIA Rhône-Alpes, 655 av. de l'Europe, 38334 Sant Ismier, FRANCE

Abstract

Image-based 3D modelling is often an underconstrained and noise-sensitive process. Incorporation of geometrical constraints increases the robustness of this process, especially when dealing with small image sets. This relies on a human operator and his intuitive knowledge of the main properties of the objects commonly present in scenes. In this thesis we propose various approaches for the camera and model constraints satisfaction. In particular, we propose an original method for the satisfaction of constraints in an exact way. Contrary to most existing work, each proposed method is completed by algorithms dealing with problems engendered by insufficient user-provided input. The introduced methods are validated by reconstruction of 3D models from small image sets, and even from single images. The images used for reconstructions are taken from various sources, such as the Internet, postcards or architectural drawings.

Keywords : Reconstruction, Calibration, Geometric Constraints.

Résumé

La modélisation tridimensionnelle à partir d'images est un processus souvent sous-contraint et sensible au bruit. L'usage de contraintes géométriques permet de rendre ce processus plus robuste, même à partir de peu d'images, en prenant en compte les connaissances intuitives de l'utilisateur sur les objets présents dans la scène. Nous proposons dans cette thèse différentes approches pour la satisfaction de contraintes sur la scène et les caméras. En particulier, nous introduisons une approche novatrice pour la satisfaction de contraintes de manière exacte. Contrairement à la plupart des approches existantes, chaque méthode proposée est accompagnée d'algorithmes pour gérer des cas dans lesquels l'information fournie par l'utilisateur n'est pas suffisante pour reconstruire un modèle unique. Les méthodes proposées dans cette thèse sont validées par la reconstruction de modèles tridimensionnels à partir de petits ensembles d'images, voire à partir d'une seule image. Les images utilisées proviennent de sources très variées, telles que l'Internet, des cartes postales ou des dessins architecturaux.

Mots Clès : Reconstruction, Calibrage, Contraintes Géométriques.

Remerciements

Tout d'abord, je tiens à exprimer mes remerciements à Radu Horaud, qui m'a accueilli dans l'équipe MOVI, et à Long Quan, qui m'a offert la possibilité d'effectuer cette thèse dans le cadre du projet européen VISIR. Je souhaite également remercier Roger Mohr pour m'avoir fait l'honneur de présider mon jury de thèse, ainsi que Roberto Cipolla et Michel Dhome, rapporteurs de thèse, pour avoir montré de l'intérêt et avoir accepté de juger mon travail.

Je souhaite remercier tout particulièrement Edmond Boyer et Peter Sturm, qui ont encadré cette thèse, d'une part pour leur disponibilité et les nombreuses discussions qui ont apporté énormément à ce travail, et d'autre part pour leur confiance et de m'avoir laissé toujours beaucoup de liberté dans mes choix. Je voudrais également les remercier pour leur sympathie et leur soutien moralăqui m'ont beaucoup aidé tout au long de la préparation de cette thèse.

Je tiens aussi à remercier très chaleureusement Gilles Trombettoni. Notre collaboration a été une grand aventure, pleine de moments de désespoir, mais grâce à sa bonne humeur, patience et amitié, ce fut une expérience extraordinaire; le fait que nous ayons réussi à donner une forme à nos idées m'a apporté beaucoup de satisfaction.

Je voudrais adresser aussi un très très très grand merci à Thomas, Dave, Norman, Ankur, Navneet, Keneth, Jean-Sebastien et Laks qui ont relu et corrigé les parties de cette thèse écrites en anglais, et Hélène, Thomas, Laurence et Frank qui se sont occupés des parties en français. Quand vous aurez des traductions polonaises à faire, vous pourrez toujours compter sur moi !

Et bon, maintenant ça devient difficile, je voudrais remercier toutes les personnes grâce à qui mon séjour à Grenoble a été un temps très agréable et qui m'ont donné beaucoup de sympathie et de soutien, surtout au cours des derniers mois de la rédaction. Surtout Laurence, Veronika, Hélène, Thomas, Christophe, Virginie, Raph, Ann, Laure, Monika, Pawel, Wiktor et Frank, vous resterez toujours dans mon coeur !

Finalement, je tiens à remercier ma famille de leur soutien et confiance, et particulièrement mes parents pour être venus à ma soutenance.

Contents

Introduction (in French)

Contexte et Motivation

La plupart des inventions techniques ont été inspirées par la nature. Aujourd'hui, nous sommes capable de comprendre et d'imiter un grand nombre des phénomènes naturels, cependant, il existe encore des domaines où les réalisations scientifiques sont loin d'égaler leur modèle naturel. C'est le cas du projet qui a inspiré des générations de chercheurs: la création d'un être à notre image. De nouvelles perspectives pour s'approcher de ce défi sont apparues avec le développement des ordinateurs. Nous sommes toujours dans l'incapacité de créer une machine capable de raisonner et d'agir indépendamment, mais le progrès de la science a rendu possible le développement de nombreux appareils pouvant remplacer ou imiter le fonctionnement de certaines parties du corps humain.

L'intérêt de la Vision Par Ordinateur est de simuler diverses fonctionnalités de la vision humaine telles que la reconnaissance des objets présents dans une scène observée, l'estimation de leur distance, de leur forme, de leur vitesse. Pour accomplir ces taches, la vision humaine emploie des mécanismes très variés tels que la vision stéréoscopique, l'analyse des perspectives ou des variations de luminosité, ou encore des connaissances acquises sur la structure de la scène.

L'analyse simultanée de toute cette information est très complexe. C'est pourquoi la majorité

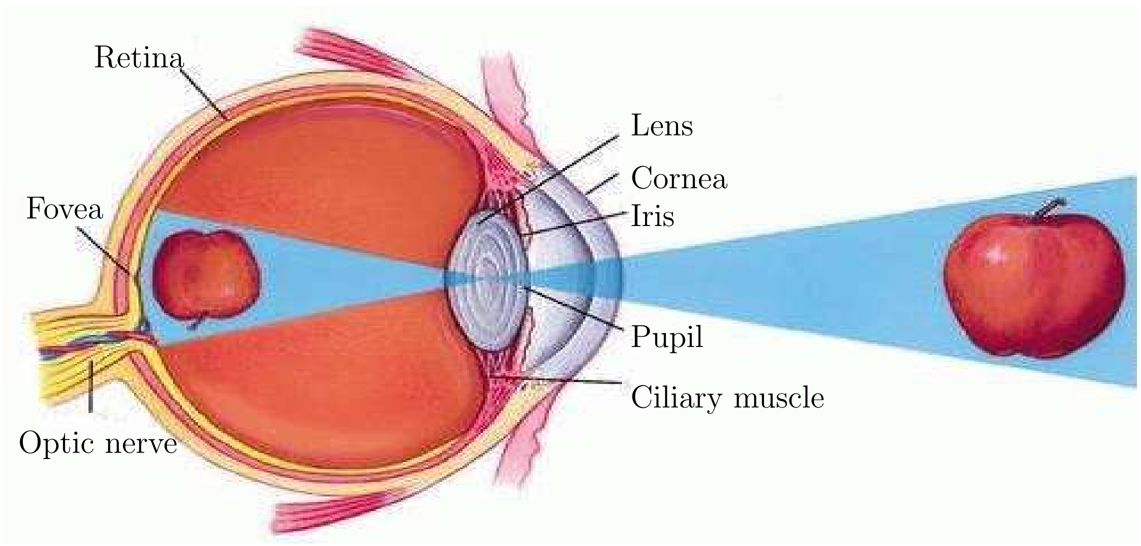


Figure 1: Principe de la création d'image par l'oeil.

des algorithmes de Vision Par Ordinateur sont basés seulement sur une partie de l'information accessible. Dans cette thèse nous nous intéressons aux méthodes combinant l'utilisation des connaissances de la structure de la scène avec l'information contenue dans les images en vue d'obtenir un modèle tridimensionnel de la scène. Afin d'expliquer comment une structure tridimensionnelle peut être extraite à partir d'images bidimensionnelles et pourquoi des informations supplémentaires sur la scène peuvent être utiles dans ce processus, résumons d'abord quels sont les instruments basiques de la Vision Par Ordinateur et comparons les avec les éléments de la vision humaine.

Naturellement, l'outil de base de la vision humaine est un oeil. Biologiquement, sa structure est extrêmement complexe. Pourtant, le principe de la création d'une image est très simple. Comme montre la figure 12, les rayons lumineux sont focalisés par la cornée et le cristallin sur la rétine. Cette image est ensuite transmise au cerveau.

En Vision Par Ordinateur, l'équivalent de l'oeil est la caméra (figure 13). Le modèle sténopé de la caméra, utilisé dans cette thèse, est similaire au modèle de l'oeil décrit ci-dessus. Les rayons lumineux sont focalisés dans le centre de projection sur un plan image. C'est un modèle très simplifié, qui ne prend pas en compte les effets non-linéaires qui se produisent en réalité, tels que la distorsion radiale. Cependant, ce modèle paraît être un bon compromis entre la simplicité et la précision de modélisation des processus de la création d'images.

Les images créées par une seul caméra sont bidimensionnelles et, sans information complémentaire, ne permettent pas d'estimer la nature tridimensionnelle de la scène (figure 14–(a)). Dans le système de la vision humaine, les images créées dans chacun des yeux sont transmises au cerveau. Ce dernier combine ces deux images en analysant leurs similarités et leurs différences. Le résultat est une image *stéréoscopique* (du grec *stereos*: solide).

Plus précisément, étant donné le centre de projection d'une image, il est possible de calculer le rayon de projection passant par le centre de projection et le point image. Des rayons de projection correspondants à un point dans l'espace se coupent en ce même point, définissant ainsi sa position. Par conséquent, un système stéréoscopique est basé sur sa capacité à mettre en correspondance les éléments différentes images et sa connaissance de la géométrie des capteurs. Chez lez hommes dès la naissance la partie du cerveau responsable du traitement des images est entraînée à recevoir et interpréter les données accumulées par la rétine. Nous pensons rarement à la complicité de ce processus. Pourtant, un bébé n'est capable de suivre des objets en mouvement que plusieurs jours après sa naissance. L'établissement de toutes les connections neurales de la rétine dure près de quatre ans. Le résultat finale est notre système de vision constitué par nos yeux dont la géométrie est parfaitement comprise et par la partie de cerveau associée, permettant la perception et analyse du monde extérieur.

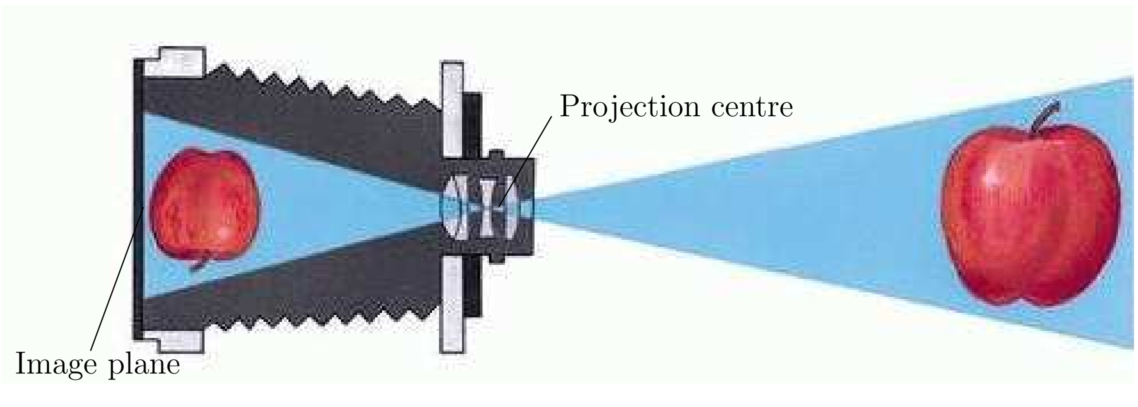


Figure 2: Principe de la création d'image par un caméra.

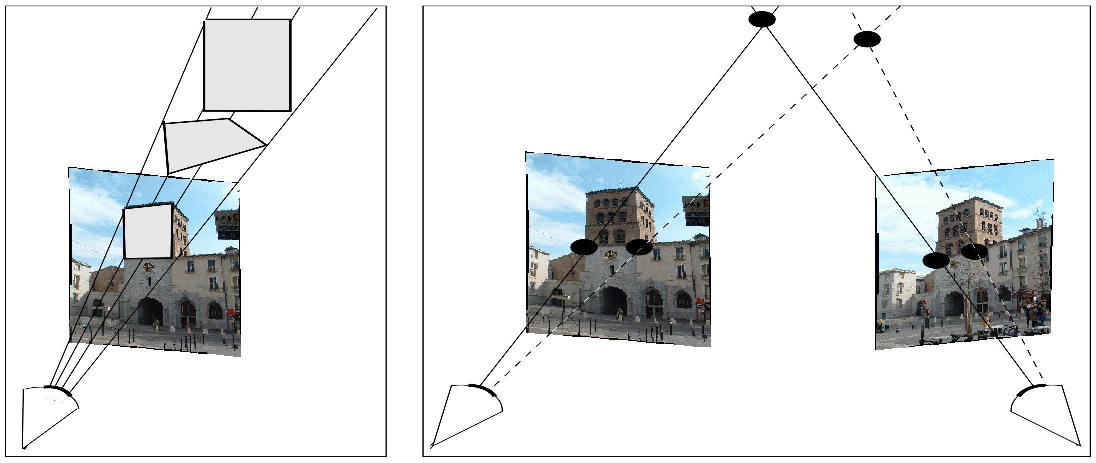

(a) (b)

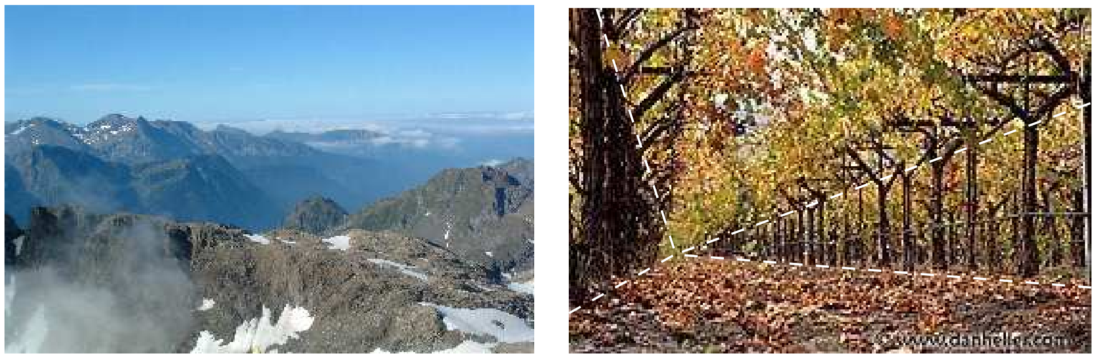
Figure 3: (a) La vision monoculaire. L'information dans l'image n'est pas suffisante pour estimer des profondeurs de points sur des rayons de projection. (b) La vision stéréoscopique. Etant donné deux rayons de projection d'un même point, il est possible d'estimer sa position dans l'espace.

Considérons maintenant un système d'acquisition composé de caméras projectives. Les premiers indices sur la structure tridimensionnelle de la scène sont donnés, en analogie avec la vision humaine, par une étude de correspondances entre les images et les simples propriétés des caméras sténopés. En effet, tous les rayons de projection d'une caméra se croisent en son centre de projection (figure 14–(b)). De plus, tous les rayons de projection d'un point se croisent en ce point. Ces observations sont suffisantes pour estimer les propriétés importantes d'une scène tridimensionnelle, sous la condition cependant que les points présents dans plusieurs images soient mis en correspondance. En utilisant des séquences d'images suffisamment denses, il est donc possible d'estimer ces correspondances et de calculer ainsi automatiquement le modèle tridimensionnel. Cependant, sans informations supplémentaires, la structure estimée de cette façon n'est définie qu'à une transformation projective près. En conséquence, la solution du problème de reconstruction est constituée d'une famille de modèles, dont chacun peut être créé à partir de l'autre par une transformation projective, i.e. transformation qui préserve la colinéarité. La reconstruction projective a de nombreuses applications en robotique. Cependant, elle ne donne pas accès aux propriétés *euclidiennes*, telles que distances et angles, qui sont habituellement utilisées pour décrire les objets.

Quels mécanismes supplémentaires sont donc utilisés par le système de la vision humain pour estimer correctement les distances et les angles même à partir d'une seule image?

Comme nous l'avons déjà mentionné, la géométrie du système composé d'une paire d'yeux est parfaitement connue par la partie du cerveau associée. Dans les systèmes calibrés, il est possible d'établir des propriétés euclidiennes du modèle. En Vision Par Ordinateur cela correspond à la situation où l'ensemble des caméras est calibré, i.e. leurs paramètres intrinsèques et poses sont connus. Malheureusement, les applications de systèmes de reconstruction basés sur des ensembles précalibrés de caméras sont limitées. Le processus d'acquisition d'images est très laborieux et doit être préparé à l'avance. En conséquence, ils ne permettent pas de reconstruire des modèles à partir d'images arbitraires, telles que les photos d'arches ou amateur. Une solution plus intéressante est d'autocalibrer une séquence d'images en utilisant les correspondances entre elles ainsi que l'information sur certains paramètres de caméras ou dee leur mouvement. Malheureusement ce genre d'approche nécessite une séquence dense d'images alors qu'un être humain est capable de décrire les propriétés tridimensionnelles d'une scène à partir d'une seule photo.

Une des raisons pour lesquelles notre système de vision est si difficile à imiter est le fait, que grâce à l'observation et l'apprentissage permanent, nous sommes capable d'extraire et d'interpréter de nombreuses informations supplémentaires servant à estimer les propriétés tridimensionnelles des scènes observées.

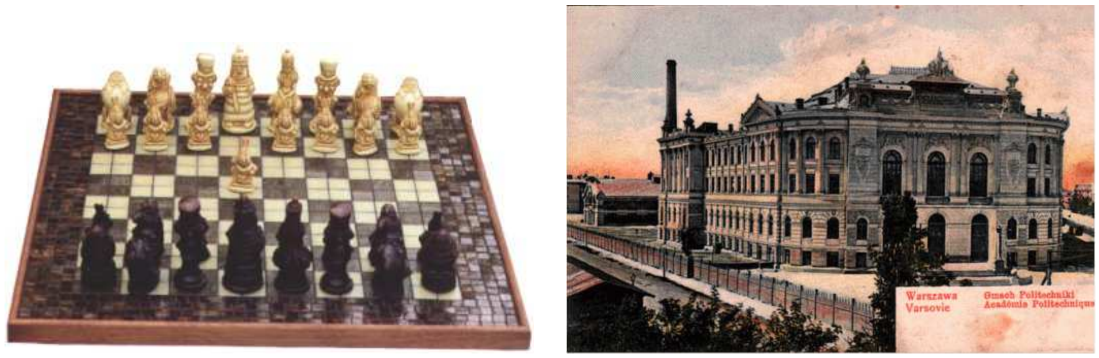


(a) (b)

Figure 4: Un exemple d'effets utiles pour estimer la nature tridimensionnelle de la scène; (a) Des variations de couleurs dues aux effets atmosphériques et changements d'illumination peuvent être utilisés pour estimer des distances et formes des objets; (b) L'effet perspectif. Des images de droites parallèles s'intersectent en un seul point de l'image.

Considérons par exemple la figure 15–(a). L'effet atmosphérique change les couleurs des montagnes selon leur distance par rapport au point de vue. Le changement d'éclairage de la surface des montagnes permet d'estimer leurs formes. Un autre indice est donné par l'effet perspective propre au modèle sténopé de caméra (figure 15–(b)). Toutes les images de droites parallèles dans la scène s'intersèctent dans un seul point de l'image, permettant d'estimer les relations spatiales entre les objets.

Plusieurs choses nous permettent d'interpréter rapidement des informations très variées : nos capacités à synthétiser rapidement les données provenant de nombreuses sources et celles à utiliser des connaissances sur des objets communs dans notre environnement acquis auparavant.



(a) (b)

Figure 5: L'information contenue dans une seule image n'est pas suffisante pour estimer les propriétés telles que l'orthogonalité ou la symétrie. La connaissance des propriétés géométriques communes des objets peut être utile dans la résolution des ambiguïtés.

Considérons par exemple la figure 16–(a). Ayant reconnu que l'objet au centre d'image est un tableau d'échecs, il est naturel de faire l'hypothèse que des polygones blancs et marrons sont carrés. Analogiquement, regardant l'image 16–(b) il serait intuitif de dire que la façade du bâtiment sur la photo est symétrique, avec des fenêtres distribuées régulièrement, etc.

Pourtant, l'information explicite délivrée par l'image est minime dans les deux cas. Des occultations, des reflectances ainsi que la mauvaise qualité des images, font que la détection automatique des éléments importants du modèle est très difficile. En outre, une seule image non-calibrée n'est pas suffisante pour établir des propriétés du modèle telles que les angles droits, le parallélisme ou la symétrie. En regardant les images dans la figure 16 nous devinons ces propriétés grâce aux connaissances que nous avons sur les propriétés des objets souvent présents dans notre entourage. Ce type d'information n'est pas toujours fiable, mais joue un rôle très important dans la façon que nous avons d'interpréter les images.

On peut constater que les connaissances sur les caméras et les propriétés euclidiennes de la scène sont complémentaires, et les images peuvent être utilisées comme une liaison projective, permettant d'exprimer des dépendances mutuelles entre ces deux entités. En effet, un ensemble d'images calibrées permet de reconstruire un modèle euclidien de la scène. D'un autre côté, des informations sur la structure euclidienne de la scène permettent de calibrer les caméras. Finalement, certaines connaissances sur les caméras et sur la structure euclidienne de la scène peuvent être utilisées simultanément pour calculer tous les paramètres des caméras et du modèle à partir de très peu d'images (voire une seule image).

Introduction de l'information sur la scène soulève plusieurs difficultés. D'abord, l'apprentissage de propriétés des objets et l'utilisation de ces connaissances sont des processus très compliqués. Ils nécessitent des capacités d'analyser la scène sur le niveau de détail approprié, de sélectionner des éléments pertinents de la scène et de les comparer avec l'information acquise auparavant. Deuxièmement, la fusion de toutes les données dans un modèle cohérent nécessite des méthodes de représentation et de synthèse de données de types très variés et des moyennes de détection, si les données sont suffisantes et cohérentes.

L'énoncé du problème

Dans cette thèse nous nous intéressons à la reconstruction euclidienne des environnements urbains à partir de petits ensembles d'images [1] en utilisant des combinaisons d'informations sur les paramètres des caméras et de la scène. Puisque la détection des éléments pertinents dans les images est un problème très complexe, l'entrée des nos méthodes est interactive, ce qui permet d'utiliser l'expérience de l'utilisateur. Parmi de nombreuses propriétés relevant la nature 3D de la scène, nous nous concentrons sur l'exploitation des informations géométriques, telles que le parallélisme, les angles ou les distances. Ces propriétés sont bien adaptées à la description de scènes urbaines et sont relativement faciles à introduire par l'utilisateur. L'introduction de contraintes géométriques dans la modélisation 3D réduit le nombre de configurations singulières pour le calibrage ainsi que pour la reconstruction. Ceci permet la création de modèles tridimensionnels à partir de très peu d'images, même en présence des nombreuses occultations.

En conséquence une scène peut être reconstruite même à partir d'une seule image. Par exemple, une ancienne carte postale de l'Ecole Polytechnique de Varsovie (figure 16–(b)) est suffisante pour reconstruire le modèle dans la figure 17. Un autre exemple est présenté dans la figure 20. Deux images trouvées sur Internet (Page Web de M. Kevin Quick, un photographe amateur), ont été utilisées pour la reconstruction d'un modèle presque complet de l'église de Chetwode. Ceci montre l'utilité d'un système basé sur des contraintes pour la conservation du patrimoine. En effet, un tel système permet de mesurer et de visualiser des bâtiments anciens, même s'il n'existent plus.

[1] des algorithmes proposés ont été validés sur des ensembles contenants de 1 à 15 images

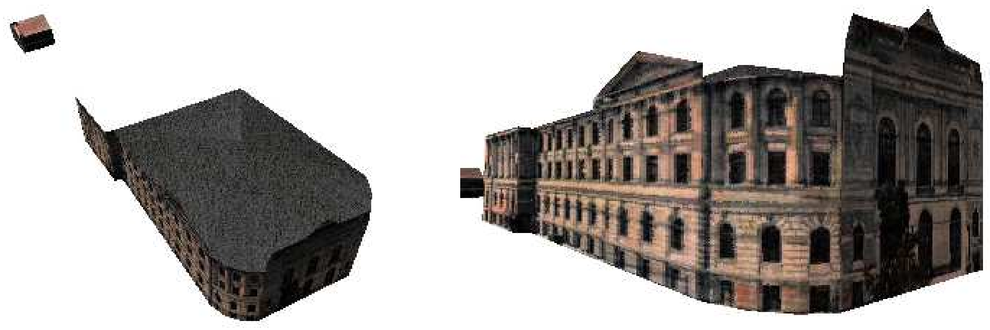


Figure 6: Un modèle reconstruit à partir d'une photo d'archive sur la figure 16–(b).

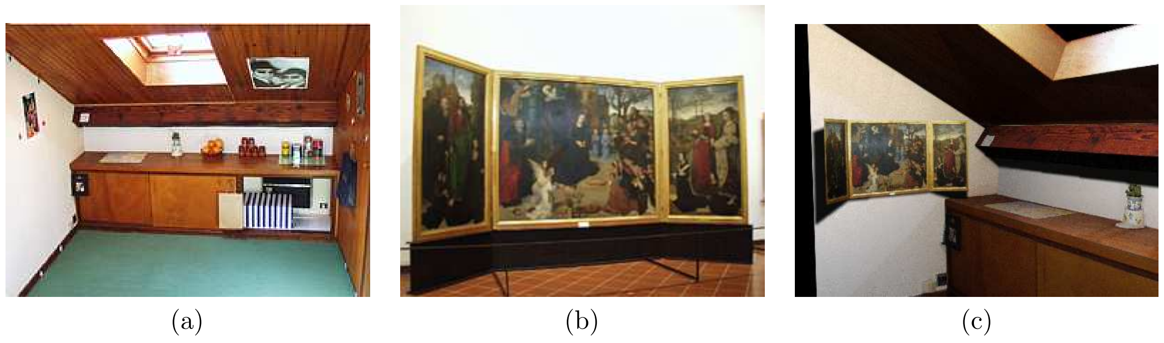


Figure 7: Reconstruction à partir d'une seule image: combinaison de modèles; (a) Une photo originale d'une chambre; (b) Une photo d'une peinture (Uffizi Gallery, Florence); (c) Combinaison de modèles construits à partir des images (a) et (b).

Une autre application directe d'un tel système est la réalité augmentée. Une fois le modèle de la scène réelle reconstruit, il peut être mélangé avec d'autres modèles d'objets réels ou synthétisés (figure 18), ou encore édités (figure 19). Ceci peut être particulièrement utile pour des applications architecturales. En effet, avant qu'un projet soit accepté, il est intéressant de voir son influence sur l'environnement existant. Actuellement cette influence est visualisée en utilisant des maquettes ou des modèles CAO. La construction de ce genre de modèles est très laborieux et nécessite des informations très précises de la scène. Un système basé sur des images permet de limiter le nombre d'informations nécessaires pour la construction du modèle, en donnant la possibilité d'incorporer des sources traditionnelles d'information, comme les cartes ou les dessins architecturaux (figure 21).

Comme nous l'avons mentionné dans la section précédente, l'introduction des contraintes supplémentaires dans un système de reconstruction à partir des images, nécessite des choix au niveau de :

- la représentation des données d'entrée ;
- des méthodes de synthèse de plusieurs sources de données ;
- des méthodes de détection si l'information sur la scène est cohérente et suffisante pour reconstruire le modèle.

Dans les paragraphes suivants nous décrivons brièvement des solutions possibles pour ces trois

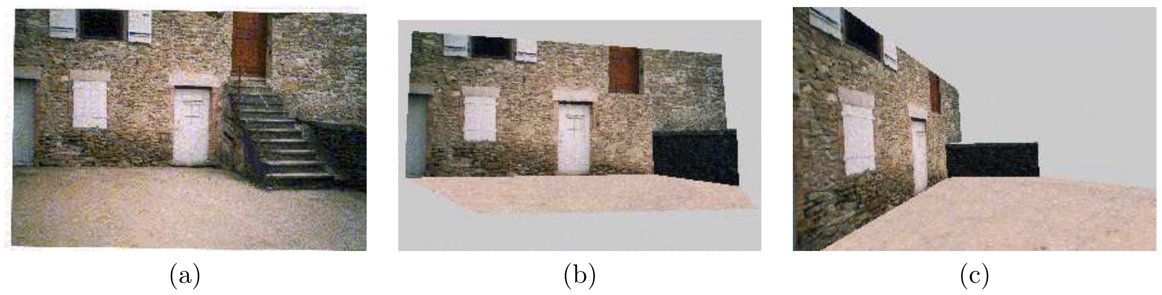


Figure 8: Reconstruction à partir d'une seule image: édition de modèles; (a) Une photo originale; (b),(c) Le modèle après l'édition.

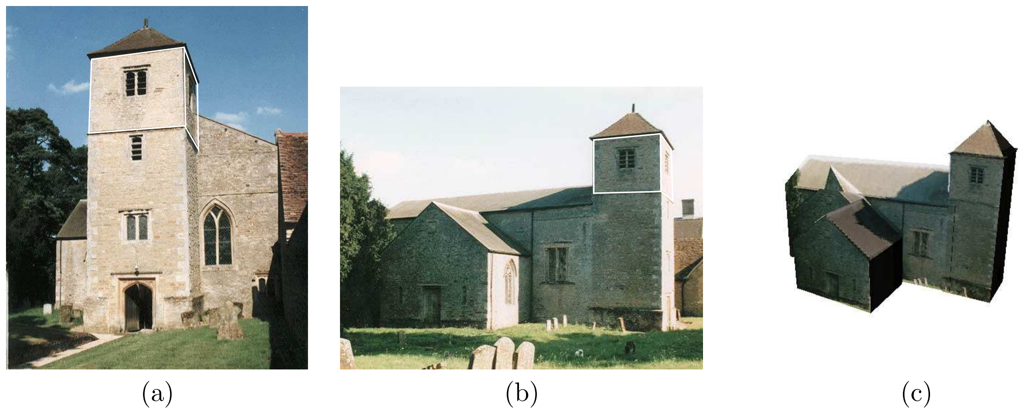


Figure 9: l'Eglise de Chetwode. Reconstruction à partir de 2 images en provenance d'Internet; (a), (b) Des images originales (remerciements à M. Kevin Quick); (c) Le modèle reconstruit.

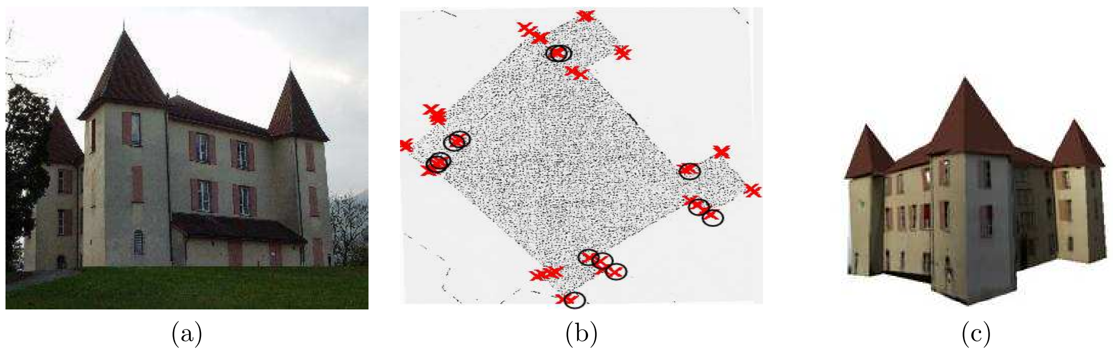


Figure 10: La mairie de Montbonnot. Reconstruction à partir de sept images. (a) Une photo de la séquence utilisée pour la reconstruction; (b) Le plan du château avec des points du modèle surimposés; Cette carte peut être incorporée dans le système interactif de la reconstruction; (c) Le modèle 3D texturé.

problèmes.

Représentation de données. La performance de chaque algorithme est dépendante de la représentation des données d'entrée. Pour la représentation d'informations sur les environnements urbains auxquels nous nous intéressons dans cette thèse, nous pouvons distinguer deux approches principales:

- La représentation de la scène par des primitives simples, telles que points, droites et plans et des relations spatiales telles que le parallélisme, l'orthogonalité, la symétrie ou la distance. Une telle représentation est très flexible. Elle permet le traitement des problèmes de calibrage et de reconstruction à l'aide de simples techniques d'optimisation. Cependant la définition des scènes complexes peut être très laborieuse et peut aboutir à des systèmes de variables et d'équations très complexes. Par exemple, la définition d'un parallélépipède nécessite l'introduction de 8 points, 12 droites, 9 parallélismes et 3 distances.

- La représentation de la scène par des primitives complexes telles que des cubes, prismes ou cylindres ainsi que les relations spatiales telles que l'incidence, le plan commun etc. Une telle représentation réduit la quantité d'interaction et le nombre de paramètres à estimer. Cependant, l'utilisation de primitives complexes nécessite souvent l'utilisation de techniques complexes de résolution de contraintes et d'optimisation.

Le choix d'algorithmes. L'introduction d'une information supplémentaire dans la modélisation 3D à partir d'images peut aboutir à des systèmes complexes de variables et dépendances. Cette complexité augmente avec l'homogénéité des primitives et contraintes contenues dans le système. Il existe plusieurs approches pour la gestion de l'information géométrique de la scène. Premièrement, la modélisation 3D peut être décomposée en plusieurs étapes. Par exemple, les contraintes introduites dans le système peuvent être d'abord utilisées pour calibrer des caméras et ensuite, les caméras calibrées peuvent être utilisées pour la reconstruction. Ceci permet une simplification de problèmes résolus à chaque étape et permet souvent l'utilisation de méthodes linéaires. Cependant, une telle décomposition ne permet pas de traiter toutes les données simultanément et peux aboutir par une propagation systématique d'erreurs. Une autre approche consiste à utiliser des techniques d'optimisation non-linéaire prenant en compte toutes les informations simultanément. Malheureusement, la traduction de dépendances tridimensionnelles en un ensemble d'équations algébriques mène à des systèmes très complexes, qui sont difficiles à résoudre à l'aide de méthodes d'optimisation classiques.

Vérification de données d'entrée. Pour assurer un fonctionnement correct des algorithmes, il est nécessaire de vérifier si les données d'entrée permettent de construire un modèle unique et correct.

Premièrement il est nécessaire de vérifier si les informations introduites dans le système sont *suffisantes* pour calculer une solution unique pour des problèmes de calibrage et de reconstruction. L'information n'est pas suffisante quand le nombre de contraintes indépendantes est inférieur au nombre de paramètres libres du problème. Ceci peut être du à une mauvaise définition de la scène par l'utilisateur ou par une configuration singulière des objets et des caméras.

Deuxièmement, il est nécessaire de vérifier si l'information introduite dans le système est *cohérente*, c'est-à-dire si les contraintes introduites ne sont pas contradictoires entre elles.

Contributions

Dans cette thèse nous proposons plusieurs algorithmes qui permettent le passage d'un ensemble des photos non-calibrées vers un modèle tridimensionnel de la scène. Tous ces algorithmes sont basés sur une exploitation des connaissances des propriétés géométriques de la scène. La figure 22 donne une intuition sur l'entrée et la sortie du système.

L'entrée du système contient:

- un ensemble d'images non-calibrées et éventuellement une information sur certains paramètres des caméras;
- un ensemble de primitives marquées et une mise en correspondance par l'utilisateur;
- un ensemble de relation géométrique entre les primitives.

La sortie du système contient:

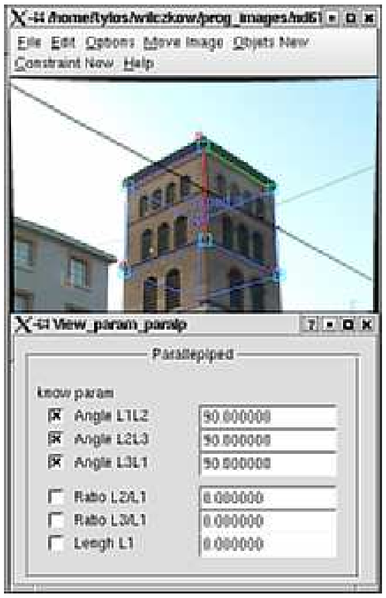
- le calibrage et les poses des caméras;
- un modèle euclidien de la scène.

Dans cette thèse nous proposons des algorithmes qui permettent de calculer une solution pour trois étapes de modélisation tridimensionnelle.

D'abord, nous présentons une approche de calibrage basée sur des parallélépipèdes, introduit initialement dans [Wilczkowiak et al., 2001, 2002, 2003c]. Les parallélépipèdes encodent naturellement des propriétés géométriques de la scène et sont très souvent présentes dans les environnements. Nous introduisons une notion de dualité entre les paramètres des caméras et des parallélépipèdes. Sur le plan théorique, une des conséquences de cette dualité est le fait que le problème du calibrage peut être formulé dans les termes d'un cube canonique plutôt que dans les termes d'une conique absolue. Sur le plan pratique, cette dualité permet de proposer des algorithmes qui regroupent les avantages d'une représentations de la scène par des primitives simples et complexes. En effet, grâce à l'utilisation de parallélépipèdes l'interaction de l'utilisateur est réduite. Grâce à la notion de dualité, il est possible d'exploiter toutes les informations sur des parallélépipèdes et des caméras simultanément en utilisant des algorithmes linéaires. La structure obtenue de manière linéaire peut être ensuite raffinée dans un processus non- linéaire. De plus, grâce à l'utilisation de primitives complexes, le nombre de paramètres est réduit.

Notre approche pour le calibrage est accompagnée par une étude détaillée des configurations singulières. Nous proposons aussi une approche originale pour la détection de variables sous-contraintes du système.

Une fois les caméras calibrées, il est possible de reconstruire la structure euclidienne de la scène. Pour assurer la flexibilité de la définition de la scène, nous proposons de représenter le modèle par des primitives simples, telles que des points, des droites et des plans, ainsi que par des contraintes géométriques entre eux. Nous limitons l'ensemble des contraintes géométriques utilisées aux contraintes qui peuvent être exprimées par des relations bilinéaires, tels que le parallélisme, l'orthogonalité ou l'incidence. Grâce à cette limitation, il est possible de proposer une approche itérative, basée sur des méthodes d'algèbre linéaire. En effet, quand les coordonnées d'un des objets liés à une contrainte bilinéaire sont connues, il est possible de formuler des équations linéaires sur le deuxième objet et d'estimer sa position. De cette façon, l'information peut être propagée jusqu'a l'estimation de tous les objets présents dans la scène. La méthode de la reconstruction décrite dans ce document a été présentée initialement dans [Wilczkowiak et al., 2003a]. L'algorithme principal derrière cette méthode est une approche pratique pour la détection de variables suffisamment contraintes dans des systèmes linéaires sous-contraints. Cette approche est basée sur la Décomposition

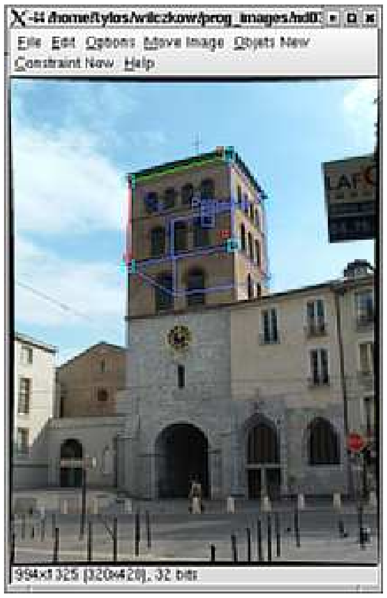

X-¤ /home/tylos/wilczkow/prog_images/nd61
File Edit Options Move Image Objets New
Constraint New Help
X-¤ View_param_paralp

Parallepiped

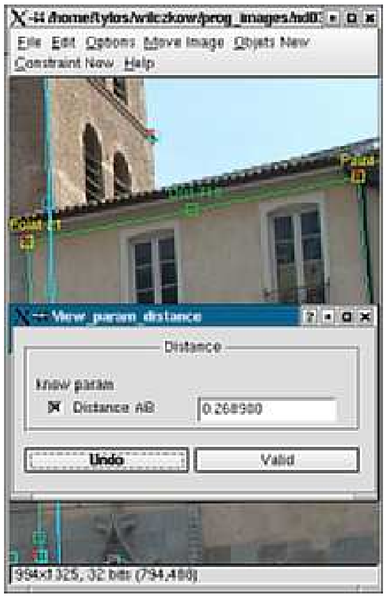
know param

- [✓] Angle L1L2 — 90.000000
- [✓] Angle L2L3 — 90.000000
- [✓] Angle L3L1 — 90.000000
- [ ] Ratio L2/L1 — 0.000000
- [ ] Ratio L3/L1 — 0.000000
- [ ] Length L1 — 0.000000

(a)


X-¤ /home/tylos/wilczkow/prog_images/nd0
File Edit Options Move Image Objets New
Constraint New Help
994x1325 (320x428), 32 bits

(b)

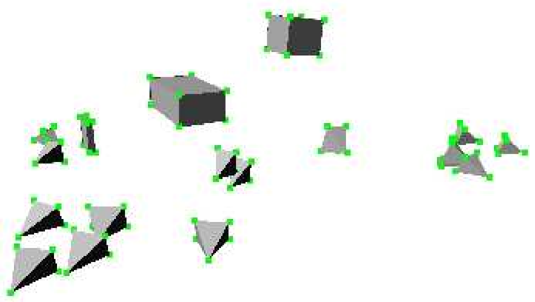

X-¤ /home/tylos/wilczkow/prog_images/nd0
File Edit Options Move Image Objets New
Constraint New Help
Point
X-- View_param_distance
Undo
Valid
994x1325, 32 bits (794,488)

Distance

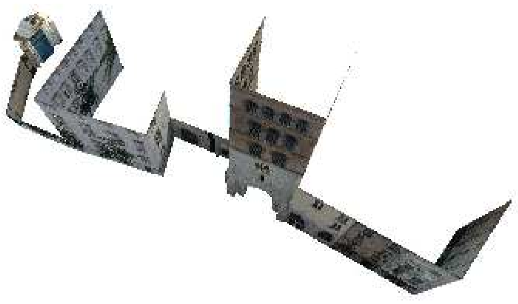
know param

- [✓] Distance AB — 0.268900

(c)



(d)



(e)

Figure 11: Exemples d'entrée et de sortie du système; L'entrée: (a),(b) Des images de primitives sont mises en correspondance entre les images; (a)(c) Des propriétés euclidiennes telles que les paramètres de primitives ou les distances sont introduites de manière interactive. La sortie: (d) Une étape intermédiaire : une estimation des poses des caméras et une reconstruction partielle de la scène; (e) Un modèle complet de la scène.

en Valeurs Singulières de la matrice formée par des contraintes et peut être appliquée directement à tous les systèmes linéaires. Dans le système de reconstruction proposée dans cette thèse, nous utilisons cette approche pour détecter les objets dont la position peut être estimée uniquement à partir de contraintes géométriques définis dans le système. Une telle détection est nécessaire pour un fonctionnement correct de la propagation d'information.

La méthode de reconstruction proposée est très rapide et donne une très bonne solution initiale pour un éventuel raffinement non-linéaire.

Finalement, nous décrivons une approche pour la satisfaction de contraintes géométriques dans un processus non-linéaire. Certaines problématiques liées à cette approche ont été publiés dans [Wilczkowiak et al., 2003d; Trombettoni and Wilczkowiak, 2003]. Contrairement à la plupart des approches existantes, notre méthode permet de gérer de grands ensembles de objets et de contraindre des types très variées. En générale, la représentation géométrique d'un problème géométrique tridimensionnelle résulte en un systèmes de variables et d'équations très complexes et difficile à résoudre à l'aide de méthodes numériques classiques. Nous proposons une approche différente, basée sur la Propagation Générale de Degrés de Liberté (General Propagation of Degrees of Freedom (GPDOF)), dérivée de celle connue dans la communauté de Programmation par Contraintes. Notre approche permet de transformer un système représenté par un ensemble de variables et d'équations en un système représenté par un ensemble de variables libres (*input parameters*) et une séquence de routines (*plan*), dont l'exécution mène à un modèle satisfaisant de contraintes. Une fois que le modèle est représenté par un ensemble des variables libres, il est possible d'utiliser des méthodes d'optimisation classiques pour estimer un modèle conformant aux images. Contrairement aux approches existantes, notre méthode permet un paramétrage du modèle dans un temps polynomial et assure de trouver une solution satisfaisant toutes les contraintes si seulement une telle solution existe. Nous montrons que le paramétrage de la scène trouvé par notre méthode n'est pas minimal, mais que ce n'est pas contradictoire avec le fait que toutes les contraintes soient respectées. Nous montrons aussi, comment un paramétrage minimal pourra être calculé.

Le problème majeur des systèmes satisfaisant des contraintes exactement est la gestion des contraintes redondantes. Ceci est un problème encore ouvert. Cependant, notre algorithme permet de séparer tous les objets et contraintes inclus dans un sous-système contenant des contraintes redondantes. Nous proposons aussi des algorithmes pour enlever certains types d'ensembles de contraintes redondantes qui sont souvent présentes dans environnements urbains.

Structure de la thèse

Cette thèse est composée de trois parties. La première partie introduit des conceptes basiques de la géométrie projective et la représentation des objets et contraintes utilisés dans cette thèse (chapitre 1) ainsi que l'état de l'art en utilisation de contraintes géométriques en Vision par Ordinateur (chapitre 2).

La seconde partie introduit des approches linéaires pour la modélisation 3D à l'aide de contraintes géométriques. D'abord nous décrivons une approche pour le calibrage à l'aide de parallélépipèdes dans les chapitres 3-5. Dans le chapitre 7 nous proposons une approche multi-linéaire pour la reconstruction de la scène. Des résultats de ces méthodes sont présentés dans le chapitre 7.

La troisième partie décrit une approche pour un paramétrage quasi-minimal de la scène et son application au raffinement non-linéaire du modèle basée sur la Propagation Générale de Degrés de Liberté GPDOF. Des chapitres 8- 10 présentent l'algorithme proposé. Chapitre 11 décrit des expériences validant notre méthode. Dans le chapitre 12 nous comparons GPDOF aux autres systèmes de décomposition développés dans le domaine de la Programmation par Contraintes.

Introduction

Context and Motivation

The inspiration for most of the technical inventions lies in nature. We can understand and imitate many natural phenomena, but there are still fields where achievements of science are modest compared to their natural models. This is the case which has inspired generations of scientists: making a creature in our image and similitude. New perspectives to approach this challenge have appeared with the development of computers. We are still far from creating a machine which can think and act independently, but progress in science has led to the development of many devices which can replace or imitate the functionality of separate parts of our bodies. Sensors analyzing smells, noises, or images are often exploited in different contexts to replace our imperfect senses.

The interest of Computer Vision is the simulation of various mechanisms of human sight such as: recognition of objects present in the observed scene, estimation of their distances, shapes, or velocities. To perform these tasks, the human vision system exploits very different mechanisms, starting from stereo information, through the analysis of perspective or shading effects and finishing with using the knowledge of the scene acquired previously.

Using all of this information simultaneously is very complicated and in Computer Vision algorithms only a subset of this information is usually treated at once. In this thesis we are mainly

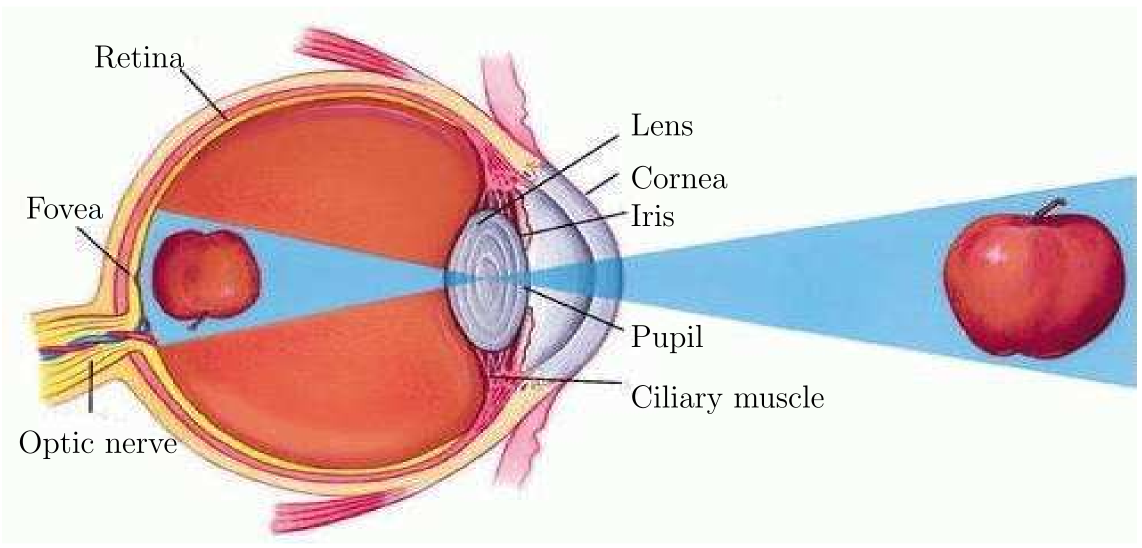


Figure 12: The principle of the image creation in human eye.

interested in methods allowing the use of prior knowledge of the scene together with the image information in order to obtain a three-dimensional scene model. To give an idea of how 3D scene structure can be extracted from 2D images and why incorporation of prior information can be very useful in this process let us first review basic tools used in Computer Vision and compare them to the elements of the human vision system.

Naturally, the basic tool of the human vision system is the eye. Biologically, its structure is extremely complicated. For example, in spite of many efforts, researchers are still not able to create an artificial retina, the most important part of the eye. However, the principle of image creation is very simple. As shown in figure 12, light rays are focused through the transparent cornea and lens upon the retina. The central point for image focus in the retina is the fovea. Here a maximally focused image initiates resolution of the finest details and direct transmission of those details to the brain for higher operations needed for perception.

An equivalent to an eye used in Computer Vision is a camera (see figure 13). The pinhole camera model, used in this thesis, is very similar to the model of the human eye described above. The light rays are focused in the projection center upon the image plane. This model is simplified and do not model non-linear effects occurring in real cameras, such as radial distortion. It seems to be however a good compromise between the simplicity and the precision of the image creation process modeling.

Images created by a single eye or camera are only two-dimensional, and without any further information do not allow the estimation of a three-dimensional nature of the scene (see figure 14–(a)). However, in the human vision system images captured by each eye are analyzed subsequently by the brain, which combines the two images by matching up the similarities and adding in the small differences, forming *stereoscopic* images (from ancient Greek *stereos*: solid).

More precisely, when the image projection center is known, it is possible to lead a projection ray joining the projection center and the image point. Projection rays corresponding to one spatial point must intersect at this point, defining its position. Thus, the ability to match features between images and understanding the geometry of image sensors lies at the ground of stereoscopic vision systems. Starting from birth, the part of the brain responsible for image processing is trained to receive and interpret the visual data received in the retina. We rarely think about the complexity of this process. It takes several weeks until a child is able to follow moving objects. It takes up to four years before all the necessary neuronal connections are established in the retina. The final result is our vision system, consisting of a pair of eyes, whose geometry is well understood by the associated part of the brain, allowing for the automatic and efficient perception of the surrounding world.

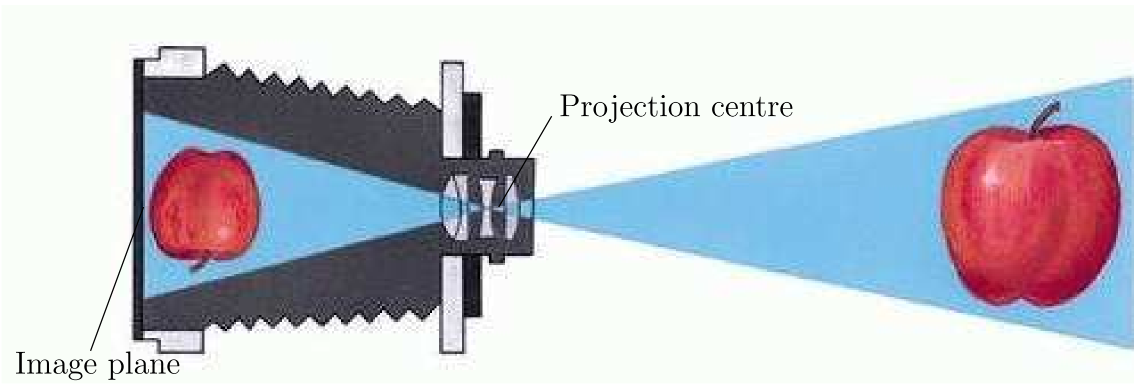
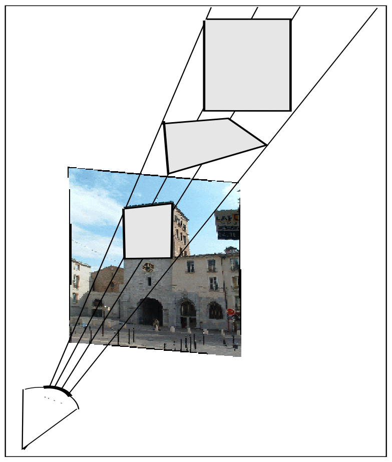


Figure 13: The principle of the image creation in camera.


(a)

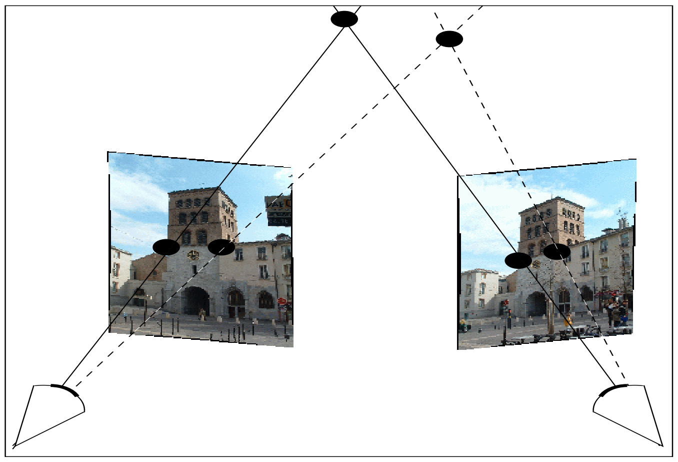

(b)

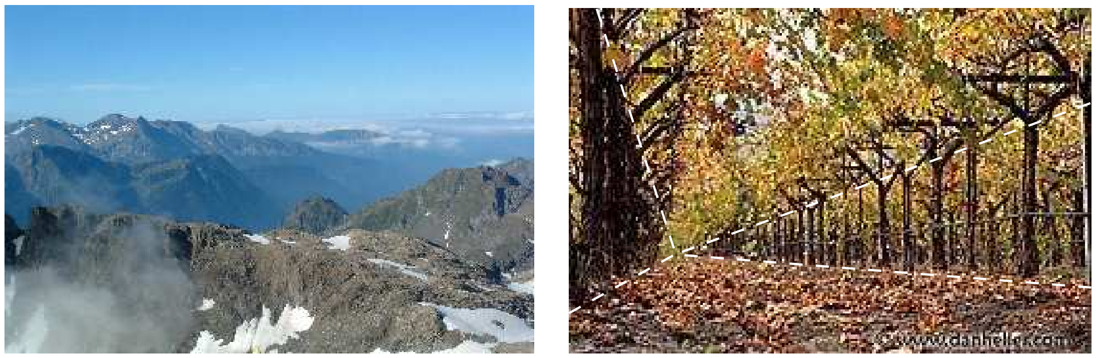
Figure 14: (a) Mono vision. The image information is not sufficient to recover depths of points along projection rays. (b) Stereo vision. When two projection rays of a point are given, it is possible to compute its position in space.

Let us consider now an acquisition system composed of projective cameras. First, important clues about the 3D scene structure are given, as in stereo human vision, by a study of correspondences between the image points and simple dependencies engendered by basic properties of pinhole camera projection. Let us consider again figure 14–(b). All of the projection rays of the cameras meet at the projection center. On the other hand, all of the projection rays of a point meet at one point. These observations are sufficient to recover important properties of a 3D scene, provided that the inter-image point correspondences are known. Using sufficiently dense image sequences it is possible to compute the inter-image correspondences and thus the 3D model automatically. However, without any additional information, such a 3D structure is computed only up to a projective transformation. This means that any transformation of the model preserving collinearity satisfies all of the correspondence information as well. Projective reconstruction has multiple applications in machine vision and robotics. However, it does not reveal *euclidean* properties such as angles and distances which are most commonly used for the description of objects.

Thus what enables us to correctly estimate distances and angles not only when watching objects with both eyes, but even from single photos, where no stereo information is given?

First, as already mentioned, the geometry of our eyes is taken well into account by the associated part of the brain. Indeed, during our whole life their relative position does not change, and focusing is steered by the brain. When sensors are calibrated, it is possible to uniquely define the projection rays corresponding to scene points and recover the euclidean scene properties. In Computer Vision this corresponds to the situation where cameras are calibrated, i.e. their intrinsic parameters and poses are known. However, reconstruction using image sequences taken with precalibrated sets of cameras has very limited applications. The image acquisition is laborious and does not allow for reconstruction from arbitrary images, such as archive or amateur photos. Much more interesting solutions consist of the *self-calibration* of cameras, i.e. camera calibration using the image correspondences and information about only *some* camera parameters or special movements. But again, such approaches for the estimation of camera intrinsic parameters and poses require dense sequences of images in general positions, while a human can well describe the 3D character of a scene from a single photo.

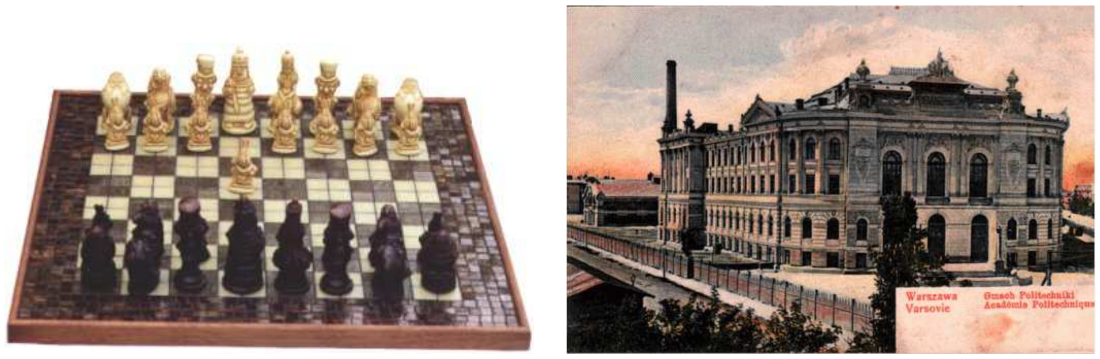
One of the most important reasons why our vision system is so difficult to imitate is the fact, that thanks to continuous observation and learning we are able to extract and interpret various complementary information revealing the 3D character of the scene. Consider for example



(a) (b)

Figure 15: Example of image information useful to recover three-dimensional nature of the scene; (a) Variation of colors due to the atmospheric and shading effects help in estimation of distances and shapes of objects. (b) Perspective effect. Parallel lines meet at one image point.

figure 15–(a). The atmospheric effect changes the colors of the mountains depending on their distance. The shading of mountain surfaces and contours allows the estimation of their shape. Another helpful hint is given by the perspective effect proper to the pinhole projection model. This is illustrated in figure 15–(b). All of the parallel lines meet at one image point. As a result, the further the object is away, the smaller it seems to be in the image, revealing important spatial properties of the scene. What allows us to rapidly interpret very different visual information is the fact that we cannot only synthesize efficiently various types of data, but that we are also able to exploit previous experience and learned knowledge about properties of objects present in our environment.



(a) (b)

Figure 16: Image information is not sufficient to establish information about orthogonality or symmetry from a single image. Knowledge of common geometrical properties of objects is helpful in solving the ambiguities.

Consider for example figure 16–(a). Since the object in the image is a chessboard, naturally white and brown polygons should be squares. Similarly, analyzing image 16–(b) it is rather instinctive to say that the facade of the building is symmetric, with regularly distributed windows of the same shape, etc.

However, the image information is very poor in both cases. Numerous occlusions, reflectances, and bad image quality make it difficult to locate the important model features automatically. But even if the image data was perfect, a single projection is not sufficient to establish information such as right angles, parallelism or symmetry in the scene. These properties of a scene are guessed by using known common properties of objects from the scene. Such information is not always reliable, but is very important in the process of image interpretation.

Thus camera and scene euclidean properties are complementary and images can be used as a projective link between them, allowing the expression of mutual dependencies. Indeed, a set of calibrated images is sufficient to reconstruct a 3D model. On the other hand, known euclidean scene features allows the calibration of cameras. Finally, partially calibrated cameras and some prior information about the scene can be used together to recover both scene and camera parameters from small image sets, or even a single image.

However, using prior information about the scene reveals several difficulties. Firstly, the process of learning and using acquired knowledge is very complicated. It necessitates the ability to analyze the scene at a correct description level, to select the pertinent features, and to compare it to already acquired information. Secondly, fusion of all the input data into a coherent model requires the ability to describe and synthesize different types of information and to detect if the given information is sufficient and consistent.

Problem Statement

In this thesis we are interested in the euclidean reconstruction of man-made environments from small sets of uncalibrated images[2] using a combination of available information about camera and scene parameters. As the detection of relevant features in images is a very complex problem, we take advantage of human experience and rely on interactive user input. Among various types of properties revealing the 3D nature of the scene, we concentrate on exploiting euclidean information such as parallelism, angles and distances between objects. Euclidean properties are well adapted to describing man-made environments and are relatively easy to provide by the user. Introduction of scene constraints into the model acquisition process reduces the number of singular configurations for calibration and allows for model reconstruction using small sets or even single images despite the presence of numerous occlusions.

This means that a scene can be reconstructed, for instant, even from a single (e.g. archival) image. For example, an old postcard of the main building of Warsaw University of Technology (figure 16–(b)) is sufficient to reconstruct the model in figure 17. Another example is shown in figure 20. Two images found on Mr. Kevin Quick web page of, an amateur English countryside photographer, were used to reconstruct an almost complete model of the Chetwode church. This reveals the utility of a constraint-based system for the conservation of cultural heritage. Indeed, using such a system enables for example the measurement and visualization of buildings, even if they do not exist anymore.

Another straightforward application of such a system is augmented reality. Once real scene models are reconstructed, they can be mixed with other reconstructed or synthesized models (see figure 18), or edited (see figure 19). This can be especially useful in architectural applications. Indeed, before an architectural project can be accepted, its impact on the existent environment is visualized using hand-made or CAD city models. The process of building such models is laborious and requires more prior information than scene reconstruction from images. Moreover, when needed, traditional tools used by architects, such as maps and architectural drawings are easy to incorporate into an interactive reconstruction approach (see figure 21).

[2]proposed algorithms were validated on sets containing from 1 to 15 images

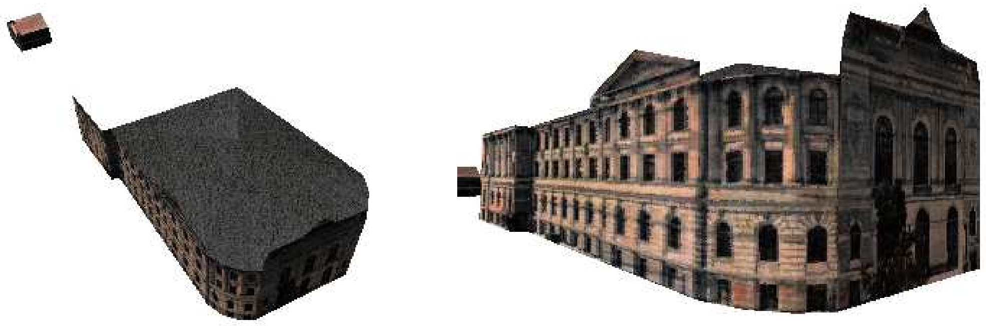


Figure 17: Model reconstructed from archival photo shown in figure 16–(b).

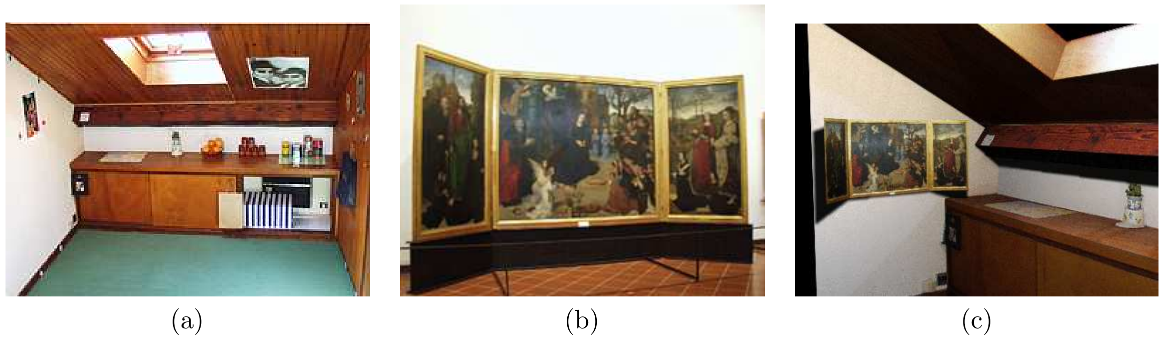


Figure 18: Single image reconstruction: model mixing; (a) Original photo of a room; (b) Photo of a painting in Uffizi Gallery (Florence); (c) Combination of models build from images (a) and (b).

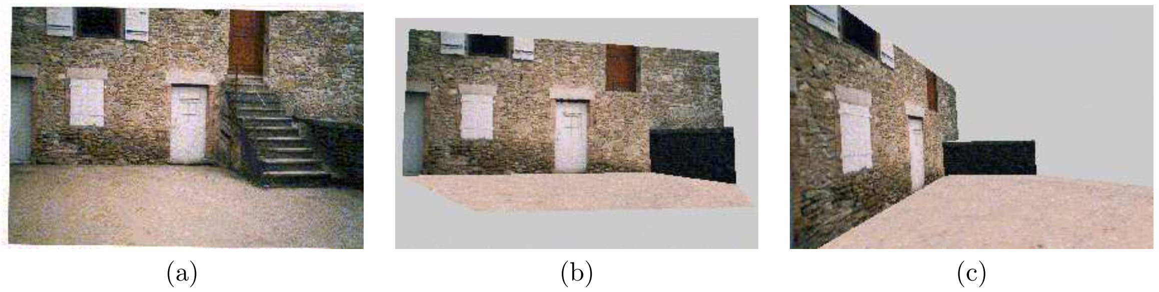


Figure 19: Single image reconstruction: model edition; (a) Original photo; (b),(c) Model after removal of stairs.

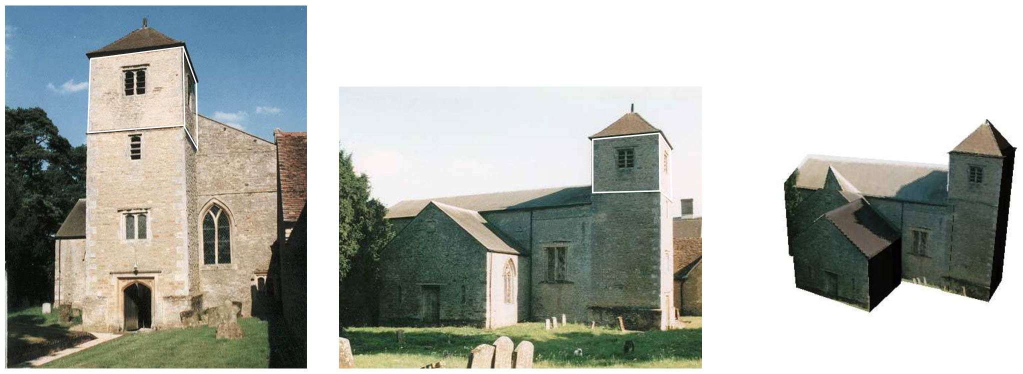

(a) (b) (c)

Figure 20: Chetwode Church. Reconstruction from 2 images found on the Internet; (a), (b) The original images (courtesy of Mr. Kevin Quick); (c) The reconstructed model.

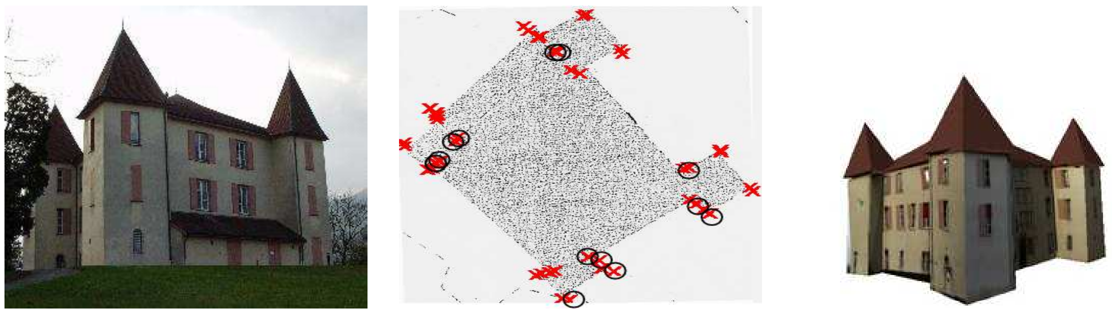

(a) (b) (c)

Figure 21: Montbonnot Townhall. Reconstruction based on seven images of the castle. (a) One of the seven images used for the reconstruction; (b) The castle plane with superimposed model points; Such map can be easily included into a reconstruction system; (c) The textured 3D model.

As mentioned at the end of the previous section, when introducing additional constraints into an image-based 3D model acquisition system it is necessary to make choices concerning how to represent the input data, how to synthesize different types of information and how to detect if the given information is sufficient and consistent. Let us briefly resume important issues related to these problems addressed in this document.

Data representation. Note, that depending on the problem, humans automatically use different descriptions for objects. Similarly, depending on the expected algorithm performance and results, the scene objects will be represented in different ways. For man–made environments, which are of special interest in this thesis, we can distinguish two main trends:

- Representation of the scene using simple primitives, such as points, lines and planes and the spatial relations between them, such as parallelism, orthogonality, symmetry, distances. Such a representation has the advantage of being very flexible. Using a limited set of simple primitives and constraints allows the solution of calibration and reconstruction problems using standard optimization techniques. However, the definition of complex scenes can be laborious. For example, a full definition of a parallelepiped in such a system may require introducing 8 points, 12 lines, 9 parallelism and 3 distance constraints.

- Representation of the scene using complex primitives, such as cubes, prisms, cylinders and the spatial relation between them, such as incidence, common ground plane etc. Using such primitives decreases the amount of necessary user interaction and the number of parameters to estimate. However, using complex structures, usually requires using advanced constraint solving and optimization techniques.

Choice of algorithms. Introducing additional information into the modeling results in a more complicated system of dependencies between the variables of the system. This complexity increases with the number of homogeneous types of objects and constraints added to the system. Several approaches to deal with information about scene geometry are possible. First, the model acquisition process can be decomposed into several stages. For example the available data can first be used to calibrate the cameras and then the calibrated cameras can be used for the reconstruction. This simplifies the problem considered at every stage and enables the use of simple (e.g. linear) methods. However, this does not allow for the simultaneous use of information and can cause the systematic propagation of errors. Another approach consists of using non-linear optimization techniques solving all constraints simultaneously. Unfortunately, translating 3D dependencies into a set of algebraic equations leads to complex equation systems which are difficult to solve using standard optimization methods. To avoid such problems, the scene is sometimes modeled by a parameterized model and the acquisition problem consists only of finding the parameters which define the model conforming to the image data.

Dealing with missing and redundant data. In order to insure a correct solution consideration must be given to the verification of the input data. First, it is necessary to test if the given information is *sufficient* to compute a unique solution for the calibration and reconstruction problem. The data is insufficient when the number of independent constraints is smaller than the number of free model parameters. This can be caused by wrong user input or by a singular configuration of cameras or model features. Consideration must also be given to verify that the input is *consistent*, i.e. the introduced constraints are not contradictory. In order to insure good performance of the algorithms it is necessary to detect such situations.

Contributions

In this thesis we propose several algorithms covering different stages of the reconstruction of the euclidean structure of a scene from a set of uncalibrated images. All these algorithms are based on the exploitation of prior information about geometrical properties in the scene. Figure 22 gives an indication of the input and output of the system.

The input consists of:

- a set of uncalibrated images with possible prior knowledge of certain intrinsic parameters;
- a set of scene primitives depicted and matched over the images;
- a set of geometrical relations between the primitives.

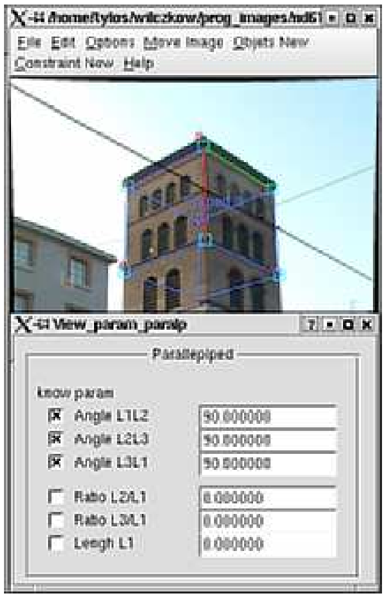
The output of the system consists of:

- calibration and pose estimation for the cameras;
- a euclidean model of the scene.

Let us resume briefly the algorithms proposed in this thesis.

First, we present a parallelepiped-based calibration method introduced in [Wilczkowiak et al., 2001, 2002, 2003c]. Parallelepipeds encode naturally the geometric properties of a scene, are very common in man–made environments, and are easy to depict by the user. We introduce a notion of duality between camera and parallelepiped parameters. On the theoretical level, one of the consequences of this duality is the fact that the calibration problem can be formulated in terms of a canonic cube instead of an absolute conic. On the practical level the duality can be used to propose algorithms joining advantages of simple and complex representations. While input is through complex primitives, the proposed algorithms are linear. The geometry of the cameras and parallelepipeds are computed simultaneously. The obtained structure can be refined in a non-linear optimization step. Then, thanks to the use of parallelepipeds the number of parameters to optimize is reduced. The approach is accompanied by a detailed study of singular configurations for intrinsic parameter estimation. We also propose an original method for the detection of underconstrained features for intrinsic and extrinsic parameter computation.

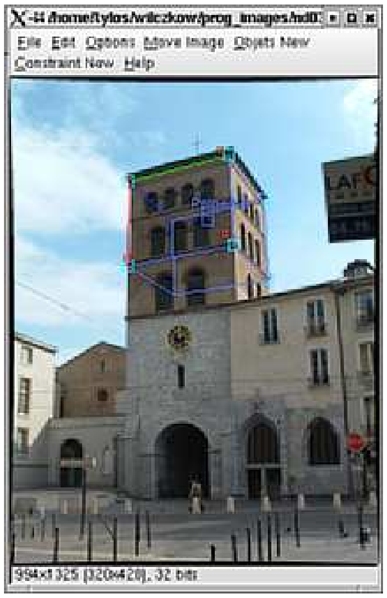
Once the cameras are calibrated, it is possible to reconstruct the euclidean structure of the scene. To insure maximal flexibility in the scene definition for the reconstruction task we model the scene using simple primitives, such as points, lines and planes. Together with geometrical constraints, they can describe most man–made structures. By limiting the set of available constraints to constraints which can be expressed by bilinear equations in terms of related primitives, it is possible to propose an iterative approach based upon linear methods. Indeed, when coordinates of one of the primitives involved in a relation are known, it is possible to express the constraints on the other primitive through linear equations. The computed information can be propagated until the whole scene is reconstructed. The reconstruction method described in this document was presented first in [Wilczkowiak et al., 2003a]. The main algorithm behind this method is a practical approach for the detection of well-constrained variables in linear equation systems. This approach is based on the SVD of the equation system's matrix and can be applied straightforwardly. Its application domain covers in particular all computer vision algorithms based on linear algebra. It can be used for example to detect cameras which are underconstrained for intrinsic parameters or pose estimation in the parallelepiped-based calibration approach described above. In the implemented reconstruction system it enables the detection of features which are sufficiently constrained and, consequently, to propagate the available geometrical information. The proposed reconstruction method is very fast and gives a very good initial solution for a possible non–linear refinement step.


X -¤ /home/tylos/wilczkow/prog_images/nd61
File Edit Options Move Image Objets New
Constraint New Help
X -¤ View_param_paralp
Parallepiped

know param

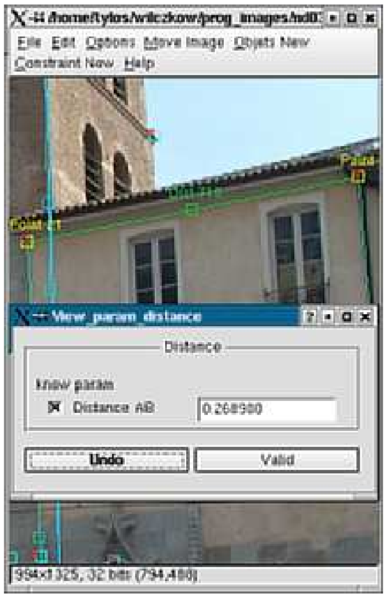
- [✓] Angle L1L2: 90.000000
- [✓] Angle L2L3: 90.000000
- [✓] Angle L3L1: 90.000000
- [ ] Ratio L2/L1: 0.000000
- [ ] Ratio L3/L1: 0.000000
- [ ] Length L1: 0.000000

(a)


X -¤ /home/tylos/wilczkow/prog_images/nd0
File Edit Options Move Image Objets New
Constraint New Help

994x1325 (320x428), 32 bits

(b)

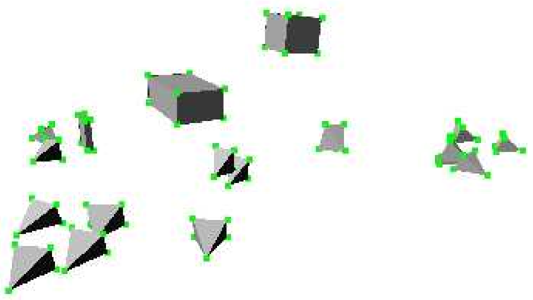

X -¤ /home/tylos/wilczkow/prog_images/nd0
File Edit Options Move Image Objets New
Constraint New Help
Point
X -- View_param_distance
Distance
Undo
Valid

know param

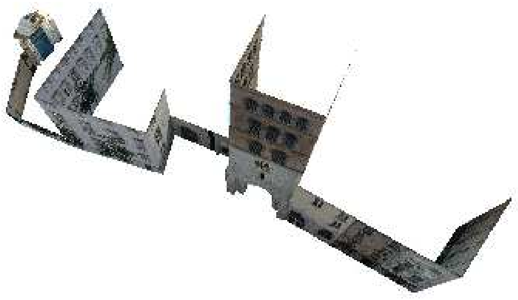
- [✓] Distance AB: 0.268900

994x1325, 32 bits (794,488)

(c)



(d)



(e)

Figure 22: Examples of input and output of the system; Input: (a),(b) Images of primitives are matched between images; (a)(c) Euclidean properties such as parameters of primitives or distances are introduced interactively. Output: (d) Reconstructed calibration primitives and cameras; (e) Full textured scene model.

Finally, we describe an approach for the satisfaction of the introduced geometrical constraints in a non-linear optimization process. Some of the results were published in [Wilczkowiak et al., 2003d; Trombettoni and Wilczkowiak, 2003]. The most important feature of our approach is that contrary to most of the existing methods it is able to deal with a large set of various types of objects and constraints. In the general case, the 3D geometrical problem is translated into a large system of linear, bilinear, or quadratic equations which are difficult to solve using numerical methods. We propose a different approach, based upon an algorithm called General Propagation of Degrees of Freedom (GPDOF), derived originally from local propagation methods. This allows the transformation of a system composed of a set of variables constrained by a set of equations into a system described by a set of free variables, called *input parameters* and a sequence of routines, called *plan*, whose execution results in a model satisfying the constraints. Having such a parameterized model it is possible to use standard optimization techniques to find a model fitting the image data. Contrary to existing approaches, our method allows for scene parameterization in a polynomial time and insures finding a solution satisfying all the constraints if only one exists. In such a case, although the model parameterization is only quasi-minimal, all the constraints are satisfied. We outline also how a minimal scene parameterization might be found, however as long as the constraints remain satisfied, the interest of such a minimization remains marginal due to the computation cost significantly high comparing to the proposed method.

When using such an approach, the main problem arising occurs when the introduced constraints are redundant. Generally, dealing with redundant data is still an open problem. Our algorithm is able to specify all of the objects and constraints involved into an over constrained subsystem. We also propose some algorithms to remove some frequently occurring redundancies.

Outline

This thesis is composed of three parts. The first part introduces basic concepts of projective geometry and representation of objects used in the thesis (chapter 1) as well as a state of the art in the use of geometrical constraints for Computer Vision algorithms (chapter 2).

The second part introduces linear approaches for 3D modeling using geometrical primitives. First, the parallelepiped-based calibration approach is introduced in chapters 3-5. Then a multi-linear reconstruction method based on a scene description by simple primitives is presented in chapter 6. Results of both methods are presented in chapter 7.

The third part describes our approach for quasi-minimal scene parameterization and its application to non-linear optimization of the scene and camera parameters. An overview of the method and necessary background are presented in chapter 8. Chapter 9 details various methods used during the constraint satisfaction process, such as the automatic r-method addition phase and the `GPDOF` algorithm. The chapter is completed by a discussion about difficulties linked to constraint satisfaction, that is, the cases of singularity and the consequences of redundant constraints. The optimization phase is described in chapter 10. Chapter 11 shows the experiments which have been performed on two models significantly differing in the number of objects and images used for reconstruction. Chapter 12 compares `GPDOF` to other equation system decomposition algorithms, some of them being developed by the Computer Aided Design Community.

Part I

Background

Notations

a, λ	scalars
$\mathbf{v}$, $\mathbf{V}$	vectors
A, Ω	matrices
$\wedge$	*join* operator
$\vee$	*meet* operator
$\times$	vector product
$[\mathbf{x}]_\times$	skew-symmetric matrix defined for 3-vectors $\mathbf{x} = (^\top x_1, x_2, x_3)$ of the form $\begin{pmatrix} & -x_3 & x_2 \\ x_3 & & -x_1 \\ -x_2 & x_1 & \end{pmatrix}$. For any 3-vector $\mathbf{y}$ holds $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$
$\mathbf{e}_i^n$	elementary n-vectors of the form $(0, \cdots, \overbrace{1}^{i}, \cdots, 0)^\top$
$\Vert\mathbf{x}\Vert$	2-norm $\Vert\mathbf{x}\Vert_2$ corresponding to euclidean vector length
$\Phi(\mathsf{A})$	nullspace of matrix A
A^+	pseudoinverse of matrix A
$\mathbf{I_n}$	identity $n \times n$ matrix
$\mathbf{0}_n$	zero n-vector
$\mathcal{A}^n$	affine n-dimensional space
$\mathcal{M}^n$	metric n-dimensional space
$\mathcal{P}^n$	projective n-dimensional space
$\mathbb{R}$	set of real numbers
$\sim$	equality of two vectors up to a scalar factor

Chapter 1

Geometric Concepts in Computer Vision

This chapter resumes the basic concepts of geometry essential to read this thesis. Generally, the image-scene relations are of a projective nature, thus the projective properties of space are reviewed at first. Then affine and euclidean subspaces are defined in order to formulate the conditions necessary to calibrate the cameras, i.e. pass from the projective scene description level to the euclidean one, which is the interest of the methods presented in this thesis. The exhaustive description of the concepts adressed in this chapter can be found in classical books concerning projective geometry and computer vision, in particular [Semple and Kneebone, 1952; Faugeras, 1993; Hartley and Zisserman, 2000]. This overview was mainly inspired by [Hartley and Zisserman, 2000] and Marc Pollefeys' tutorial on 3D modelling from images [Pollefeys, 2000].

After a general overview, the details of the scene representation are given. First the linear camera model is described in terms of its projective and euclidean properties. Then the scene 3D structure modelling is discussed. The choice of the scene representation influences the performance of any method. General models allow for easy extensions. On the other hand, models strictly adapted to the given information and required results increase the computation efficiency.

In this thesis we use two different scene representations. In the reconstruction algorithms (section 6 and part III) stress is put on the flexibility of the representation of different objects and constraints, and so the scene is represented by simple primitives such as points, lines and planes and the constraints between them. Such primitives are often represented using homogeneous coordinates [Semple and Kneebone, 1952; Stolfi, 1991; Carlsson, 1993; Hartley and Zisserman, 2000]. This representation is discussed here with special consideration of its euclidean interpretation, partially following the formalism used in [Heuel, 2001]. It is followed by an overview of the various representations of geometrical constraints used in reconstruction approaches.

Contrary to reconstruction algorithms, our calibration algorithms are based on scene descriptions using complex primitives, which are discussed at the end of this chapter. Complex primitives allow the representation of many useful properties in a compact form, and constrain the calibration, and reconstruction tasks. However, with such scene representations it is not always convenient to model non-standard shapes. First a classical approach [Debevec et al., 1996] is discussed. Then it is compared to the parameterisation used in our calibration method (part II).

As most of the readers are familiar with the geometrical framework provided in this chapter, we give only the definitions and properties necessary to situate our work in the state of the art, and focus mainly on giving geometrical interpretations of the presented concepts.

1.1 Projective Geometry: Basic Concepts

The elements of the n-dimencsional projective space $\mathcal{P}^n$ are represented by $n+1$-vectors of homogeneous coordinates. We make no distinction between objects and their vector or matrix representations. For example point $\mathbf{x}$ is represented by an $n+1$ dimensional homogeneous vector $\mathbf{x} = (x_0, \cdots, x_n)$ such that $\mathbf{x} \neq \mathbf{0}_{n+1}$. Two representations $\mathbf{x}_1$, $\mathbf{x}_2$ in the projective space $\mathcal{P}^n$ correspond to the same element if there exists a scalar λ such that $\mathbf{x}_1 = \lambda \mathbf{x}_2$, or, equivalently $\mathbf{x}_1 \sim \mathbf{x}_2$.

1.1.1 Points and Hyperplanes

0-dimensional elements of $\mathcal{P}^n$ are called points. The $(n-1)$ dimensional subspaces of $\mathcal{P}^n$ are called hyperplanes. Both points and hyperplanes are represented by $(n+1)$-vectors of homogeneous coordinates. A point $\mathbf{x}$ is incident with a hyperplane π if:

$$\pi^T \mathbf{x} = 0 \Leftrightarrow \mathbf{x}^T \pi = 0.$$

This means that a hyperplane can be defined as the *join* of (minimal linear space spanned by) n points and *vice versa* a point can be defined by the *meet* (intersection) of n hyperplanes. This leads to the following Duality Principle:

To any theorem in projective space $\mathcal{P}^n$ which includes points and hyperplanes there exists a dual theorem, which can be derived by interchanging the role of the points and the hyperplanes and the operators (for example join and meet).

For example, in $\mathcal{P}^2$, points and lines are dual, and in $\mathcal{P}^3$, points and planes are dual.

1.1.2 Conics and Quadrics

Let us consider a quadratic form

$$\mathsf{Q}(\mathbf{v}) = \mathbf{v}^T \mathsf{Q} \mathbf{v} \tag{1.1}$$

represented by a symmetric matrix $\mathsf{Q}_{n+1 \times n+1}$.

A *quadric (quadric locus)* Q in $\mathcal{P}^n$ consists of the points $\mathbf{x}$ in $\mathcal{P}^n$ satisfying the equation $\mathsf{Q}(\mathbf{x}) = 0$.

A *dual quadric (quadric envelope)* Q^* in $\mathcal{P}^n$ consists of the hyperplanes π in $\mathcal{P}^n$ satisfying the equation $\mathsf{Q}^*(\pi) = 0$.

It is easy to show, that the symmetric matrix $\mathsf{Q}_{n+1 \times n+1}$ representing a quadric in $\mathcal{P}^n$ is defined only up to a scale factor, and has $\frac{n^2+3n}{2}$ degrees of freedom. A quadric (locus and envelope) is called *proper* when represented by a full-rank matrix Q and *degenerated* otherwise.

A point $\mathbf{x}$ and a quadric Q define a hyperplane $\pi \sim \mathsf{Q}\mathbf{x}$. The hyperplane π is called the *polar* of $\mathbf{x}$ with respect to Q and the point $\mathbf{x}$ is called the *pole* of π with respect to Q (this relation for points and lines in $\mathcal{P}^2$ is shown on the figure 1.1–(a)).

Points $\mathbf{x}$ and $\mathbf{y}$ are *conjugate* with respect to quadric Q if $\mathbf{x}$ lies on the polar hyperplane of $\mathbf{y}$ and *vice versa*. This means that for two conjugate points $\mathbf{x}$, $\mathbf{y}$ we have:

$$\mathbf{y}^{\mathsf{T}} \mathsf{Q} \mathbf{x} = 0 \Leftrightarrow \mathbf{x}^{\mathsf{T}} \mathsf{Q} \mathbf{y} = 0.$$

The hyperplane π polar with respect to Q to a point $\mathbf{x}$ lying on Q is *tangent* to Q in point $\mathbf{x}$.

Hyperplanes tangent to a quadric Q belong to the *quadric dual to* quadric Q: Q^* (see figure 1.1–(a) and 1.1–(b)).

A quadric dual to a proper quadric Q represented by matrix Q is represented by matrix $\mathsf{Q}^* = \mathsf{Q}^{-1}$.

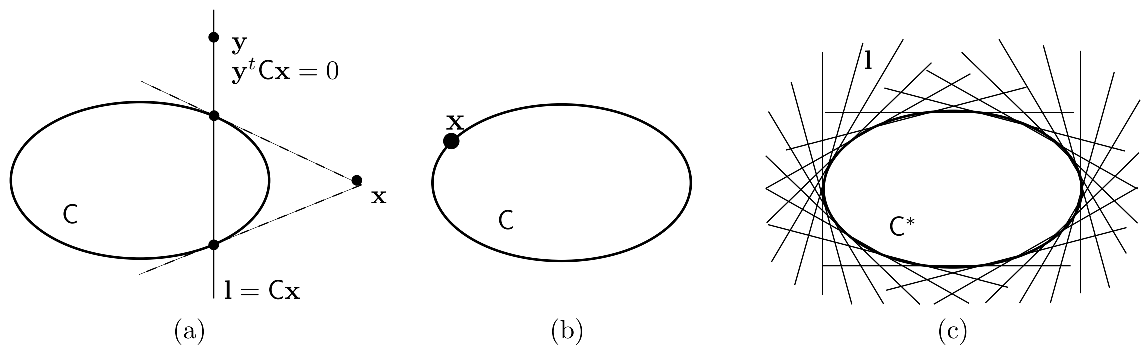

y
$\mathbf{y}^t\mathsf{C}\mathbf{x} = 0$
x
C
$\mathbf{l} = \mathsf{C}\mathbf{x}$
(a)
x
C
(b)
l
C*
(c)

Figure 1.1: Basic conic properties. (a) Conjugacy and polarity concept. Point $\mathbf{x}$ is the pole of line $\mathbf{l}$ with respect to C. Points $\mathbf{x}$ and $\mathbf{y}$ are conjugate with respect to C; (b) Point conic; (c) Dual conic (envelope).

In the literature, the term *quadric* is usually used for the quadric of $\mathcal{P}^3$. In the following we will use the following definitions: A *quadric (surface)* Q is a quadric in $\mathcal{P}^3$ defined by points lying on the surface or, dually, by planes tangent to the quadric surface. It is defined by nine independent parameters. A *conic (curve)* C is a quadric in $\mathcal{P}^2$ defined by points lying on the conic curve or, dually, by 2D lines tangent to the conic curve. It is defined by five independent parameters.

1.1.3 Projective Transformation

An invertible linear mapping $\mathcal{P}^n \rightarrow \mathcal{P}^n$ is called a *projectivity*, *projective transformation*, *collineation* or *homography*. It can be represented by a $n+1 \times n+1$ matrix T. The transformation equations for points, hyperplanes and quadrics under transformation T are the following:

$$\begin{aligned}
\text{point: } \mathbf{x} &\rightarrow \mathsf{T}\mathbf{x} \\
\text{hyperplane: } \pi &\rightarrow \mathsf{T}^{-\mathsf{T}}\pi \\
\text{quadric: } \mathsf{Q} &\rightarrow \mathsf{T}^{-\mathsf{T}}\mathsf{Q}\mathsf{T}^{-1} \\
\text{dual quadric: } \mathsf{Q}^* &\rightarrow \mathsf{T}\mathsf{Q}^*\mathsf{T}^{\mathsf{T}}
\end{aligned}$$

1.1.4 Projection

A linear mapping $\mathcal{P}^n \rightarrow \mathcal{P}^m$ where $n > m$ is called a *projection* and can be represented by an $(m+1) \times (n+1)$ matrix P.

1.2 From Projective to Euclidean Space

A 3D space can be described at different levels. When describing the geometry of 3D objects we generally use euclidean attributes such as distances and angles. However in a perspective image of a 3D scene only projective invariants, such as cross-ratio, incidence and collinearity are preserved. Consequently, without any additional information, we can recover at most those properties from a set of images. Depending on the given information, it is possible to recover different properties of the scene. Usually we use three different "layers"-strata of the space: projective, affine and euclidean. The transformations, which applied to an object, leave the result in the same stratum, are respectively called, projective, affine and euclidean. The group of euclidean transformations is a subgroup of the group of affine transformations which is a subgroup of the group of projective transformations. Every type of transformation can be characterized by its invariants: the spatial properties it leaves unchanged. Invariants are the basis of all the existing calibration and reconstruction algorithms. In the following we resume the properties of the projective, affine, and euclidean spaces necessary to understand the theorical foundations of calibration and reconstruction algorithms proposed in this thesis. By default, all the given properties apply to $\mathcal{P}^3$, however some equivalents are also given for $\mathcal{P}^2$.

1.2.1 Projective Stratum

The group of projective transformations is the most general group of linear transformations, i.e. transformations mapping any line to a line.

As stated in the previous section a projective transformation in $\mathcal{P}^3$ is represented by a 4×4 invertible matrix:

$$\mathsf{T}_{\mathcal{P}} \sim \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{pmatrix} \tag{1.2}$$

Projective transformations have 15 degrees of freedom and preserve collinearity, incidence and conjugacy.

Other invariants of projective transformations are cross-ratios, e.g. the point cross-ratio. Let us consider four collinear points $\mathbf{X}_1 \cdots \mathbf{X}_4$. They can be represented as $\mathbf{X}_i \sim \mathbf{X}_0 + \lambda_i \mathbf{X}'$ for certain $\mathbf{X}_0, \mathbf{X}',\ \mathbf{X}_0 \not\sim \mathbf{X}'$. The cross-ratio of $\mathbf{X}_1 \cdots \mathbf{X}_4$ is defined as follows:

$$\{\mathbf{X}_1; \mathbf{X}_2; \mathbf{X}_3; \mathbf{X}_4\} = \frac{\left(\frac{\lambda_1 - \lambda_3}{\lambda_1 - \lambda_4}\right)}{\left(\frac{\lambda_2 - \lambda_3}{\lambda_2 - \lambda_4}\right)}.$$

The cross-ratio can be seen as the coordinate of a fourth point in the basis consisting of the remaining three, since three points form a basis for the projective line $\mathcal{P}^1$. It is defined also for points at infinity and does not depend on the choice of $\mathbf{X}_0$ and $\mathbf{X}'$. The cross-ratio of three points and a point at infinity is the ratio of distances of two of the three points with the third one. As already mentioned the value of cross-ratio remains constant under any projective transformation.

1.2.2 Affine Stratum

The affine space $\mathcal{A}^3$ is embedded in $\mathcal{P}^3$ under the mapping:

$$\mathcal{A}^3 \rightarrow \mathcal{P}^3 : (x, y, z)^{\mathsf{T}} \rightarrow (x, y, z, 1)^{\mathsf{T}}. \tag{1.3}$$

It is easy to see that points from $\mathcal{P}^3$ of the form $(x, y, z, 0)$ do not belong to the range of this transformation. Those points could be seen as limit points of the form $(x, y, z, \frac{1}{t})^{\mathsf{T}}$ for $t \rightarrow \infty$. Consequently, they are called *points at infinity.* The plane $\pi_\infty = (0, 0, 0, 1)$ formed by those points is called *plane at infinity.* Mapping (1.3) can be seen as a one-to-one mapping between the affine space and the projective space minus the plane at infinity. Note that lines characterized by the same direction vector always meet at points lying on the plane at infinity. Thus direction $\mathbf{d}$ in 3D space can be identified with a point at infinity $\mathbf{d} = (x_d, y_d, z_d, 0)$. Since such points do not belong to the affine space, parallel lines never intersect in the affine space.

In the projective plane $\mathcal{P}^2$, all the points at infinity lie on the *line at infinity* $\mathbf{l}_\infty$, so the affine plane corresponds to the projective plane minus the line at infinity.

The *affine transformation* is a projective transformation represented by a 4×4 invertible matrix:

$$\mathsf{T}_\mathcal{A} \sim \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.4}$$

Affine transformations have 12 degrees of freedom and, unlike general projective transformations, leave the plane at infinity globally unchanged, although individual points at infinity may vary. In consequence, parallelism (points at infinity remain at infinity) and distance ratios on parallel lines (cross-ratios with points at infinity) are also preserved.

From projective to affine. Let us consider a structure that is known up to a projective transformation. As was already stated, generally projective transformations do not preserve the plane at infinity (see figure 1.2). Thus, to resolve the projective ambiguity, the plane at infinity first has to be located in $\mathcal{P}^3$ (or line at infinity for the projective plane $\mathcal{P}^2$). Usually this can be done by using information about characteristics of the "true" scene which are invariant under affine transformations: parallelism and length ratios on parallel lines.

1.2.3 Metric and Euclidean Stratum

The euclidean stratum corresponds to euclidean transformations which can be represented by the composition of a rotation R and translation $\mathbf{t}$. The metric stratum corresponds to the group of similarities, i.e. euclidean tranformations combined with a scaling by factor s.

It is represented by a 4×4 invertible matrix:

$$\mathsf{T}_\mathcal{M} \sim \begin{pmatrix} sr_{11} & sr_{12} & sr_{13} & t_1 \\ sr_{21} & sr_{22} & sr_{23} & t_2 \\ sr_{31} & sr_{32} & sr_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1.5}$$

where r_{ij} are the elements of matrix R and t_i are the elements of vector $\mathbf{t}$. A metric transformation has 3 (rotation) + 3 (translation) + 1 (scaling) = 7 degrees of freedom.

Absolute Conic and Dual Absolute Quadric. The metric properties of a projective space are encoded in the *absolute conic* Ω_∞ or its dual entity, the *dual absolute quadric* Q^*_∞.

The *absolute conic* is a conic curve constituted of points $\mathbf{X} = (x_1 \cdots x_4)$ lying on the intersection of the imaginary sphere defined by $\mathbf{x}^\mathsf{T}\mathbf{I_4}\mathbf{x}$ with the plane at infinity $\pi_\infty : x_4 = 0$. Thus it could be seen as an imaginary circle located in the plane at infinity. In a metric space the absolute conic is the conic $\Omega_\infty \sim \mathbf{I_3}$ of the plane at infinity. Note the following properties of the absolute conic:

- All spheres intersect π_∞ in Ω_∞.
- Any circle intersects Ω_∞ in two points: the *circular points* of the plane supporting the circle.
- Ω_∞ can be used to express the angle Θ between two directions in space $\mathbf{d}_1$, $\mathbf{d}_2$:

$$\cos(\Theta) = \frac{(\mathbf{d}_1^\mathsf{T}\Omega_\infty\mathbf{d}_2)}{\sqrt{(\mathbf{d}_1^\mathsf{T}\Omega_\infty\mathbf{d}_1)(\mathbf{d}_2^\mathsf{T}\Omega_\infty\mathbf{d}_2)}}$$

  Thus two vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ are orthogonal if $(\mathbf{d}_1^\mathsf{T}\Omega_\infty\mathbf{d}_2) = 0$, which is equivalent to the conjugacy of $\mathbf{d}_1$ and $\mathbf{d}_2$ with respect to Ω_∞.

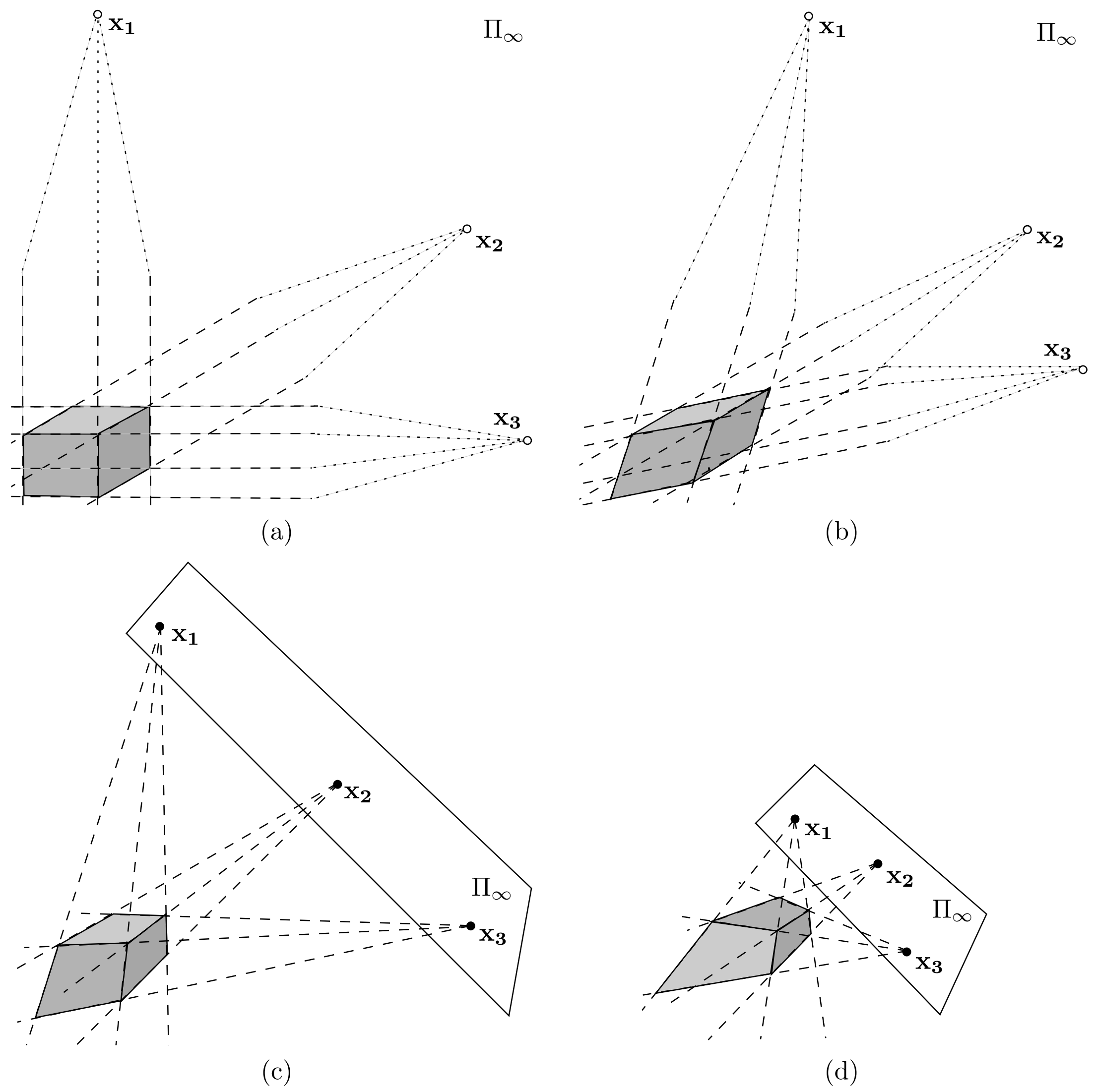


Figure 1.2: Cube under (a) metric; (b) affine; (c), (d) two projective transformations differing in the position of the plane at infinity Π_∞ and locations of points at infinity $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ on Π_∞. Pairs of points $\mathbf{x}_i$, $\mathbf{x}_j$ are conjugated with respect to the absolute conic Ω_∞.

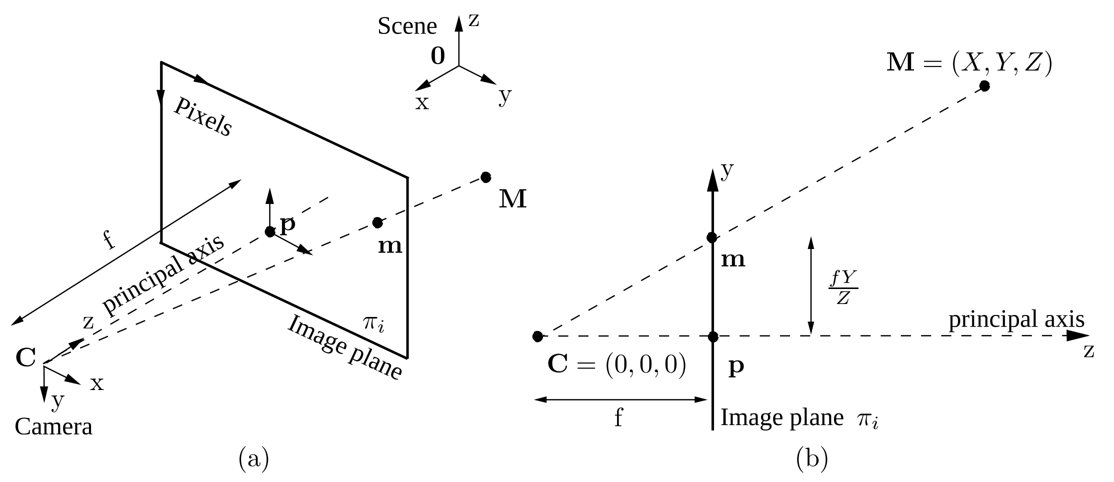


Figure 1.3: Pinhole camera geometry. (a) Scene, camera, image plane and pixel coordinate frames; (b) The projection to the image plane in camera coordinate frame.

The *Absolute dual quadric* Q^*_∞ in metric space is represented in canonical form by the rank 3 symmetric matrix

$$\mathsf{Q}^*_\infty = \begin{pmatrix} \mathbf{I_3} & 0 \\ 0 & 0 \end{pmatrix}$$

Geometrically, the dual absolute quadric Q^*_∞ is formed by the the planes tangent to Ω_∞. Q^*_∞ has the following properties:

- The plane at infinity π_∞ is the null vector of Q^*_∞.
- The angle Θ between two planes π_1, π_2 in the metric space are given by:

$$\cos(\Theta) = \frac{(\pi_1^\mathsf{T} \mathsf{Q}^*_\infty \pi_2)}{\sqrt{(\pi_1^\mathsf{T} \mathsf{Q}^*_\infty \pi_1)(\pi_2^\mathsf{T} \mathsf{Q}^*_\infty \pi_2)}}$$

It is easy to check that the absolute conic and the dual absolute quadric and, consequently, angles and distance ratios, are preserved under the metric transformations. The absolute distances are preserved only under euclidean transformations.

From affine to metric and euclidean. To obtain a metric reconstruction of the scene, the absolute conic must be located on the plane at infinity π_∞ (or the circular points on the line at infinity in $\mathcal{P}^2$). Usually this can be done using information about characteristics of the "true" scene which are invariant under metric transformations: angles and length ratios. An euclidean reconstruction can be obtained when the absolute scene scale is known.

1.3 Camera Representation

A camera induces a mapping between the 3D world and the 2D image. A camera model inducing a linear mapping is called a *pinhole camera*, and will be used all over this thesis. It is a simplified model, not taking lense distortion into consideration; however it turns out to be sufficient for many applications and allows an important reduction of computational complexity. In the following we briefly describe its properties.

The action of the pinhole camera on 3D points is depicted on figure 1.3. The camera is defined by its *projection centre* **C** and the *image plane* π_i. The axis orthogonal to the image plane and passing through the projection centre **C** is called the *principal axis* and its intersection with the image plane is called the *principal point.* Under the pinhole camera model, a 3D point **X** is mapped to the point **x** lying on the intersection of the image plane with the line joining **X** and the projection centre **C**.

Algebraically the projection can be represented by a *projection matrix* $\mathsf{P}_{3\times 4}$:

$$\mathbf{x} \sim \mathsf{P}\mathbf{X}. \tag{1.6}$$

This mapping has 11 degrees of freedom, which, as explained below, correspond to 5 intrinsic and 6 position and orientation parameters. It can be represented as a product of mappings between four coordinate frames:

3D scene coordinate frame Coordinate frame in which the 3D scene points, as well as camera position and orientation are expressed.

3D camera coordinate frame Coordinate frame centered at projection centre **C** and such that its z-axis is the camera principal axis and the image plane is represented by: $z = f$. f is called the *focal length.* The axes of this coordinate frame will be called the *camera axes.*

2D image plane coordinate frame Coordinate frame on the image plane, centered in the principal point and such that axes x, y are parallel to the camera x, y axes.

2D image pixel frame Coordinate frame associated to pixels inside the image plane, such that at least one axe is parallel to one of the axes of the image plane coordinate frame.

The mapping from the scene frame to the image pixel frame is composed of the following mappings:

1. Euclidean mapping from the scene to camera coordinates:

$$\mathsf{T} = \begin{pmatrix} \mathsf{R}_{3\times 3} & -\mathsf{R}\mathbf{t}_3 \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{pmatrix},$$

where R is the rotation matrix representing the camera orientation and $\mathbf{t}$ is the camera position in the scene frame.

2. Perspective projection from the 3D camera frame to the image plane (see also figure 1.3–(b):

$$\mathsf{P}^{\mathcal{P}} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

3. Affine transformation between the image plane coordinate frame and the pixel frame:

$$\mathsf{K} = \begin{pmatrix} k_u & -k_u \cot\Theta & u_0 \\ 0 & \frac{k_v}{\sin\Theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix},$$

where k_u, k_v correspond to number of pixel per milimetr, Θ the angle between the pixel axes and (u_0, v_0) coordinates of the principal point in the image pixel frame.

Thus, the projection matrix P can be decomposed as:

$$\begin{aligned}\mathsf{P} \sim \mathsf{A}\mathsf{P}^{\mathcal{P}}\mathsf{T} &\sim \begin{pmatrix} k_u f & -k_u f \cot\Theta & u_0 & 0 \\ 0 & f\frac{k_v}{\sin\Theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathsf{R}_{3\times3} & -\mathsf{R}\mathbf{t}_3 \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{pmatrix} \\ &\sim \underbrace{\begin{pmatrix} \alpha_u & s & u_0 & 0 \\ 0 & \frac{\alpha_v}{\sin\Theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathsf{K}} \begin{pmatrix} \mathsf{R}_{3\times3} & -\mathsf{R}\mathbf{t}_3 \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{pmatrix}.\end{aligned}$$

where $\alpha_u = k_u f$, $\alpha_v = f k_v$ and $s = -k_u f \cot\Theta$ together with the principal point (u_0, v_0) depend on the internal camera settings, and are called the *intrinsic camera parameters*, while R and $\mathbf{t}$ correspond to the camera orientation and position in space and are called the *extrinsic camera parameters*. The projection such defined has 3 (rotation) +3 (translation) +5 (intrinsic parameters) = 11 degrees of freedom, which confirms the statement referring to equation (1.6). Matrix K is called the camera *calibration matrix*. In the following we will only consider a camera model with rectangular pixels ($\Theta = 90°$), so matrix K will be of the form:

$$\mathsf{K} \sim \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

The ratio $\tau = \frac{\alpha_u}{\alpha_v}$ is called the *aspect ratio*.

Note, that when matrix P is known, it is possible to decompose it into K, R, $\mathbf{t}$, using for example QR decomposition [Tsai, 1986].

Infinite homography and image of the absolute conic (IAC). The image of the absolute conic plays a special role in the theorical basis of calibration algorithms and is strictly connected to the camera internal parameters. First let us consider the action of the projection transformation on points at infinity. For a point $\mathbf{X} \sim (X, Y, Z, 0)^{\mathsf{T}}$ and its image $\mathbf{x} \sim (x, y, t)^{\mathsf{T}}$ this transformation can be expressed as:

$$\begin{pmatrix} x \\ y \\ t \end{pmatrix} \sim \mathsf{K}\begin{pmatrix} \mathsf{R}_{3\times3} & -\mathsf{R}\mathbf{t}_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 0 \end{pmatrix} \sim \mathsf{K}\mathsf{R}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Thus, the image of points at infinity does not depend on the camera position, and $\mathsf{H} = \mathsf{K}\mathsf{R}$ is a homography between the plane at infinity and the image plane. The homography H_{ij} between two images i, j induced by the plane infinity, such that $\mathsf{H}_{ij} = \mathsf{H}_j\mathsf{H}_i^{-1}$ is called the *infinite homography*.

Using results presented in section 1.1 the IAC (image of the absolute conic) is:

$$\omega \sim \mathsf{H}^{-\mathsf{T}}\mathbf{I_3}\mathsf{H}^{-1} \sim \mathsf{K}^{-\mathsf{T}}\mathsf{K}^{-1} \sim \begin{pmatrix} 1 & 0 & -u_0 \\ 0 & \tau^2 & -\tau^2 v_0 \\ -u_0 & -\tau^2 v_0 & \tau^2\alpha_v^2 + u_0^2 + \tau^2 v_0^2 \end{pmatrix} \tag{1.7}$$

This means that given the IAC, the camera's intrinsic parameters can be recovered, as the IAC depends solely and bijectively on these parameters. This can be done *via* the Cholesky factorization of ω for example. The absolute conic and dual absolute quadric and their images are depicted on figure 1.4.

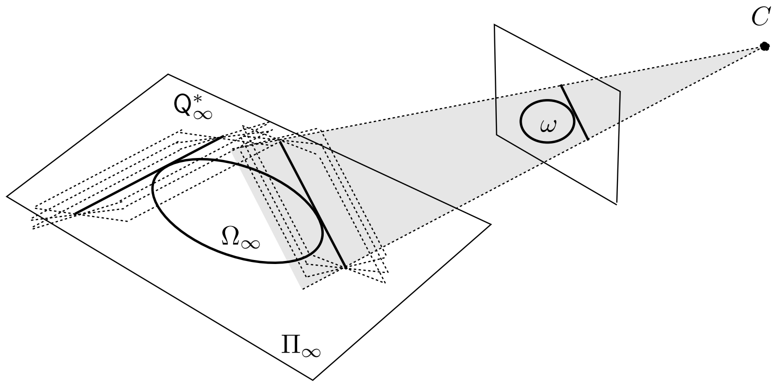


Figure 1.4: Absolute conic Ω_∞ and dual absolute quadric Q^*_∞ and their projection in a camera.

1.4 Scene Representation

The elegancy and efficiency of any algorithm depend crucially on the internal object representation. Choosing a general method facilitates its extensions. On the other hand, using a model strictly adapted to the given information and required results usually increases the computation efficiency. Thus the choice of object representation is the first problem that must be solved before the algorithm can be implemented. The algorithms for calibration and for scene reconstruction presented in this thesis differ in the level of the scene representation. In the reconstruction algorithms presented in section 6 and part III the stress is put on the flexible definition of various types of scene primitives and constraints. The scene is thus represented by simple primitives such as points, lines and planes and constraints between them. This allows easy modelling of various shapes in 3d scenes, however may require considerable amount of human interaction. On the contrary, complex primitives allow the representation of many useful properties in a compact form and constrain the calibration and reconstruction task. As the interest of the calibration method presented in part II is to obtain results using a minimal amount of data, we decided to use a complex primitive, the parallelepiped, which naturally encodes many interesting scene properties.

The scene modelisation using simple primitives and constraints is discussed in section 1.4.1.2. Two examples of scene parameterisation using complex primitives are then given in section 1.4.2.

1.4.1 Basic Primitives and Constraints

Here, all objects are represented using homogeneous vectors [Hartley and Zisserman, 2000]. An elegant description of the relations between them is given by the Grassman-Cayley algebra. A good introduction to this algebra for scene and camera modelling can be found in [Carlsson, 1993]. Here, we focus on giving an intuition of the geometric interpretation, and a practical expression which can be used directly to impose the geometrical constraints.

Various relations in projective space can be described using two operators: *meet* $\vee$ and *join* $\wedge$ (mentioned already in section 1.1.1). Applied on geometric entities, they respectively result in their intersection and the minimal linear space containing the entities. For example, a point can be computed as the *meet* of 3 planes, or a plane can be computed as the *join* of 3 points.

However the Grassmann-Cayley algebra is designed to describe the projective object properties, when dealing with objects not lying at infinity it is possible to reformulate its results in euclidean terms, using properties like distances and orthogonality. As the interest of work presented in this thesis is the euclidean reconstruction, such a euclidean understanding of objects and constraints will often be considered.

1.4.1.1 Primitives

Generally, the primitives in space are represented by homogeneous vectors $\mathbf{V}$ defined up to scale. When working in a euclidean space, such vector $\mathbf{V}$, for any type of primitive considered, can be decomposed [Förstner et al., 2000; Heuel, 2001] in a homogeneous part $\mathbf{V}_h$ and euclidean part $\mathbf{V}_0$. When the vector $\mathbf{V}$ is scaled such that $\|\mathbf{V}_h\| = 1$, the distance of the object from the origin is represented by $d = \|\mathbf{V}_0\|$. From now on, when this decomposition is used we assume that all the objects are not at infinity and that the homogeneous part has unit length.

Apart from coordinate vectors, the matrix representation of objects is introduced. Such matrices are very useful to express the inter-object constraints and were used in the implementation of methods presented in part III and appendix 6. Although it is not necessary in order to understand the algorithms presented in this thesis, we find the geometric interpretation of these matrices interesting and describe it according to [Förstner et al., 2000].

The object descriptions are sorted by their dimensionality, but for readers unfamiliar with the Plücker line representation we suggest they first read the line representation description further below.

Points. A 3D point has 3 degrees of freedom and is represented by a homogeneous vector:

$$\mathbf{X} = \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} \mathbf{X}_0 \\ \\ X_h \end{pmatrix}.$$

In a euclidean space, when the homogeneous part $X_h = t$ is scaled such that $\|X_h\| = 1$ the euclidean part $\|\mathbf{X}_0^\top\|$ corresponds to the distance from the point to the origin.

Matrix representation A point can be localised in space by its coordinates, but other definitions are also possible.

First, let us consider four lines passing through the point $\mathbf{X}$, three of which are parallel to the coordinate frame axes, and one passing through the origin (see figure 1.5–(a)). Actually, those four lines are joins of the point $\mathbf{X}$ and the points from the basis $\mathbf{e}_i^4$. Such representation is unique for all the points in space. Evidently, it is redundant.

Similarly, as shown on figure 1.5–(b), the point $\mathbf{X}$ can be uniquely defined by six planes passing through $\mathbf{X}$ and six lines represented by vectors $\mathbf{e}_i^6$ (see next paragraph for the line representation). For a given point $\mathbf{X} = (x, y, z, t)$ a 4×6 matrix Π, whose rows and columns are respectively composed of the coordinates of the 4 lines and 6 planes representing $\mathbf{X}$, is of the form:

$$\Pi(\mathbf{X}) = \begin{pmatrix} X_h \mathbf{I_{3\times3}} & [\mathbf{X_0}]_\times \\ -\mathbf{X_0}^\top & \mathbf{0}^\top \end{pmatrix} = \begin{pmatrix} t & 0 & 0 & 0 & z & -y \\ 0 & t & 0 & -z & 0 & x \\ 0 & 0 & t & y & -x & 0 \\ -x & -y & -z & 0 & 0 & 0 \end{pmatrix}$$

Lines. A line in space has 4 degrees of freedom and can be represented in Plücker coordinates as a homogeneous 6-vector:

$$\mathbf{L} = (l_1, l_2, l_3, l_4, l_5, l_6)^\top = (\mathbf{L}_h{}^\top, \mathbf{L}_0{}^\top)$$

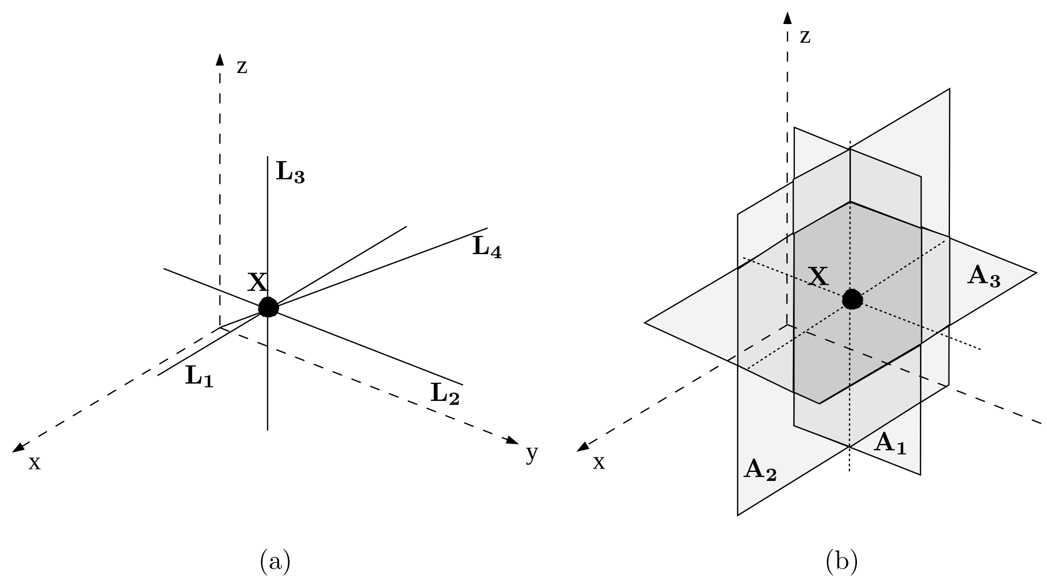


Figure 1.5: Point $\mathbf{X}$ can be represented as the meet of (a) 4 lines defined as join of $\mathbf{X}$ and 4 points represented by $\mathbf{e}_i^4$; (b) 6 planes defined as the meet of $\mathbf{X}$ and 6 lines represented by $\mathbf{e}_i^6$ (only the planes $\mathbf{A}_1$-$\mathbf{A}_3$ are shown on the picture).

respecting the Plücker condition [1]

$$l_1 l_4 + {}_2 l_5 + l_3 l_6 = \mathbf{L_h}^\top \mathbf{L_0} = 0. \tag{1.8}$$

Another interpretation of the Plücker condition is based on the fact that in euclidean space the sub-vector $\mathbf{L}_h$ represents the line's direction, and $\mathbf{L}_0$ the normal of the plane passing through the line and the origin. The Plücker condition imposes the orthogonality of these two vectors. When the vector is scaled such that $\|\mathbf{L}_h\| = 1$, $\|\mathbf{L}_0\|$ represents the distance of the line from the origin (see figure 1.6–(b)).

A dual Plücker representation $\overline{\mathbf{L}}$ of line $\mathbf{L}$ holds the following relation with the line representation $\mathbf{L}$:

$$\overline{\mathbf{L}} \sim (\bar{l}_1, \bar{l}_2, \bar{l}_3, \bar{l}_4, \bar{l}_5, \bar{l}_6)^\top \sim (\overline{\mathbf{L}}_h{}^\top, \overline{\mathbf{L}}_0^\top) \sim (\mathbf{L}_0^\top, \mathbf{L}_h^\top).$$

Note, that for two dual line representations $\mathbf{L}$, $\overline{\mathbf{L}}$ we have:

$$l_1\bar{l}_1 + {}_2\bar{l}_2 + l_3\bar{l}_3 + l_4\bar{l}_4 + {}_5\bar{l}_5 + l_6\bar{l}_6 \sim (l_4, l_5, l_6, l_1, l_2, l_3) \sim \mathbf{L}_0^\top \overline{\mathbf{L}}_h^\top + \mathbf{L}_h^\top \overline{\mathbf{L}}_0^\top \sim 0. \tag{1.9}$$

The dual representation of lines is convenient for expressing geometrical dependencies between objects and will be discussed in the following.

Matrix representation Similarly to points, a line $\mathbf{L}$ can be represented canonically as:

- A join of four points lying on the intersection of $\mathbf{L}$ with four planes represented by $\mathbf{e}_i^4$ (figure 1.6–(a)), three of which are parallel to planes xy, yz, zx, and a fourth one is the plane at

[1]Note that this representation differs from the one used in [Hartley and Zisserman, 2000]

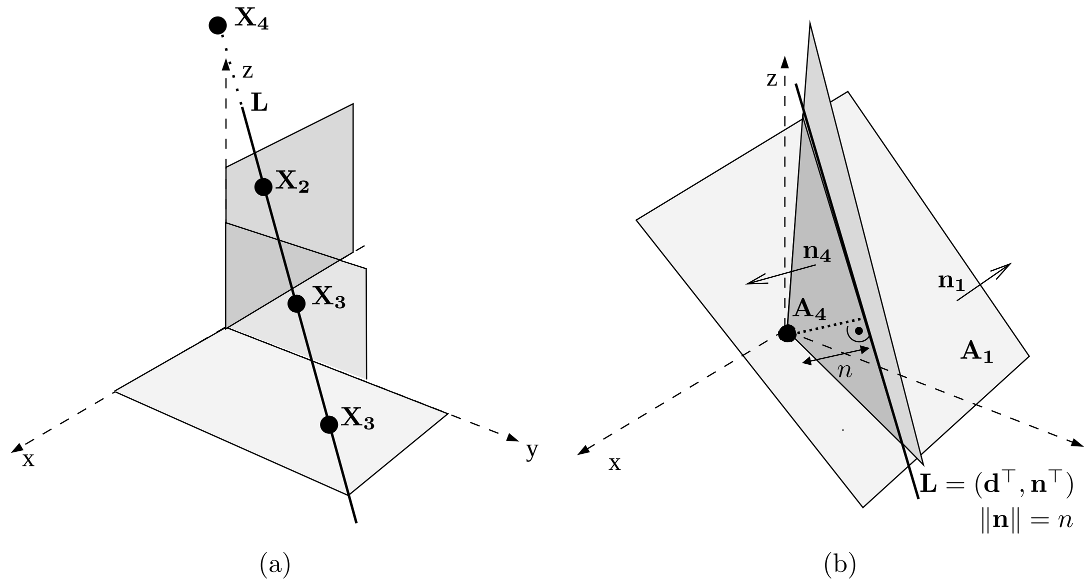


Figure 1.6: Line $\mathbf{L}$ can be represented as (a) join of 4 points defined as meet of $\overline{\mathbf{L}}$ and 4 planes represented by $\mathbf{e}_i^4$; (b) meet of 4 planes defined as join of $\mathbf{L}$ and 4 points represented by $\mathbf{e}_i^4$ (for clarity of the figure only the planes $\mathbf{A}_1$ and $\mathbf{A}_4$ are shown).

infinity. The skew-symmetric matrix $\overline{\Gamma}(\mathbf{L})$ whose rows (and columns) contain the coordinates of those points is of the following form:

$$\overline{\Gamma}(\mathbf{L}) \sim \Gamma(\overline{\mathbf{L}}) \sim \begin{pmatrix} -[\mathbf{L_0}]_\times & -\mathbf{L_h} \\ \mathbf{L_h}^\top & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & l_6 & -l_5 & -l_1 \\ -l_6 & 0 & l_4 & -l_2 \\ l_5 & -l_4 & 0 & -l_3 \\ l_1 & l_2 & l_3 & 0 \end{pmatrix}.$$

- Dually, as meet of four planes spanned by line $\mathbf{L}$ and the points represented by $\mathbf{e}_i^4$ (figure 1.6–(b)). The skew-symmetric matrix $\Gamma(\mathbf{L})$ whose rows (and columns) contain the coordinates of those planes is of the following form:

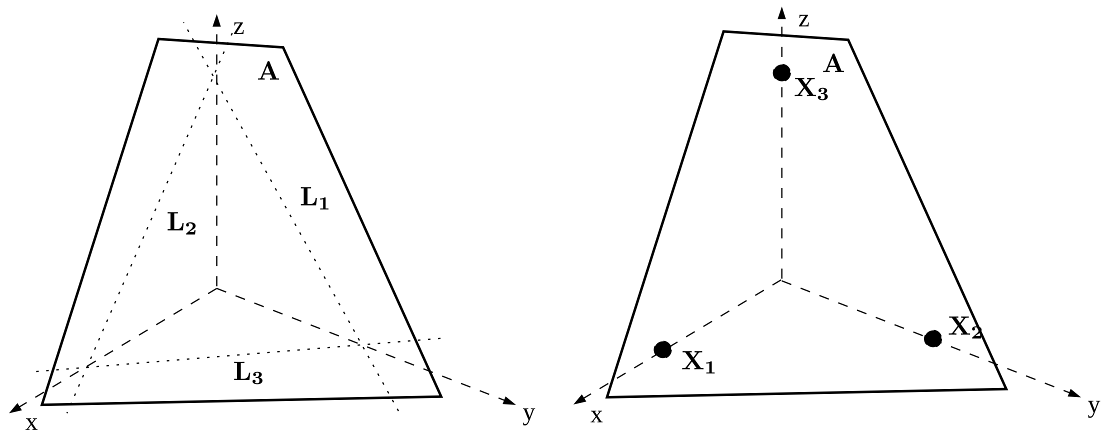
$$\Gamma(\mathbf{L}) \sim \begin{pmatrix} -[\mathbf{L_h}]_\times & -\mathbf{L_0} \\ \mathbf{L_0}^\top & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & l_3 & -l_2 & -l_4 \\ -l_3 & 0 & l_1 & -l_5 \\ l_2 & -l_1 & 0 & -l_6 \\ l_4 & l_5 & l_6 & 0 \end{pmatrix}.$$

Planes. A 3D plane has 3 degrees of freedom and is represented by a homogeneous vector:

$$\mathbf{A}^\mathsf{T} \sim (a, b, c, d) \sim (\mathbf{A}_h^\mathsf{T}, A_0).$$

In euclidean space, when the homogeneous part (the plane normal) $\mathbf{A}_h = (a, b, c)^\mathsf{T}$ is scaled such that $\|\mathbf{A}_h\| = 1$ the euclidean part $\|d = A_0\|$ corresponds to the distance between the plane and origin.

Matrix representation Similarly to points, a plane $\mathbf{A}$ can be represented by:



(a) (b)

Figure 1.7: Plane **A** can be represented as join of (a) 4 lines defined as meet of **A** and 4 planes represented by $\mathbf{e}_i^4$ (only the lines $\mathbf{L}_1$-$\mathbf{L}_3$ are depicted); (b) 6 points defined as join of **A** and 6 lines represented by $\mathbf{e}_i^6$ (only the points $\mathbf{X}_1$-$\mathbf{X}_3$ are depicted).

object	dof	vector representation	related matrix
point	3	$\mathbf{X}^\top = (x,y,z,t) = (\mathbf{X}_0^\top, X_h)$	$\Pi(\mathbf{X}) = \begin{pmatrix} X_h\mathbf{I}_{3\times 3} & [\mathbf{X_0}]_\times \\ -\mathbf{X_0}^\top & \mathbf{0}^\top \end{pmatrix}$
line	4	$\mathbf{L}^\top = (l_1,l_2,l_3,l_4,l_5,l_6) = (\mathbf{L}_h^\top, \mathbf{L}_0^\top)$	$\Gamma(\mathbf{L}) = \begin{pmatrix} -[\mathbf{L_h}]_\times & -\mathbf{L_0} \\ \mathbf{L_0}^\top & 0 \end{pmatrix}$ $\overline{\Gamma}(\mathbf{L}) = \Gamma(\overline{\mathbf{L}}) = \begin{pmatrix} -[\mathbf{L_0}]_\times & -\mathbf{L_h} \\ \mathbf{L_h}^\top & 0 \end{pmatrix}$
plane	3	$\mathbf{A}^\top = (a,b,c,d) = (\mathbf{A}_h^\top, A_0)$	$\overline{\Pi}(\mathbf{A}) = \begin{pmatrix} -[\mathbf{A}_h]_\times & A_0\mathbf{I}_{3\times 3} \\ \mathbf{0}^\top & -\mathbf{A}_h^\top \end{pmatrix}$

Table 1.1: Representation of points, lines and planes in 3D.

- A join of four lines lying on the intersection of **A** with planes xy, yz, zx, and the infinite plane (represented by $\mathbf{e}_i^4$-figure 1.7–(a)).
- A join of 6 points lying on the intersection of **A** with six lines represented by vectors $\mathbf{e}_i^6$ (figure 1.7–(b)).

For a given plane $\mathbf{A}^\top \sim (a,b,c,d)$, a 4×6 matrix $\overline{\Pi}$, whose rows and columns are composed of coordinates of, respectively, lines and points representing **A** is of the form:

$$\overline{\Pi}(\mathbf{A}) \sim \begin{pmatrix} -[\mathbf{A}_h]_\times & A_0\mathbf{I}_{3\times 3} \\ \mathbf{0}^\top & -\mathbf{A}_h^\top \end{pmatrix} \sim \begin{pmatrix} 0 & c & -b & d & 0 & 0 \\ -c & 0 & a & 0 & d & 0 \\ b & -a & 0 & 0 & 0 & d \\ 0 & 0 & 0 & -a & -b & -c \end{pmatrix}$$

The object representation is summarised in table 1.1.

1.4.1.2 Relations Between Primitives

Incidence and distances. Incidence is one of the projective invariants and can be expressed using meet and join operators. In euclidean space, incidence can be considered as a special case of a distance constraint between two objects $\mathbf{v}_1$, $\mathbf{v}_2$: $d(\mathbf{v}_1, \mathbf{v}_2) = 0$. In the following, we resume the incidence and distance constraints between points, lines and planes, and show how the same expressions can be used to express both constraints when working in euclidean space.

As stated in section 1.1.1, a plane can be defined as the join of three points and dually a point can be defined as meet of three planes. The incidence between point $\mathbf{X}$ and plane $\mathbf{A}$ is expressed by:

$$\mathbf{A}^\mathsf{T}\mathbf{X} = 0. \tag{1.10}$$

However, when $\mathbf{X}$ and $\mathbf{A}$ do not lie at infinity and are both scaled such that their homogeneous parts X_h and $\mathbf{A}_h$ have unit length, the absolute value of the left side of equation (1.10) gives the distance between point $\mathbf{X}$ and plane $\mathbf{A}$.

In the following, we analyse the properties which can be derived using meet and join operators on points, lines and planes.

A join of two points $\mathbf{X}$, $\mathbf{Y}$ is a line $\mathbf{L}$ expressed by:

$$\mathbf{L} \sim \Pi^\top(\mathbf{X})\mathbf{Y} \sim \begin{pmatrix} X_h\mathbf{Y}_0 - Y_h\mathbf{X}_0 \\ \mathbf{X}_0 \times \mathbf{Y}_0 \end{pmatrix} \tag{1.11}$$

The above expression reduces to $\mathbf{0}_{6\times1}$, if the vectors $\mathbf{X}$ and $\mathbf{Y}$ represent the same point: $\mathbf{X} \sim \mathbf{Y}$. Otherwise, the norm of the upper part of the vector corresponds to the distance between points $d(\mathbf{X}, \mathbf{Y}) = \|X_h\mathbf{Y}_0 - Y_h\mathbf{X}_0\| = \|\mathbf{Y}_0 - \mathbf{X}_0\|$ (as mentioned on the beginning, we assume that the vectors are scaled such that $\|X_h\| = \|Y_h\| = 1$).

Dually, a line $\overline{\mathbf{L}}$ can be defined by the meet of two planes $\mathbf{A}$, $\mathbf{B}$:

$$\overline{\mathbf{L}} \sim \overline{\Pi}^\top(\mathbf{A})\mathbf{B} \sim \begin{pmatrix} \mathbf{A}_h \times \mathbf{B}_h \\ A_0\mathbf{B}_h - B_0\mathbf{A}_h \end{pmatrix} \tag{1.12}$$

The above expression reduces to $\mathbf{0}_{6\times1}$, if the vectors $\mathbf{A}$ and $\mathbf{B}$ represent the same plane: $\mathbf{A} \sim \mathbf{B}$. When the planes are parallel (the upper part of the vector $\overline{\mathbf{L}}$ is $\mathbf{0}_3$) the norm of the lower part of the vector corresponds to the distance between the two planes: $d(\mathbf{A}, \mathbf{B}) = \|A_0\mathbf{B}_h - B_0\mathbf{A}_h\| = \|A_0 - B_0\|$.

The join of line $\mathbf{L}$ and point $\mathbf{X}$ results in the plane $\mathbf{A}$ passing through $\mathbf{L}$ and $\mathbf{X}$:

$$\mathbf{A} \sim \Gamma(\mathbf{L})\mathbf{X} \sim \begin{pmatrix} \mathbf{X}_0 \times \mathbf{L}_h - X_h\mathbf{L}_0 \\ \mathbf{L}_0^\mathsf{T}\mathbf{X}_0 \end{pmatrix} \tag{1.13}$$

The above expression reduces to $\mathbf{0}_{4\times1}$, if $\mathbf{X}$ is incident with $\mathbf{L}$. Otherwise, the norm of the upper part of the vector corresponds to the distance between the point and the line: $d(\mathbf{X}, \mathbf{L}) = \|\mathbf{X}_0 \times \mathbf{L}_h - X_h\mathbf{L}_0\|$.

The meet of line $\mathbf{L}$ and plane $\mathbf{A}$ gives the point $\mathbf{X}$ lying on the intersection of $\mathbf{L}$ and $\mathbf{A}$:

$$\mathbf{X} \sim \overline{\Gamma}(\mathbf{L})\mathbf{A} \sim \begin{pmatrix} \mathbf{A}_h \times \mathbf{L}_0 - A_0\mathbf{L}_h \\ \mathbf{L}_h^\mathsf{T}\mathbf{A}_h \end{pmatrix} \tag{1.14}$$

The above expression reduces to $\mathbf{0}_{4\times1}$, if line $\mathbf{L}$ is incident with $\mathbf{A}$. When the plane and line are parallel (the lower part of the vector is 0) the norm of the upper part of the vector corresponds to the line-plane distance $d(\mathbf{L}, \mathbf{A}) = \|\mathbf{A}_h \times \mathbf{L}_0 - A_0\mathbf{X}_h\|$.

Table 1.2 summarises the incidence relations. They are bilinear in the coordinates of the objects. Thus, when one object is known, linear constraints in terms of the other object can be formulated.

Incidence of two objects $\mathbf{V}_1$, $\mathbf{V}_2$ of the same type can be expressed in two different ways:

incidence	unknown					
known ↓	*(dof)*	point $\mathbf{X}$	*(dof)*	line $\mathbf{L}$	*(dof)*	plane $\mathbf{A}$
point $\mathbf{Y}$	(*3*)	$\Pi^\top(\mathbf{Y})\mathbf{X} = \mathbf{0}$	(*2*)	$\overline{\Pi}(\mathbf{Y})\mathbf{L} = \mathbf{0}$	(*1*)	$\mathbf{Y}^\top\mathbf{A} = 0$
line $\mathbf{M}$	(*2*)	$\Gamma(\mathbf{M})\mathbf{X} = \mathbf{0}$	(*4*)	$\overline{\Gamma}(\mathbf{M})\Gamma(\mathbf{L}) = \mathbf{0}$	(*2*)	$\overline{\Gamma}(\mathbf{M})\mathbf{A} = \mathbf{0}$
plane $\mathbf{B}$	(*1*)	$\mathbf{B}^\top\mathbf{X} = 0$	(*2*)	$\Pi(\mathbf{B})\mathbf{L} = \mathbf{0}$	(*3*)	$\Pi^\top(\mathbf{B})\mathbf{A} = \mathbf{0}$

Table 1.2: Incidence relations. The number of degrees of freedom fixed by a constraint is equal to the number of independent equations in the expression.

distance	point $\mathbf{Y}$	line $\mathbf{L}$	plane $\mathbf{B}$
point $\mathbf{X}$	$\|\mathbf{X}_0 - \mathbf{Y}_0\|$	-	-
line $\mathbf{L}$	$\|\mathbf{Y}_0 \times \mathbf{L}_h - \mathbf{L}_0\|$	$\|\mathbf{M}_0 - \mathbf{L}_0\|$; ($\mathbf{M}_h \parallel \mathbf{L}_h$); $\frac{\|\mathbf{M}_0^\mathsf{T}\mathbf{L}_h + \mathbf{L}_0^\mathsf{T}\mathbf{M}_h\|}{\|\mathbf{M}_h \times \mathbf{L}_h\|}$; otherwise	-
plane $\mathbf{A}$	$\|\mathbf{A}^\mathsf{T}\mathbf{Y}\|$	$\|\mathbf{A}_h \times \mathbf{M}_0 - A_0\mathbf{M}_h\|$	$\|A_0 - B_0\|$

Table 1.3: Equations induced by distance constraints, provided that $\|\mathbf{V}_h\| = 1$. The distance between two objects always fixes 1 degree of freedom. The distance between two planes or a line and a plane can only be defined if they are parallel. The distance between two non-parallel lines is derived using the fact that the distance between lines is the distance between two parallel planes containing the lines. The corresponding expression is composed of two terms corresponding to the signed distances of the two planes from the origin.

1. through the equations specific to object type, given in table 1.2;
2. associating the coordinates the objects: $\mathbf{V}_1 := \mathbf{V}_2$

The distance relations are summarized in table 1.3.

Angles. Angles are not projective or affine invariants and are only used here between lines and planes. They can be expressed in terms of the homogeneous subvectors $\mathbf{V}_h$ of vectors $\mathbf{V}$ representing lines or planes. In the following we describe in brief equations related to angles between directions.

Parallelism Two directions $\mathbf{V}_{1h}$ $\mathbf{V}_{2h}$ are parallel when their vector representations are equal up to a scalar factor: $\mathbf{V}_{1h} \sim \mathbf{V}_{2h}$. The parallelism, in the same way as the incidence of two objects of the same typem can be expressed in two ways:

1. through the equation: $[\mathbf{V}_{1h}]_\times \mathbf{V}_{2h} \sim \mathbf{0}_3$;
2. associating the coordinates of the objects: $\mathbf{V}_{1h} := \mathbf{V}_{2h}$

The parallelism constraints are summarized in table 1.4.

Orthogonality Two directions $\mathbf{V}_{1h}$ $\mathbf{V}_{2h}$ are parallel when the vector product of their vector representations is null: ${\mathbf{V}_{1h}}^\mathsf{T}\mathbf{V}_{2h} = 0$. It fixes 1 degree of freedom between the vectors.

Parallelism between a line and a plane is equivalent to orthogonality between the line direction and the plane normal vector. On the other hand, the line and plane are orthogonal when the line direction and plane normal vector are identical.

Other angles General angle constraints are summarized in table 1.5.

$\parallel$	unknown			
known $\downarrow$	*(dof)*	line $\mathbf{L}$	*(dof)*	plane $\mathbf{A}$
line $\mathbf{M}$	*(2)*	$[\mathbf{M_h}]_\times \mathbf{L_h} = \mathbf{0}$ *or* $\mathbf{L_h} := \mathbf{M_h}$	*(1)*	$\mathbf{M_h}^\top \mathbf{A_h} = 0$
plane $\mathbf{B}$	*(1)*	$\mathbf{B_h}^\top \mathbf{L_h} = 0$	*(2)*	$[\mathbf{B_h}]_\times \mathbf{A_h} = \mathbf{0}$ *or* $\mathbf{A_h} := \mathbf{B_h}$

Table 1.4: Equations induced by parallelism constraints. The number of degrees of freedom fixed by a constraint is equal to the number of independent equations in the expression.

$\angle\alpha$	unknown			
known $\downarrow$	*(dof)*	line $\mathbf{L}$	*(dof)*	plane $\mathbf{A}$
line $\mathbf{M}$	*(1)*	$\mathbf{M}_h^\top \mathbf{L}_h = \cos\alpha$	*(2)*	$\mathbf{M}_h^\top \mathbf{A}_h = -\sin\alpha$
plane $\mathbf{B}$	*(2)*	$\mathbf{B}_h^\top \mathbf{L}_h = -\sin\alpha$	*(1)*	$\mathbf{B}_h^\top \mathbf{A}_h = \cos\alpha$

Table 1.5: Equations induced by angle constraints. The number of degrees of freedom fixed by a constraint is equal to number of independent equations in the expression.

Affine point configurations. An affine point configuration (or affine shape) is defined by coordinates of points belonging to the configuration in an arbitrary affine coordinate system. These coordinates stay invariant under affine transformations. Affine point configurations were introduced first by Sparr [Sparr, 1991b,a, 1992b] for the reconstruction from single and multiple images. Here we use a different formulation of those constraints.

Relations like points lying on a parallelogram or the symmetry of point configurations are very useful in practice and can be described in a common framework. Generally, all the constraints of the type $\sum_{i=1}^{n} \alpha_i \mathbf{X_i} = \mathbf{0}$ can be expressed as:

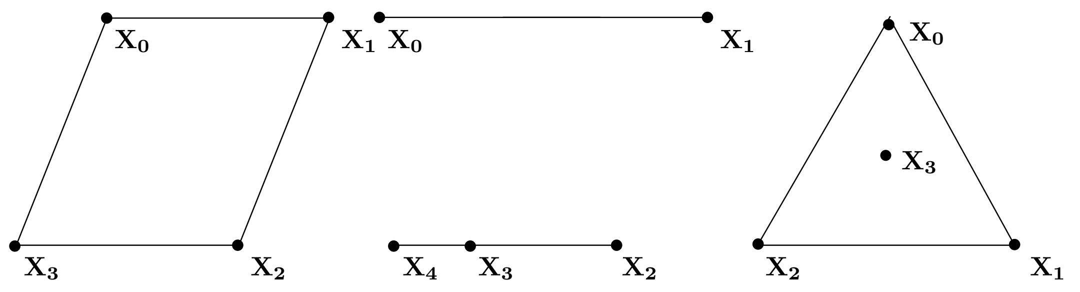
$$\begin{bmatrix} \alpha_1 \mathbf{e}_1^\top & \alpha_1 \mathbf{e}_1^\top & \cdots & \alpha_n \mathbf{e}_1^\top \\ \alpha_1 \mathbf{e}_2^\top & \alpha_1 \mathbf{e}_2^\top & \cdots & \alpha_n \mathbf{e}_2^\top \\ \alpha_1 \mathbf{e}_3^\top & \alpha_1 \mathbf{e}_3^\top & \cdots & \alpha_n \mathbf{e}_3^\top \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{bmatrix} = \mathbf{0}; \qquad (1.15)$$

For example, constraints arising from a parallelogram consisting of vertices $\mathbf{X_0} \cdots \mathbf{X_3}$, can be written in the form (1.15), with $n = 4$ and setting $\alpha_0 = \alpha_2 = -\alpha_1 = -\alpha_3$. The coefficients α_i remain constant under any affine transformation of the parallelogram. Other examples are shown in figure 1.8. All the constraints of this type give three linearly independent equations in all involved points.

Constraints expressed using known directions. When some directions in space are known, it becomes possible to express the distances between points in a linear framework. Using known lengths on lines with known directions was used for example in [Shum et al., 1998]. Different possibilities of expressing symmetries were proposed in [Grossmann and Santos-Victor, 2002]. Figure 1.9 shows some examples of using this information.

1.4.2 Complex Primitives

Using complex primitives allows enclosing the major scene features in a compact representation. This reduces the complexity of the human interaction needed to define scene constraints and computation. Indeed, the user has to depict in the image only the corners of the predefined primitive, after which and then only its shape parameters have to be computed. On the other



$$\mathbf{X_0} - \mathbf{X_1} + \mathbf{X_2} - \mathbf{X_3} = 0$$

(a)

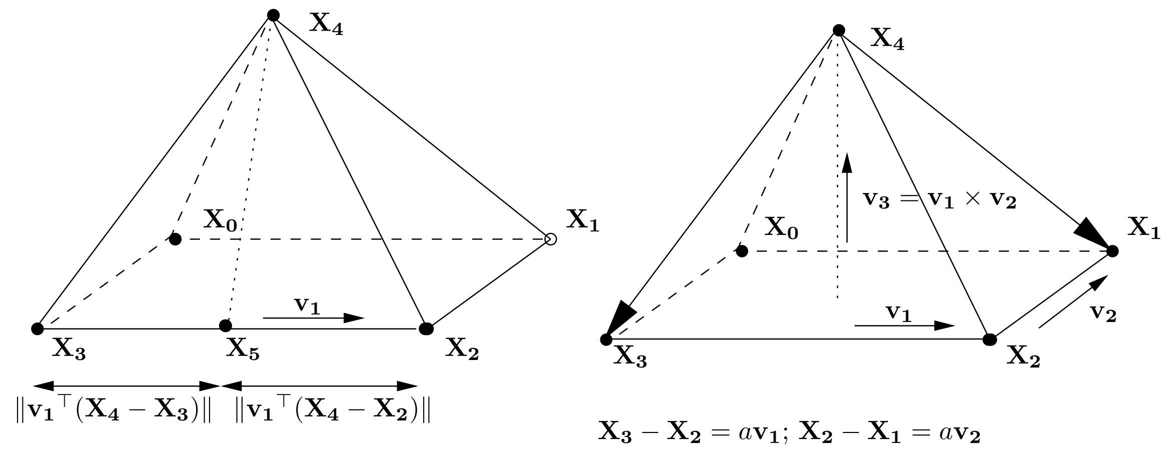
$$\mathbf{X_4} - \mathbf{X_3} = \tfrac{1}{2}(\mathbf{X_3} - \mathbf{X_2}); \quad \mathbf{X_4} - \mathbf{X_3} = \tfrac{1}{4}(\mathbf{X_0} - \mathbf{X_1})$$

(b)

$$\mathbf{X_0} + \mathbf{X_1} + \mathbf{X_2} - 3\mathbf{X_3} = 0$$

(c)

Figure 1.8: Examples of affine point configurations. (a) Parallelogram; (b) Known distance ratios on parallel lines; (c) Triangle centre. All known configurations of this type give 3 equations in terms of points coordinates (x_i, y_i, z_i).



$\mathbf{X_0}$
$\mathbf{X_1}$
$\mathbf{X_2}$
$\mathbf{X_3}$
$\mathbf{X_4}$
$\mathbf{X_5}$
$\mathbf{v_1}$
$\|\mathbf{v_1}^\top(\mathbf{X_4} - \mathbf{X_3})\|$
$\|\mathbf{v_1}^\top(\mathbf{X_4} - \mathbf{X_2})\|$
$\mathbf{v_3} = \mathbf{v_1} \times \mathbf{v_2}$
$\mathbf{v_2}$

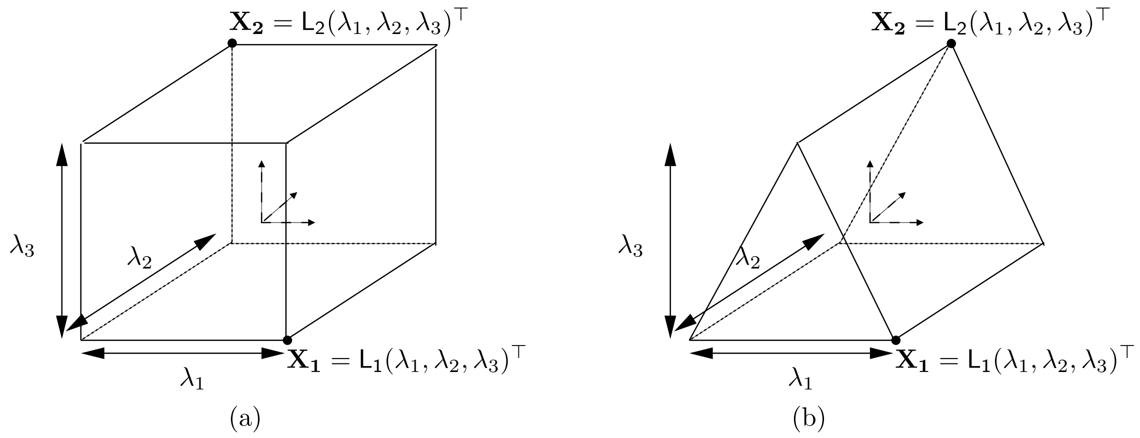
$$\mathbf{v_1}^\top(\mathbf{X_4} - \mathbf{X_3}) = -\mathbf{v_1}^\top(\mathbf{X_4} - \mathbf{X_2})$$

(a)

$$\mathbf{X_3} - \mathbf{X_2} = a\mathbf{v_1}; \; \mathbf{X_2} - \mathbf{X_1} = a\mathbf{v_2}$$
$$\mathbf{v_3}^\top(\mathbf{X_4} - \mathbf{X_3}) = -\mathbf{v_3}^\top(\mathbf{X_4} - \mathbf{X_1})$$

(b)

Figure 1.9: Examples of using known scene directions to impose euclidean constraints: symmetry, distances and distance ratios: (a) Symmetry of vectors $\mathbf{X}_5\mathbf{X}_3$ and $\mathbf{X}_5\mathbf{X}_2$ with respect to plane perpendicular to both vectors and passing through $\mathbf{X}_5$: projections of vectors $\mathbf{X}_3\mathbf{X}_4$ and $\mathbf{X}_4\mathbf{X}_2$ on direction $\mathbf{v}_1$ have the same length; (b) Symmetry of vectors $\mathbf{X}_4\mathbf{X}_3$ and $\mathbf{X}_4\mathbf{X}_1$: projections of vectors $\mathbf{X}_1\mathbf{X}_4$ and $\mathbf{X}_3\mathbf{X}_4$ on direction $\mathbf{v}_3$ have the same length. Distances: known distances on lines with known directions give linear constraints in terms of unknown points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$. When only the distance ratio is known, the expression remains linear in terms of unknown points and distance a.



$\mathbf{X_2} = \mathsf{L}_2(\lambda_1, \lambda_2, \lambda_3)^\top$
λ_3
λ_2
λ_1
$\mathbf{X_1} = \mathsf{L}_1(\lambda_1, \lambda_2, \lambda_3)^\top$
(a)
$\mathbf{X_2} = \mathsf{L}_2(\lambda_1, \lambda_2, \lambda_3)^\top$
λ_3
λ_2
λ_1
$\mathbf{X_1} = \mathsf{L}_1(\lambda_1, \lambda_2, \lambda_3)^\top$
(b)

Figure 1.10: Examples of linearly parameterized models [Debevec et al., 1996; Jelinek and Taylor, 2000]. (a) Cube; points $\mathbf{X}_1$ and $\mathbf{X}_2$ are parameterized by, respectively, matrices $\mathsf{L}_1 = \text{diag}(\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, $\mathsf{L}_2 = \text{diag}(-\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$; (b) Prism; points $\mathbf{X}_1$ and $\mathbf{X}_2$ are parameterized by, respectively matrices $\mathsf{L}_1 = \text{diag}(\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2})$, $\mathsf{L}_2 = \text{diag}(0, \frac{1}{2}, \frac{1}{2})$.

hand, using only complex primitives decreases the flexibility of a scene definition. Also the inter-primitive relations are often more complicated to model (this will be discussed further in section 2).

1.4.2.1 Linearly Parameterized Models

The classical approach for scene modelling from images using complex primitives was introduced to Computer Vision in the Façade system [Debevec et al., 1996] and extended in [Jelinek and Taylor, 2000]. The scene primitives are represented as *linearly parameterized models* (examples are given in figure 1.10). The position of every vertex $\mathbf{X}_i$ is defined as

$$\mathbf{X}_i = \mathsf{L}_i \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$$

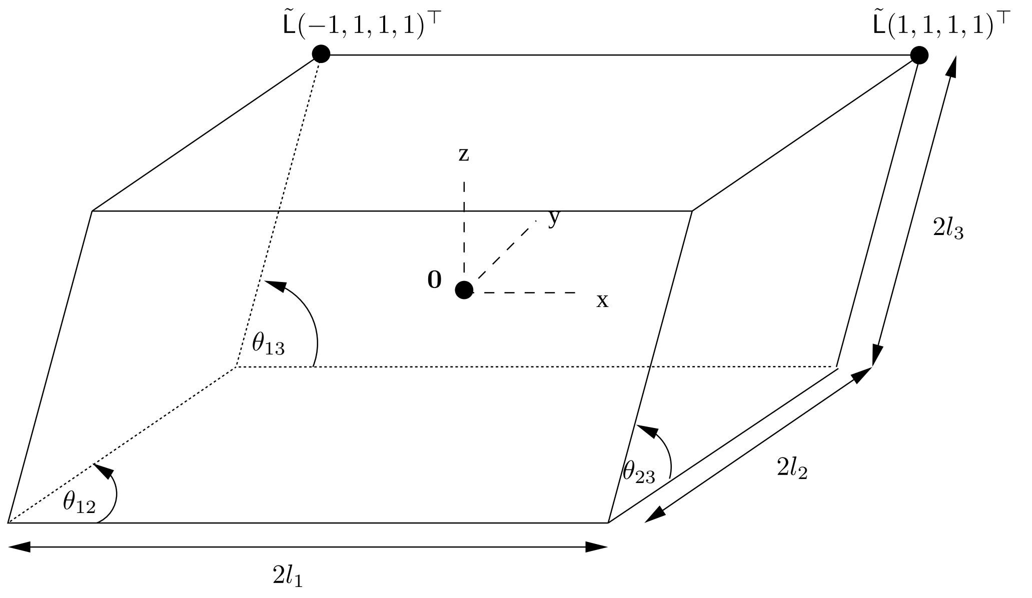
where the $3 \times n$ matrix L_i is *known* and depends on the vertex location on the primitive and λ is the primitive dimension vector. Thus only the dimension vector has to be estimated to obtain the model reconstruction.

1.4.2.2 Our Approach

We propose to represent a primitive by a *shape matrix*, whose parameters define metric shape of the primitive (e.g. angles and length ratios for a parallelepiped). All the vertices of a primitive can be computed by multiplying the shape matrix and a matrix containing the coordinates of vertices forming a *canonical primitive* (e.g. a cube for a parallelepiped).

As will be shown in part II, contrary to the previous approach, this allows using information about the primitive in a linear framework.

Parallelepipeds. The scene constraints used for calibration in part II are modelled using a parallelepiped. It encodes naturally the affine properties of the scene and facilitates modelling the remaining metric part. The representation of a parallelepiped given below is based on the formalism proposed in [Wilczkowiak et al., 2001, 2002].



$\tilde{\mathsf{L}}(-1,1,1,1)^\top$
$\tilde{\mathsf{L}}(1,1,1,1)^\top$
z
y
x
0
$2l_3$
θ_{13}
θ_{23}
$2l_2$
θ_{12}
$2l_1$

Figure 1.11: Parameterization of a parallelepiped: $2l_i$ are the edge lengths; θ_{ij} are the angles between non-parallel edges.

A parallelepiped is defined by twelve parameters: six extrinsic parameters describing its orientation and position, and six intrinsic parameters describing its metric shape: three dimension parameters (edge lengths l_1, l_2 and l_3) and three angles between edges $(\theta_{12}, \theta_{23}, \theta_{13})$. These intrinsic parameters are illustrated in figure 1.11. They can be represented compactly in a triangular 4×4 matrix $\tilde{\mathsf{L}}$ (see section 3.1 for details), representing the affine transformation between a canonical cube and a parallelepiped with the given shape. Full transformation between the canonical cube and the world coordinate frame can be represented by matrix

$$\mathsf{N} = \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathsf{L}}$$

where S is a rotation matrix and $\mathbf{v}$ a vector, representing the parallelepiped's pose (extrinsic parameters). Thus, a vertex $(\pm1, \pm1, \pm1, 1)^\top$ of the canonical cube is mapped, by $\tilde{\mathsf{L}}$, to a vertex of our parallelepiped's intrinsic shape.

Other primitives. The same model can be used to parameterize other 3D objects defined by affine properties, such as a parallelogram and prism.

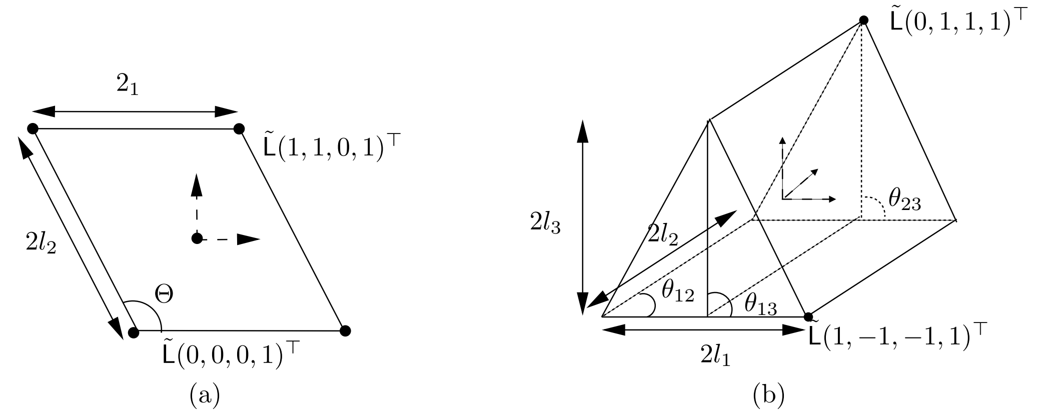

2_1
$\tilde{\mathsf{L}}(1,1,0,1)^\top$
$2l_2$
Θ
$\tilde{\mathsf{L}}(0,0,0,1)^\top$
(a)
$\tilde{\mathsf{L}}(0,1,1,1)^\top$
$2l_3$
$2l_2$
θ_{23}
θ_{12}
θ_{13}
$\mathsf{L}(1,-1,-1,1)^\top$
$2l_1$
(b)

Figure 1.12: Examples of primitive parameterisation. (a) Parallelogram; (b) Prism.

Chapter 2

A State of the Art on Using Geometric Constraints in Computer Vision

In this chapter we present different approaches for introducing the prior information on camera and scene parameters into the 3D model acquisition process. As mentioned in section 1.2, depending on the available information, a 3D scene can be reconstructed from 2D image data up to a certain transformation, for example, projective, affine, or metric. Using projective information, e.g. projections of points or lines in the images, only the projective nature of the scene can be recovered [Faugeras, 1992; Hartley et al., 1992]. Additional information enables us to locate the plane at infinity and the absolute conic in the projective reconstruction. As explained in the previous chapter, this allows to upgrade the reconstruction to, respectively, an affine or metric one. There is a substantial variety of information which can be incorporated into a 3D modeling process. It can be simple knowledge about camera intrinsic parameters or pose (camera moving by pure rotation, pure translation, etc.) or on 3D scene structure (calibration patterns); it can also be the information about scene elements such as points, lines and planes, as well as more complex primitives like cubes, prisms, cylinders. Nonetheless, note that whatever the information is, it can often be used at different stages of the 3D modeling process, including the initial intrinsic calibration, the pose estimation, the model reconstruction or an additional non-linear adjustment of the initial estimate at every step. Incorporation of geometrical constraints allows to disambiguate many calibration scenarios (rotating cameras, planar point configurations) singular for the generic calibration algorithms and allows 3D reconstruction even when essential parts of the model are occluded in the images. Sometimes using prior information about scene structure also simplifies the algebraic complexity of problems and allows solutions based on linear algebra. Finally, incorporation of prior information on the scene reduces significantly number of images needed for the reconstruction, making possible even a single-image reconstruction. In this chapter, different approaches for using constraints will be summarized briefly.

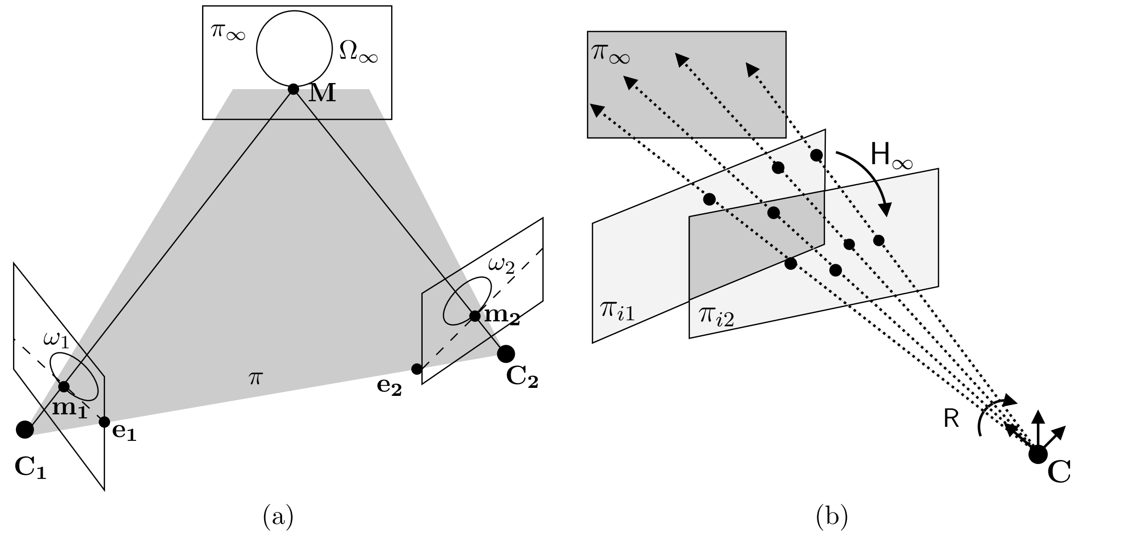


Figure 2.1: (a) Interpretation of Kruppa equations. An epipolar plane tangent to the absolute conic Ω_∞ defines corresponding epipolar lines which are tangent to the absolute conic images ω_1, ω_2. When the camera parameters are constant ($\omega \sim \omega_1 \sim \omega_2$) epipoles $\mathbf{e}_1$, $\mathbf{e}_2$ can be used to constrain ω. (b) Principle of calibration of rotating cameras. All the corresponding points are related by infinite homography H_∞.

2.1 Camera Constraints

In this section we provide a brief overview of information about camera parameters useful for calibration and reconstruction. Evidently, simple information about camera intrinsic parameters give direct constraints on calibration matrices. However, constant camera intrinsics, pose, or orientation constrain the calibration and reconstruction process as well. We first review methods based on known or constant intrinsic camera parameters, and subsequently, based on special camera motions. We finish with methods splitting the self-calibration process into two steps, consisting on localization, in order of the plane at infinity, and the absolute conic.

2.1.1 Information on Intrinsic Parameters

The idea of camera self-calibration has first appeared in [Faugeras et al., 1992] and [Maybank and Faugeras, 1992] in the context of calibration of cameras with *constant* intrinsic parameters. The camera parameters are recovered using the Kruppa equations. They are based on the observation that the epipolar planes [1] tangent to the absolute conic project to epipolar lines tangent to the image of the absolute conic in each view (see figure 2.1(a)).

In [Triggs, 1997] the self-calibration problem is formulated in terms of the absolute quadric. The projection of the absolute quadric has to be constant in each view (see figure 1.4), which, for every image pair gives five independent constraints on the calibration parameters. [Pollefeys and Van Gool, 1997] introduces the modulus constraint, based on the fact that with two identical cameras the infinite homography is conjugated to a rotation, thus has eigenvalues of equal modulus. It is then possible to formulate quadratic constraints on the plane at infinity equation.

[1]In this thesis we do not directly use, and consequently, do not introduce, in section 1 the epipolar geometry. A good overview can be found in [Hartley and Zisserman, 2000].

In practice, camera parameters do not always remain constant over the sequence, especially when using several photos rather than a video sequence. In [Heyden and Åström, 1997], is proposed a non-linear self-calibration approach based only on known values of camera skew and aspect ratio. [Pollefeys et al., 1998] proves that the information about the zero skew ratio only is sufficient, and proposes a non-linear optimization procedure. The linear estimation of initial parameters is computed using approximate positions of principal points.

2.1.2 Special Camera Motions

Besides known or constant intrinsic camera parameters, information about a camera's movements can also be used for calibration. The idea that, in cameras with constant orientation (thus purely translating) the infinite homography is the identity, was used in [Moons et al., 1993; Armstrong et al., 1994] for an affine reconstruction. Also a less restricted planar motion leaves three particular points at infinity constant over the sequence. Locating them is equivalent to the location of the plane at infinity and an affine reconstruction [Armstrong et al., 1996; Faugeras et al., 1998].

It has been shown that a setup consisting of purely rotating cameras has interesting properties as well. In the absence of translation there is no depth information enclosed in the point projections (see figure 2.1–(b)). On one hand, it makes a 3D reconstruction impossible. On the other hand, it means that homography exists between each pair of views: the infinite homography. Thus, simply using points matched between the images it is possible to compute infinite homographies between all views. Then they can be used to formulate calibration constraints in the case of the constant [Hartley, 1994] or varying [de Agapito et al., 1999] values of certain camera parameters.

2.1.3 Stratified Approaches

Independently from the used constraints, most of the self-calibration algorithms [Hartley, 1993; Armstrong et al., 1994; Pollefeys and Van Gool, 1997], tend to split the metric reconstruction process into three steps, recovering, in order, the projective, affine and Euclidean strata, the projective-affine step being considered as the most non-linear and thus most difficult step. It is due to the fact, that without any information about the infinite plane, most of the constraints on the calibration parameters which can be delivered on the camera intrinsic parameters are non-linear [Triggs, 1997; Heyden and Åström, 1997; Pollefeys et al., 1998]. Once the infinite plane is known, it is possible to use the infinite homographies to incorporate directly the prior knowledge on camera parameters in a linear framework, in a way analogous to the methods proposed for rotating cameras [Hartley, 1994; de Agapito et al., 1999].

Unfortunately, self-calibration algorithms suffer from critical motion sequences for which there is no unique solution [Sturm, 1997]. To get stable results with self-calibration, a large number of images in general positions is usually necessary. To solve the remaining ambiguities it is advantageous to incorporate prior knowledge about the scene [Zisserman et al., 1998; Liebowitz and Zisserman, 1999].

2.2 Scene Constraints

2.2.1 Simple Primitives and Constraints

2.2.1.1 Points and Lines

Points and lines are the basic features used in the 3D modelling from images: it is relatively easy to extract and match them automatically in the images, as well as constrain them using simple

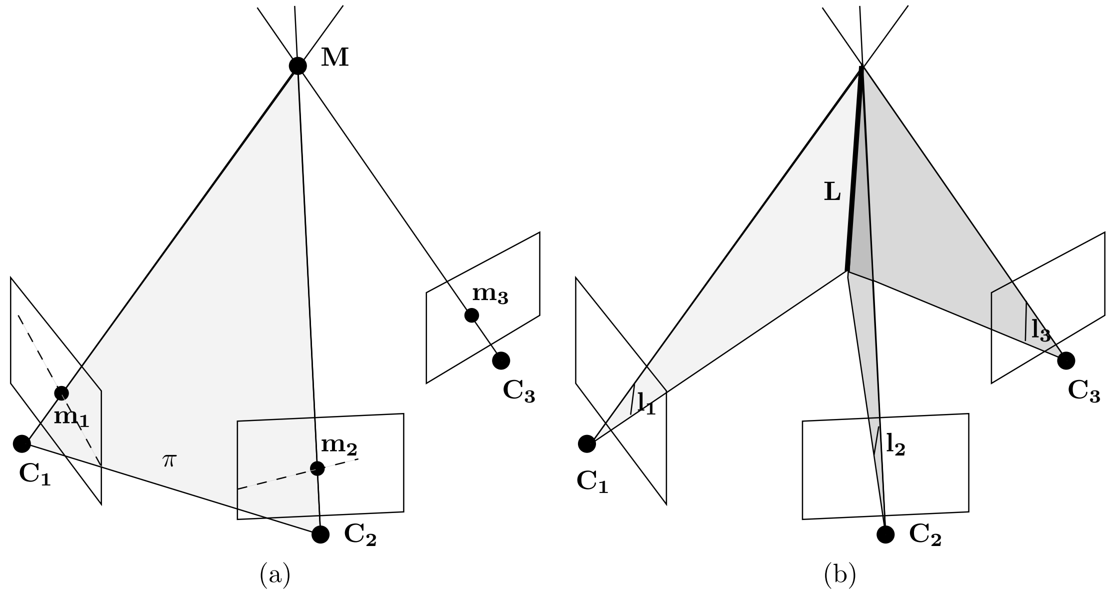


Figure 2.2: (a) Relations engendered by images of a point in two and three views. Points $\mathbf{M}$, $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{m}_1$, $\mathbf{m}_2$ are coplanar (they form the epipolar plane π). Point $\mathbf{m}_2$ must lie on the epipolar line of point $\mathbf{m}_1$. Lines $\mathbf{C}_1\mathbf{m}_1$, $\mathbf{C}_2\mathbf{m}_2$, $\mathbf{C}_3\mathbf{m}_3$ intersect in point $\mathbf{M}$. Point $\mathbf{m}_3$ might be computed by intersecting the epipolar lines of points $\mathbf{m}_1$ and $\mathbf{m}_2$ in the third image unless it belongs to the plane $\mathbf{C}_1\mathbf{C}_2\mathbf{C}_3$. The trifocal geometry describes the geometry of three images simultaneously. (b) The three backprojection planes of a line must intersect in one space line.

geometrical relations. In the following we summarize point and line properties useful for calibration and reconstruction algorithms.

3D positions. The classical calibration approach consists of the computation of the projection matrix based on correspondences between the 3D world coordinates of points on a calibration object and their 2D positions in the image. Such a computed matrix represents the transformation between a 3D and an image frame. Thus, as mentioned in section 1.3, it can be decomposed simply using the QR decomposition [Tsai, 1986; Faugeras and Toscani, 1986] resulting in the camera intrinsic parameters as well as its orientation and position in the world coordinate frame.

Projections. Projections of 3D points and lines in images are information of projective nature, thus without any other information the scene structure and projection matrices can only be recovered up to a projective transformation [Faugeras, 1992; Hartley et al., 1992]. The fundamental projective relation is that lines and planes back-projected from matching image points and lines must intersect in the 3D space (see figure 2.2). The multi-view dependencies can be described using tensor calculus (fundamental matrix, trifocal and quadrifocal tensor) [Triggs, 1995; Hartley, 1995, 1997a], or using the Grassmann-Cayley (double) algebra [Carlsson, 1993; Faugeras and Mourrain, 1995]. The tensors can be computed directly from the image correspondences, and then used directly to compute projection matrices and 3D feature positions.

When dealing with more than four images, the data furnished by image pairs, triplets or quadruples, can be registered into a common framework using sequential [Avidan and Shashua, 1998; Beardsley et al., 1996] or hierarchical [Fitzgibbon and Zisserman, 1998] methods. To use all the available data simultaneously, the factorization-based approaches has been proposed [Tomasi and Kanade, 1992; Sturm and Triggs, 1996; Sparr, 1996; Triggs, 1996]. Indeed, a *measurement matrix* composed of all the image-point or -line projections can be factorized into two parts, corresponding respectively to camera projection matrices and 3D point positions. Before the factorization, however, two conditions must be satisfied: the measurement values must be scaled correctly and defined for every feature-image pair. Various strategies exist for satisfying these two conditions. First, the factorization algorithm was proposed for the affine camera model [Tomasi and Kanade, 1992], where the problem of ambiguous scalar factors is easy to solve. [Sturm and Triggs, 1996] computes the scalar factors for the projective model using the epipolar geometry. [Sparr, 1996; Triggs, 1996] introduce an iterative method for the estimation of these factors. As for dealing with the missing data, [Tomasi and Kanade, 1992] propose choosing the maximal complete submatrix of the measurement matrix to compute an initial solution, and then propagate the computed information. However, finding such a maximal matrix is an NP-complete problem. [Jacobs, 1997] introduces an approach based on the fact that for every triple (n-tuple in general) of columns in the original matrix, the space spanned by all possible completions of these columns must contain the column space of the completely filled matrix. Solutions satisfying these partial constraints can be found using standard techniques of linear algebra, however the size of matrices that have to be dealt with increases very quickly with the amount of missing data. Since, different approaches combining the above ideas were proposed for both points and lines represented in various ways [Quan and Kanade, 1996; Triggs, 1996; Morris and Kanade, 1998; Martinec and Padjla, 2002; Martinec and Pajdla, 2002].

Vanishing points. A vanishing point is the image of an intersection of a point at infinity. Thus, it can be computed from the projections of a set of parallel lines. Several properties make the vanishing points interesting for calibration and reconstruction algorithms:

- Three vanishing points matched across the images define the plane at infinity which allows to establish an affine reconstruction and facilitates formulation of calibration equations (section 2.1.3).
- Vanishing points of orthogonal directions are conjugate with respect to the Image of Absolute Conic, which results in linear expressions on the IAC. This property is useful especially in calibration of small sets of images, e.g. a single image. Vanishing points were first used for a camera calibration in photogrammetry [Gracie, 1968]. Since the seminal work of Caprile and Torr [Caprile and Torre, 1990], they were commonly exploited in computer vision algorithms [Debevec et al., 1996; Cipolla and Boyer, 1998; Jelinek and Taylor, 2000; Liebowitz and Zisserman, 1999; Svedberg and Carlsson, 2000]. When the scene can be characterised by three dominant orthogonal directions, it is possible to estimate automatically up to three camera intrinsic parameters and rotation form single images [Coughlan and Yuille, 1999] or sequences [Kosecka and Zhang, 2002].
- Knowing the plane at infinity and vanishing points, it is possible to compute line directions up to a global affine transformation, and when cameras are calibrated, up to a euclidean transformation. This can be done even before the scene structure is recovered and is commonly used in reconstruction approaches ([Shum et al., 1998; Criminisi et al., 2000; Grossmann and Santos-Victor, 2002; Rother, 2003a]).

2.2.1.2 Planes

Planes, differently from points and lines, can not be represented by a single image feature. However, identification of image correspondences of coplanar points can furnish very useful constraints for

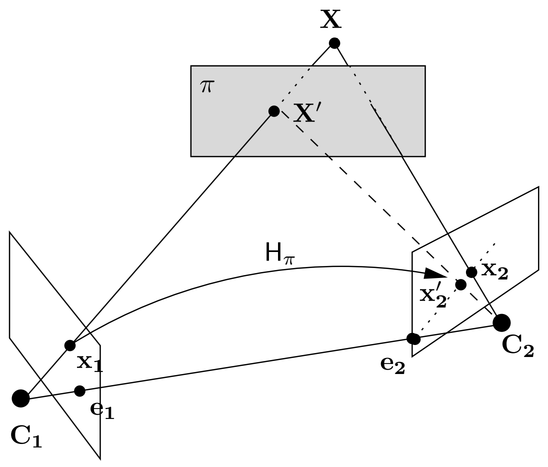


Figure 2.3: The concept of plane induced parallax. Point $\mathbf{x}_2' = \mathsf{H}_\pi \mathbf{x}_1$ is the image of point $\mathbf{X}'$ lying on the intersection of ray $\mathbf{C}_1\mathbf{x}_1$ and plane π. It is collinear with the point $\mathbf{x}_2$ being the true image of point $\mathbf{X}$ and the epipole $\mathbf{e}_2$. The vector $\mathbf{x}_2\mathbf{x}_2'$ is the parallax relative to homography H_π.

camera self-calibration and reconstruction.

Inter-image homographies. Knowing four inter-image correspondences between coplanar points it is possible to compute homographies induced by the underlying plane. Those homographies have many interesting properties useful for calibration and reconstruction algorithms:

- The homography induced by a plane, when applied to points not lying on the plane, generates a virtual parallax (see figure 2.3–(a)). Its specific properties can be used to simplify the computation of the fundamental matrix [Luong and Faugeras, 1993; Szeliski and Torr, 1998]. The definition of a reference plane in an image sequence simplifies also the reconstruction task, whether using linear algorithms [Kaucic et al., 2001; Hartley and Zisserman, 2000; Rother and Carlsson, 2001, 2002; Rother, 2003b], or factorization methods [Kumar et al., 1994; Triggs, 2000; Rother et al., 2002]. The reconstruction computed this way is generally of a projective nature. When using the affine camera model the obtained reconstruction is affine.

- Let us consider homographies H_k^{ij}, H_l^{ij} between two images i, j induced by two planes k, l. A relative homography $\mathsf{H}_{kl}^{ij} = {\mathsf{H}_k^{ij}}^{-1}\mathsf{H}_l^{ij}$ was proven to be a planar homology [Semple and Kneebone, 1952], i.e. two of its eigenvalues are equal. This fact can be used to stabilize the homography computation [Zelnik-Manor and Irani, 2002], compute the fundamental matrix [Johansson, 1999] and estimate the euclidean reconstruction from two views [Xu et al., 2000].

- Metric information about planar structures is very useful for calibration. Indeed, every plane contains two points which are invariant under metric transformations: the circular points (see section 1.2.3). The circular points lie on the intersection of the plane with the absolute conic. Therefore, their projections must lie on the absolute conic's image. Once the inter-image homographies are known, the images of circular points can be propagated between the images. Known circular points of a plane give two constraints on the IAC per plane-image pair. This observation lies at the ground of several calibration methods. When the metric structure of a plane is known (thus circular points are known), the plane-image homographies can be computed directly and used as constraints on the IAC [Sturm and Maybank, 1999a; Zhang, 1999; Gurdjos and Payrissat, 2001]. [Liebowitz and Zisserman, 1998] show how, when the metric structure of a plane is not known directly, it can be rectified using the accessible metric information, for example on angles between lines in the plane. Even when no metric information is given, it is still possible to use planar structures for self-calibration. Several methods for the non-linear self-calibration process based on the properties of the circular points projections were proposed [Triggs, 1998; Malis and Cipolla, 2002; Gurdjos and Sturm, 2003]. The most delicate point of those methods is the initialization step.

- When the metric structure of a plane and the intrinsic parameters of a camera are computed it is easy to compute the positions and orientations of the cameras and planes in a common frame [Sturm, 2000].

- The coplanarity constraints are often incorporated in a final non-linear optimization step. Such methods specific to planes are discussed, for example, in [Szeliski and Torr, 1998; Xu et al., 2000; Bartoli and Sturm, 2001]. An overview of methods dealing with a larger variety of constraints, including coplanarity, is given in section 2.2.1.3.

Maps. A map can be considered as an orthographic projection of a scene onto a ground plane. Including it into the image sequence as an affine image simplifies the calibration and reconstruction tasks [Zhang et al., 1999; Navab et al., 2003; Bondyfalat et al., 1999; Robertson and Cipolla, 2002]. Also, additional information furnished by maps, such as parallelism and orthogonality of scene directions can be included into the calibration and reconstruction process [Bondyfalat et al., 1999; Robertson and Cipolla, 2002].

Vanishing line. A vanishing line is the image of an intersection of a line at infinity. The vanishing points of lines on a world plane lie on the vanishing line of the plane. Two or more such points define the vanishing line. [Liebowitz and Zisserman, 1998] uses the plane vanishing lines to rectify the plane structure up to an affine stratum. [Criminisi et al., 2000] uses the vanishing line of a reference plane together with a vanishing point of a reference direction(not parallel to the plane) to compute affine properties, such as distance ratios on, and between, planes parallel to the reference plane. Given the plane at infinity and vanishing lines it is possible to compute plane normals (up to a global affine transformation), which was used by [Rother, 2003a] among others for the simultaneous reconstruction of camera and plane positions. When the camera is calibrated, the vanishing line defines the plane normal in the euclidean frame. This is used in many algorithms for reconstruction of piecewise planar scenes [Shum et al., 1998; Sturm and Maybank, 1999b; Robertson and Cipolla, 2000; Grossmann and Santos-Victor, 2002].

2.2.1.3 Geometrical Constrains

The way scene's geometrical properties are modeled depends on many factors, including the object representation, whether the constraints should be hard or soft, the scene size, the required computation speed. While it is relatively easy to find an optimal representation when using a small number of objects and constraint types, it is much more difficult to find a general representation which can deal with multiple types of primitives and constraints in a common framework. In the previous section we have presented properties of points, lines or planes projections useful for calibration and reconstruction. In this section we briefly describe strategies for dealing simultaneously with those objects and different kinds of constraints between them.

We describe first approaches based on linear algebra. Introducing the constraints allows obtaining quickly a model respecting approximately the geometrical constraints and avoiding the degeneracies of non-constrained methods. However, due to the multilinear nature of the problem, using only the linear algebra it is usually not possible to deal optimally with noise introduced by image projections and the solution obtained this way is not always satisfactory. That is why a non-linear structure and motion refinement is usually applied. Algorithms of this type are described in the second part of this section.

Approaches based on linear algebra. A simple source of information about the scene structure simplifying the reconstruction algorithms is the presence of a reference plane visible in all the images. [Triggs, 2000] suggests an algorithm for simultaneous factorization of points, lines, and planes, using the inter-image homographies induced by a reference plane. As in all factorization methods, the missing parts of a measurement matrix caused by features occluded in some images from the sequence have to be filled in. [Triggs, 2000] hallucinates the missing projections. [Rother, 2003a] uses the reference plane to reconstruct points, lines and planes and camera positions in a linear framework. First the vanishing points of lines and vanishing lines of planes are computed, in order to recover the directions in a common frame. Then, linear constraints on the remaining unknowns, such as positions of the primitives and cameras, are applied, forming a constraint matrix. Finally, the solution is obtained as a null vector of the constraint matrix. The algorithm results generally in a reconstruction of projective nature, except in the case where the reference plane is the plane at infinity, and the reconstruction is of affine nature. In spite of the fact that the reconstruction of all primitives is done simultaneously, there is no possibility to impose incidences between the different scene primitives. It is also not clear how the system deals with underconstrained features.

In their presence the dimension of the nullspace of the constraint matrix is bigger then 1 and the reconstruction is ambiguous. The method offers no possibility of detecting such situations.

An approach for imposing point and line incidences together with parallelism and orthogonality constraints on line directions was presented in [Robertson and Cipolla, 2000]. First scene structure is approximatively estimated in unconstrained bundle adjustement process. Then lines are grouped into parallel sets and for every set the direction fitting optimally the image data is computed. The orthogonality constraints are imposed via an SVD decomposition of matrices containing triples of orthogonal directions. Next, the computed directions are used together with the image data to constrain the scene points. This method is the base of PhotoBuilder, system for semi-automatic modeling of man-made environments.

One of the first methods dealing with point, line, and plane features and constraints between them appeared in [Shum et al., 1998], in the context of single or multiple view reconstruction from panoramic images. The input consists of precomputed intrinsic camera parameters and rotations as well as line and plane normal directions computed using the camera parameters and the vanishing points and lines. First, parallelism and orthogonality constraints can be used to adjust the scene directions. Then equations representing coplanarity, collinearity, and distances on parallel directions are collected into a constraint matrix. Depending on the requirements, the geometric information can be included into the system as either hard or soft constraints. Indeed, the matrix including equations generated only by the hard constraints can be decomposed via QR decomposition (SVD might be used as well) to generate the multidimensional linear space of solutions respecting exactly the constraints. Then the remaining model parameters can be fitted in the least squares sense to the soft constraints, again using linear methods. In order to check if the furnished data are consistent and provide a unique solution, a test based on counting the number of equations and variables, as well as zero values in the right hand vector of the equation system, is proposed. However such a test is not sufficient to detect degeneracies caused by singular camera positions or redundant constraints. Indeed, in such cases, the lack of data is not caused by an insufficient number of equations but by linear dependencies between them (examples are shown

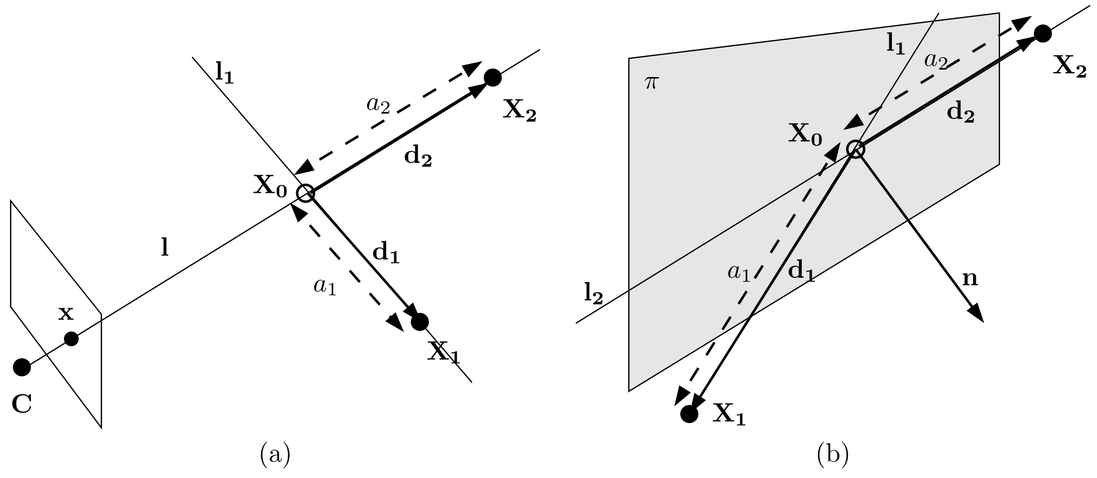


Figure 2.4: Examples of singularities of scene definition using points, lines and planes. (a) Let lines $\mathbf{l}$, $\mathbf{l}_1$ and points $\mathbf{X}_1$, $\mathbf{X}_2$ are given. Point $\mathbf{X}_0$ can be computed using the backprojection line $\mathbf{l}$ and the collinearity constraint $\mathbf{X}_1 - \mathbf{X}_0 = a_1\mathbf{d}_1$. However, using the same set of constraints involving $\mathbf{X}_2$ and direction $\mathbf{d}_2$ leads to dependent set of equations. (b) Generally, a point can be defined as intersection of a line and a plane (line $\mathbf{l}_1$ and plane π for example). When the line direction is parallel to the plane however, this definition is ambiguous (line $\mathbf{l}_2$ and plane π for example).

in figure 2.4).

A different approach for dealing with the underconstrained data in a similar context was presented in [Grossmann and Santos-Victor, 2002; Grossmann, 2002]. The constraint matrix is created using the same input as in the previous method. Additionally, symmetry constraints with respect to lines with known directions are allowed. The system is not degenerated when the corank of the constraint matrix is 4 (scale factor plus a global translation). Usually in practice this is not the case due to noisy image data. That is why the authors propose to generate randomly "perfect" image data using randomly distributed cameras and test the rank of the constraint matrix based on this data. When the corank of such constructed matrix is 4, the SVD of the matrix containing real measurements gives a solution for all the features simultaneously. In contrary to the previous approach, such a test can detect singularities caused by underconstrained definitions of scene objects, provided that there is no degeneracy caused by camera positions (see figure 2.4–(b)). As the camera information used for the test is not the true one, when a point is defined, for example, as in figure 2.4–(a), the system will not take it into account.

All the mentioned methods need the directions of all the lines and planes involved in the reconstruction to be precomputed, which usually requires an additional interactive step. Also, only the objects which are initially involved in a sufficient number of constraints with precomputed objects can be computed. Sometimes it is advantageous to proceed in iterations, and propagate the values computed in the previous steps in order to reconstruct the remaining objects. A simple example where such a procedure would increase the number of reconstructed objects is given in figure 2.5. [Poulin et al., 1998; Heuel, 2001; Sturm and Maybank, 1999b] propose approaches which propagate sequentially the accessible information. They exploit the fact that basic relations between objects, such as incidence, parallelism, orthogonality, are bilinear in terms of the involved objects. This implies that if coordinates of one object involved in the bilinear relation are known, it gives linear constraints on the second related object. Thus, iteratively, the known information can be propagated in the scene, leading to the reconstruction of all sufficiently constrained elements

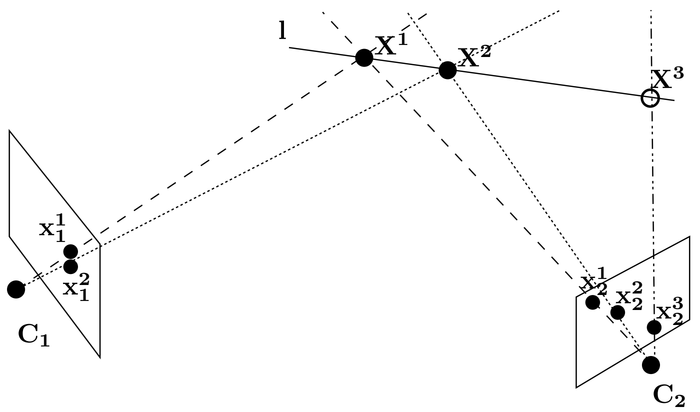


Figure 2.5: Points $\mathbf{X}^1$, $\mathbf{X}^2$, $\mathbf{X}^3$ lie on line $\mathbf{l}$. When a line equation is not given, at first only points $\mathbf{X}^1$ and $\mathbf{X}^2$, visible in both images, can be computed. In the second step it would be possible to compute the line $\mathbf{l}$ coordinates, and, subsequently, the point $\mathbf{X}^3$.

of the scene. [Poulin et al., 1998] proposes also to alternate the reconstruction step with a camera re-calibration step.

A method for propagation of the computed data was proposed also in the Constraint Programming community in the Desargues system [Lhomme et al., 1997]. It uses local propagation techniques to obtain a 3D scene consistent with 2D sketches and user-provided constraints. Using predefined geometrical rules, the known object coordinates are propagated in the system. Projection constraints are used in the same way as scene constraints, which can lead to problems when, for example, a point is defined in three images, and thus is overconstrained. The way that the rules are defined and values propagated does not assure that system will find a correct solution, even if one exists.

A drawback of sequential methods is that, together with the object coordinate values, the error is also propagated, which can decrease the quality of the reconstruction.

To avoid such a problem, [van den Heuvel and Vosselman, 1997; van den Heuvel, 1998] propose to start the reconstruction by adjusting the measurements to fit the prior information. The constraints like coplanarity, collinearity, parallelism, orthogonality are transformed into condition equations on observations (points and lines projections). The condition equations are first linearized and then used to adjust sequentially the observations. Finally, closed-form expressions involving the adjusted observations allow a sequential reconstruction of the 3D objects.

A principle of adjusting the measurements before a reconstruction appears also in [Sun et al., 2002] in the context of a factorization framework for an affine camera model. In a preliminary step, the measurement matrix is iteratively updated to satisfy the conditions engendered by collinearity, coplanarity and parallelism constraints. Then the factorization of the measurement matrix provides the feature coordinates satisfying the desired properties.

In section 6 we propose an iterative algorithm for the reconstruction of constrained scenes introduced originally in [Wilczkowiak et al., 2003a]. It is similar in spirit to [Grossmann and Santos-Victor, 2002; Shum et al., 1998; Poulin et al., 1998] in the sense that at every step all the accessible constraints are gathered into a common constraint matrix. It does not require the scene directions to be precomputed, however this information can be included into the system. As in [Shum et al., 1998] soft as well as hard constraints can be involved. All the objects sufficiently constrained at a given stage are computed simultaneously. Moreover, we propose a technique for detecting underconstrained features and excluding them from the reconstruction until enough objects constraining the unknown features are computed.

Non-linear methods. The most strightforward approach to satisfy geometrical constraints in a model is to represent them as a set of algebraic equations among real-valued model parameters and solve the system in a non-linear numerical optimization process. A classical approach consists on incorporation of penalties corresponding to unsatisfied constraints into the cost-function. An approach of this type is presented in [McGlone, 1995]. Constraints like coplanarity, collinearity, angles are incorporated in the bundle adjustment process. A detailed study on the effect of constraints on the final precision and the liability of solutions is provided. This solution does not guarantee however that the final model respects exactly the constraints. This problem can be dealt with by using constrained minimization techniques. [McLauchlan et al., 2000] use the Lagrange multipliers to impose the coplanarity constraints *exactly* on the model. Other constraints like parallelism can be added in the same way.

However, in a general case, the geometrical problem in 3D is translated into a large system of linear, bilinear or non-linear equations that are difficult to solve, which makes model reconstruction using purely numerical approaches unfeasible. That is why several authors searched for other ways to incorporate the constraints into the non-linear optimization.

Dedieu et al. in the Reality system [Dedieu et al., 2001] propose an ad-hoc method for adjusting the scene objects to the given constraints. After the primary camera calibration, the constraints are satisfied in an iterative approach, alternating the reconstruction and calibration steps. After each reconstruction step, the 3D points' positions are updated to satisfy the geometrical constraints. For example, if two 3D points do not respect the given distance constraints, they are arbitrarily moved into positions satisfying the constraint.

An original approach for the scene parameterization called the *dual representation* was proposed in [Grossmann and Santos-Victor, 2000]. The constrained scene points belonging to different sets of parallel lines and planes are represented by functions $\mathbf{f}^* : \mathbb{R}^{\mathbf{3}} \rightarrow \mathbb{R}$: the elements of the dual space to $\mathbb{R}^3$. Thus it is possible to split the scene representation into *shape* and its *parameters*. Then the *parameters* are computed using Maximum Likehood estimator.

Another group of authors tends to find a parameterization of the model satisfying all the given constraints and then optimize only on the remaining parameters. [Bazin, 2000] proposes a quasi-minimal parameterization of the scene containing constraints like collinearity, coplanarity, parallelism and orthogonality. The scene directions (line and plane normal directions) are combined in parallel sets represented by 2 common parameters. Then orthogonality constraints are imposed by removing parameters from the remaining set. Collinear points are represented by an origin point and vectors with common direction and coplanar points are represented by an origin point and vectors orthogonal to the plane normal. The choice of the point representing a plane or line is random. In consequence, especially when dealing with strongly constrained scenes, it is not sure that the minimal parameterization will be found, even if one exists.

[Cornou et al., 2003] uses the same model to parameterize complex primitives, like cubes, prisms etc. Such parameterized scenes are optimized using the Levenberg-Marquardt algorithm. [Cornou et al., 2003] presents also a fusion method for combining the pre-optimised structures from smaller image sequences into a global model.

An algorithm for the minimal scene parameterization inspired by theorem proving techniques is described in [D.Bondyfalat et al., 1999]. A propagation mechanism is used to place one object after the other. The constraints must be binary (relating at most 2 objects), and only linear or bilinear constraints are taken into account. Although efficient heuristics have been made to choice the next object to place, this algorithm cannot guarantee finding a correct order without a backtracking step, i.e. in a polynomial time. This approach is described in more details in section 12.1.

The same authors [Bondyfalat and Bougnoux, 1998] propose a minimal parameterization algorithm using theorem proving algorithms. A set of independent monomials characterizing the scene is found by the Wu-Ritt triangulation method. Then the whole model, satisfying naturally the imposed constraints, can be expressed as a polynomial function of the characteristic set. Optimization over variables containted in the characteristic set allows to find the model fitting the image data. As the authors admit, the algorithms for theorem proving are designed rather for small data sets and the complexity of computation increases very quickly with the model size, especially when dealing with constraints more complicated than bilinear. Also the polynomial reduction causes the necessity of dealing with polynomials of very high degree. As in every approach based on the elimination of model parameters, the system fails if redundant data was not removed from the system before the parameter elimination step.

In part III we present an original method for quasi-minimal scene parameterization. The scene is represented, as in [D.Bondyfalat et al., 1999] by an object-constraint graph. The algorithm uses a dictionary of predefined geometric rules to find a set of model parameters and the sequence of methods propagating the known model parameters until all the objects are reconstructed. Thanks to using advanced constraint decomposition techniques the propagation algorithm does not require the backtrack step and its complexity is linear. In the final step the model parameters are fitted to the image data. This algorithm was published in [Wilczkowiak et al., 2003d; Trombettoni and Wilczkowiak, 2003]

2.2.2 Model-Based Approaches

Model-based approaches are treated in this section, that is, approaches based on a scene description using primitives more complex than points, lines or planes. Using a compact model, naturally enclosing important scene features, it is possible to reduce the number of interactions needed for the scene definition and reduce the search space for the scene parameters. However, as applications of algorithms are limited to scenes which can be well described by the predefined model, the more complex are the primitives, the less flexible becomes the scene definition. We give several examples of models used to represent scene features for calibration and reconstruction. We focus on models enclosing relations considered in this thesis, such as parallelism, coplanarity or distances.

Affine shape. In a series of articles, Sparr introduces the notion of affine shape for N-point configurations (see section 1.4.1.2. Some examples of affine point configurations are shown in figure 1.8. Shape is defined as the nullspace of the measurement matrix containing image coordinates of points constituting the configuration. It is invariant under affine transformations. Using its properties, eventually combined with available information about point configuration (coplanar configurations, parallelograms...), depths of points can be computed from single [Sparr, 1992a] or multiple [Sparr, 1991b,a, 1998] images, up to a euclidean, affine or projective transformation.

Parallelograms, parallelepipeds. Special cases of affine point configurations, parallelograms and parallelepipeds, are very convenient for the modeling of basic scene properties. Very common in man-made environments and easy to depict for the user, they naturally enclose the information about parallelism in the scene. It is also easy to use them to impose orthogonality and length ratios. The information contained in a parallelepiped projection is usually sufficient to model scene elements from single images.

An original approach for single image modeling was presented in [Horry et al., 1997]. The scene is modeled by a kind of a rectangular corridor, with the observer at one, and the infinitely distant background at the other side. The central perspective and user-defined billboards are used to navigate inside the 2D image.

A classical approach for using the parallelepiped structure is to use the information about two adjacent rectangles [Svedberg and Carlsson, 2000] or cubes [Caprile and Torre, 1990; Cipolla and Boyer, 1998] to compute the vanishing points of orthogonal directions in the scene and calibrate the cameras from single images. However, decomposing the information enclosed in the parallelepiped into 3 directions and treating them separately causes loss of information. Indeed, the projection of a parallelepiped has only 11 degrees of freedom (see section 3 for details). Thus representation of the same information in terms of 12 degrees of freedom of 3 distinct vanishing points is overparameterized. It is particularly important in close-to-degenerated positions, e.g. when one of the parallelepiped's faces is parallel to the image plane. [Chen et al., 1999] proposes a method for simultaneous use of the projections of 6 parallelepiped vertices for camera calibration and reconstruction. The information about the parallelepiped angles is used to formulate polynomial constraints on camera intrinsic parameters. These polynomials are of degree 4 in case of a general parallelepiped and linear for a rectangular parallelepiped. Once camera intrinsic parameters are computed, the parallelepiped can be reconstructed in 3D in a separate step.

In this thesis we propose an approach which recovers simultaneously the camera's and parallelepiped's intrinsic parameters and rotations up to an affine transformation directly from the parallelepipeds vertices projections. Every known parameter of a camera or a parallelepiped can then be used simultaneously to upgrade the reconstruction to a euclidean one. Some of the results were published in [Wilczkowiak et al., 2001, 2002].

Complex primitives. Depending on the application, a variety of models of objects commonly present in man-made environments can be defined and located in the scene. In the Facade system [Debevec et al., 1996] the scene is defined using the user-provided projections of library objects, such as cubes, prisms, polygons. Every such primitive is defined by a set of dimension parameters, for example length, width and height for a cube, as well as by rotation and translation parameters. It is also possible to introduce inter-primitive relations, such as common orientation or base plane. The relation between two objects is defined by their relative rotation and translation. All the objects are grouped in a hierarchical tree structure, where links represent the inter-object constraints. The position of each node is defined as the composition of all the relative rotations and translations between objects situated higher in the hierarchy. Such a representation is more complex than dealing with the simple primitives described in the previous section. Also, it does not allow for loops in the object-constraint graph. This subject will be discussed further in part III. After the constraint analysis, the scene is reconstructed in a non-linear optimization process, where the remaining camera and primitive parameters are estimated. In [Jelinek and Taylor, 2000] a mathematical model of the scene parameterization via complex primitives is developed. Every primitive is represented by a linearly parameterized model: its coordinates can be expressed as a linear function of a dimension vector (see section 1.4.2.1 for examples).

In both approaches the initial calibration and reconstruction are done using parallelism and orthogonality constraints engendered by scene primitives. When enough constraints are provided, the reconstruction can be done using only a single image, which was used to create the commercial software Canoma [Schrand and Seidl, 2000].

Generally, model-based methods constraining strongly the scene are well adapted for single image applications. An interesting approach for reconstruction of human models from single orthographic images is presented in [Taylor, 2000]. Using stick-model of a human body with known distance ratios between the segments and eventual coplanarity and coplanarity constraints the pose of a human can be detected using close-form solutions.

Recently, several works [Dick et al., 2001; Werner and Zisserman, 2002] have been devoted to the automatic detection of predefined architectural elements in the scene in order to include them into reconstruction frameworks.

The use of different kinds of primitives has been investigated also in photogrammetry, especially in the context of (semi-) automatic reconstruction of cities from aerial images. [Henricsson et al., 1996; Brunn et al., 1998] propose approaches using roof and corner models, respectively. The 3D reconstruction is based on the hypothesize and verify paradigm: in the approximately reconstructed scene, first the image interest points and segments are used to create the corner (roof) hypotheses. The next step consists of the verification of the corner(roof) belonging to predefined classes. Additionally hard and soft constraints, like symmetry and orthogonality are introduced by the specific corner class definition and included into a non-linear optimization process.

Part II

Linear Approaches for Using Geometric Primitives for Calibration and 3D Modeling

Introduction

In this part of the thesis, we study linear approaches for 3D model acquisition from non-calibrated images. First, the intrinsic and extrinsic camera calibration is taken into consideration. In particular, we study the use of a specific calibration primitive: the parallelepiped. Parallelepipeds are frequently present in man-made environments and naturally encode the affine structure of the scene. Any information about their euclidean structure (angles or ratios of edge lengths), possibly combined with information about camera parameters is useful to obtain the euclidean reconstruction. We propose an elegant formalism to incorporate such information, in which camera parameters are dual to parallelepiped parameters, i.e. any knowledge about one entity provides constraints on the parameters of the others. Consequently, an image a parallelepiped with known euclidean structure allows to compute the intrinsic camera parameters, and reciprocally, a calibrated image of a parallelepiped allows to recover its euclidean shape (up to size). On the conceptual level, this duality can be seen as an alternative way to understand camera calibration: usually, calibration is considered to be equivalent to localizing the absolute conic or quadric in an image, whereas here we show that other primitives, such as canonic parallelepipeds, can be used as well.

The camera and parallelepiped parameters are recovered in two steps. First, their intrinsic and orientation parameters are computed. The original approach for this step was introduced in [Wilczkowiak et al., 2002] and consists on parameterization of the intrinsic matrices of all the objects (cameras and parallelepipeds) in terms of the intrinsic matrix of a reference object. The algorithm proposed later [Wilczkowiak et al., 2003c] exploits the fact that the parallelepipeds projection matrices can be factorized into two parts, representing respectively camera and parallelepiped parameters and allows to treat the available data simultaneously without privileging any primitive.

To complete the calibration, a least squares optimization is used to simultaneously recover the scale and position parameters in the common euclidean frame. The use of the well-constrained calibration primitives allows to obtain good calibration results even from a very small number of images.

Our calibration approach is conceptually close to self-calibration methods, especially those that upgrade affine to euclidean structure [Hartley, 1993; Pollefeys and Van Gool, 1997] or those that consider special camera motions [Hartley, 1997b; de Agapito et al., 1999; Armstrong et al., 1994]. The way metric information on a parallelepiped is used is also similar to vanishing point based methods [Caprile and Torre, 1990; Cipolla and Boyer, 1998; Chen et al., 1999; Kosecka and Zhang, 2002]. Some properties of our algorithm are also common with plane-based approaches [Sturm and Maybank, 1999a; Zhang, 1999; Triggs, 1998; Malis and Cipolla, 2002; Triggs, 2000; Sturm, 2000]. While more flexible than standard calibration techniques, plane-based approaches still require either euclidean information or, for self-calibration, many images in general position [Triggs, 1997]. In this sense, our approach is a generalization of plane-based methods with metric information to three-dimensional parallelepipedic patterns. This allows to handle missing data and unknown scale factors and simplifies the formulation of calibration constraints. Finally, our approach can be compared to methods using complex primitives for the scene representation. However, unlike most methods of this type, we use the parallelepiped parameters directly to solve the calibration problem without requiring non-linear optimization methods.

Depending on the prior information about the scene, singularities might occur. They are collected into a detailed dictionary, accompanied by a sketch of methodology used for its construction.

While the main contributions of this work concern the estimation of camera and parallelepiped parameters, we show how the results can be combined with a multi-linear reconstruction approach, using other primitives than parallelepipeds, described in chapter 6. The complete system allows both calibration and 3D model acquisition from a small number of arbitrary images with a reasonable amount of user interaction.

Outline

In chapter 3, the major properties of the parallelepiped-camera projections are described and the concept of camera-parallelepiped duality is introduced. Then in chapter 4, our algorithms for the estimation of camera and parallelepiped intrinsic and rotation parameters are described (sections 4.2 and 4.3). These algorithms are completed by a study of minimal and singular configurations in sections 4.4 and 4.5. Chapter 5 describes our approach for pose and scale estimation. Chapter 6 contains an overview of the full algorithm used to reconstruct scenes containing primitives other than parallelepipeds. Chapter 7 presents experimental results on synthetic and real data of calibration and reconstruction algorithms.

Chapter 3

Parallelepipeds and Their Projections

In this chapter, we present our parameterization of parallelepipeds and study the properties of their perspective images. We show that, in analogy to points, projection of a parallelepiped can be represented by a projection matrix. As shown in figure 3.1, it can be computed assuming that the vertices of a parallelepiped projected in the image belong to a canonical cube, and thus will be called in the following the *canonical parallelepiped projection matrix.* This matrix naturally encodes the affine structure of the scene. Thus, once computed, it directly provides an affine reconstruction. In a similar way like standard point-image projection matrices, it can be split into two parts, representing the homography between the plane at infinity and the image plane as well as the relative pose of objects. Using the canonical parallelepiped projection matrix it is possible to introduce a concept of duality between camera and parallelepiped intrinsic parameters. Apart from theoretical, this duality has an important practical value, allowing to use the prior information on all the cameras and parallelepipeds in a common frame.

3.1 Representation of Parallelepipeds

A parallelepiped encodes naturally the affine properties of the scene and facilitates modeling the remaining metric part. The representation of a parallelepiped given below is based on the formalism proposed in [Wilczkowiak et al., 2001, 2002].

A parallelepiped is defined by twelve parameters: six extrinsic parameters describing its orientation and position, and six intrinsic parameters describing its metric shape: three dimension parameters (edge lengths l_1, l_2 and l_3) and three angles between edges $(\theta_{12}, \theta_{23}, \theta_{13})$. These intrinsic parameters are illustrated in figure 1.11. The parallelepiped may be represented compactly in matrix form by a 4×4 matrix N:

$$\mathsf{N} = \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \tilde{\mathsf{L}}$$

where S is a rotation matrix and $\mathbf{v}$ a vector, representing the parallelepiped's pose (extrinsic parameters). The 4×4 matrix $\tilde{\mathsf{L}}$ represents the parallelepiped's shape:

$$\tilde{\mathsf{L}} = \begin{pmatrix} l_1 & l_2 c_{12} & l_3 c_{13} & 0 \\ 0 & l_2 s_{12} & l_3 \frac{c_{23} - c_{13} c_{12}}{s_{12}} & 0 \\ 0 & 0 & l_3 \sqrt{\frac{s_{12}^2 - c_{13}^2 s_{12}^2 - (c_{23} - c_{13} c_{12})^2}{s_{12}^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

with: $c_{ij} = \cos\theta_{ij}$, $s_{ij} = \sin\theta_{ij}$, $\theta_{ij} \in \ ]0 \ \pi[$, $l_i > 0$.

The matrix $\tilde{\mathsf{L}}$ represents the affine transformation between a canonical cube and a parallelepiped with the given shape. Formally, a vertex $(\pm 1, \pm 1, \pm 1, 1)^\mathsf{T}$ of the canonical cube is mapped, by $\tilde{\mathsf{L}}$, to a vertex of our parallelepiped's intrinsic shape. Then, the pose part of N maps the vertices into the world coordinate system.

Other parameterizations for $\tilde{\mathsf{L}}$ may be chosen, but the above one is attractive due to its upper triangular form. This underlines the fact that $\tilde{\mathsf{L}}$ plays the same role for the parallelepiped as the calibration matrix K for a camera (see section 1.3).

The analogous entity to a camera's IAC ω, is the matrix μ, defined by:

$$\mu \sim \mathsf{L}^\mathsf{T}\mathsf{L} \sim \begin{pmatrix} l_1^2 & l_1 l_2 \cos\theta_{12} & l_1 l_3 \cos\theta_{13} \\ l_1 l_2 \cos\theta_{12} & l_2^2 & l_2 l_3 \cos\theta_{23} \\ l_1 l_3 \cos\theta_{13} & l_2 l_3 \cos\theta_{23} & l_3^2 \end{pmatrix}, \tag{3.1}$$

where L is the upper left 3×3 matrix of $\tilde{\mathsf{L}}$.

Hence, there is a seemingly perfect symmetry between intrinsic parameters of cameras and parallelepipeds. The only difference is that in some cases, the *size* of a parallelepiped matters, as will be explained below. As for cameras, the fact that $K_{33} = 1$ allows us to fix the scale factor in the relation $\omega \sim \mathsf{K}^{-\mathsf{T}}\mathsf{K}^{-1}$, and thus to extract K uniquely from the IAC ω, e.g. using Cholesky decomposition. As for parallelepipeds, however, we have no such constraint on its "calibration matrix" L, so the relation $\mu \sim \mathsf{L}^\mathsf{T}\mathsf{L}$ gives us a parallelepiped's Euclidean shape, but not its (absolute) size. This does not matter in general, since we are usually only interested in reconstructing a scene up to some scale. However, when reconstructing several parallelepipeds, one needs to recover at least their *relative* sizes.

There are many possibilities of defining the size of parallelepipeds. We choose the following definition, due to its appropriateness in the equations underlying our calibration and reconstruction algorithms below: the **size** of a parallelepiped is defined as

$$s = (\det \mathsf{L})^{1/3} \ .$$

This definition is actually directly linked to the parallelepiped's volume: $s^3 = \det \mathsf{L} = \mathrm{Vol}/8$ (the factor 8 arises since our canonic cube has an edge length of 2).

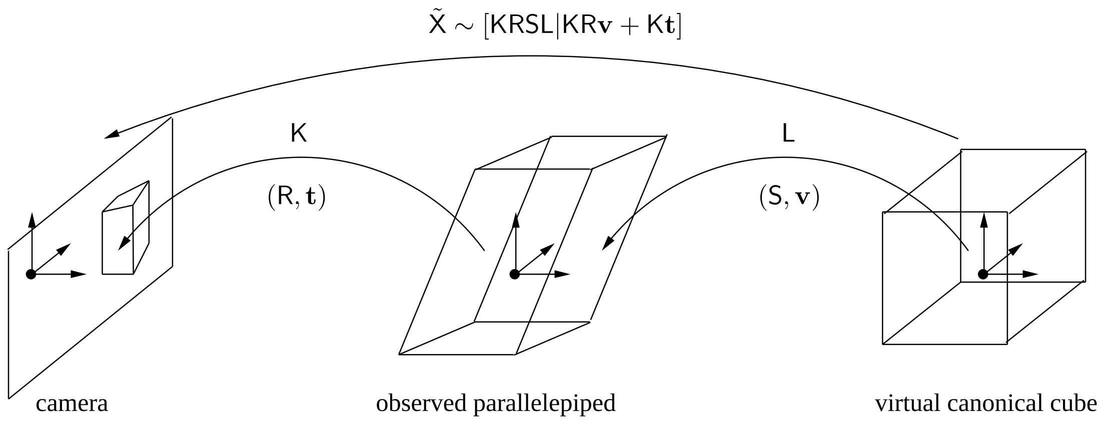

$\tilde{\mathsf{X}} \sim [\mathsf{KRSL}|\mathsf{KR}\mathbf{v}+\mathsf{K}\mathbf{t}]$
K
$(\mathsf{R},\mathbf{t})$
L
$(\mathsf{S},\mathbf{v})$
camera
observed parallelepiped
virtual canonical cube

Figure 3.1: The projection of the canonic parallelepiped (cube) into the image. Matrices K, L correspond to intrinsic parameters of camera and parallelepiped and $(\mathsf{R},\mathbf{t})$, $(\mathsf{S},\mathbf{v})$ correspond to extrinsic parameters of camera and parallelepiped, respectively.

3.2 One Parallelepiped in a Single View

In this section, we introduce the concept of duality between the intrinsic characteristics of a camera and those of a parallelepiped.

Consider the projection of the parallelepiped's vertices into the camera. Let $\mathbf{C}_{i,i\in[1\ldots8]}$ be the homogeneous coordinates of the canonic cube's vertices of the form $(\pm1,\pm1,\pm1,1)^{\mathsf{T}}$. Let matrices K and $\tilde{\mathsf{L}}$ (with affine part L) correspond to intrinsic, and matrices $(\mathsf{R},\mathbf{t})$, $(\mathsf{S},\mathbf{v})$ correspond to extrinsic camera and parallelepiped parameters, as described in section 3.1. The corresponding vertex of the parallelepiped is given as:

$$\mathbf{P}_i = \mathsf{N}\mathbf{C}_i = \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix} \tilde{\mathsf{L}}\mathbf{C}_i$$

and its image point is:

$$\mathbf{p}_i \sim \mathsf{M}\mathbf{P}_i = \mathsf{K}\begin{pmatrix} \mathsf{R} & \mathbf{t} \end{pmatrix} \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix} \tilde{\mathsf{L}}\mathbf{C}_i \ . \tag{3.2}$$

In the above equation, we define the *canonical parallelepiped projection matrix*:

$$\tilde{\mathsf{X}} \sim \mathsf{K}\begin{pmatrix} \mathsf{R} & \mathbf{t} \end{pmatrix} \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix} \tilde{\mathsf{L}} \ . \tag{3.3}$$

This matrix represents a perspective projection that maps the vertices of the canonic cube onto the image points of the parallelepiped's vertices. This is illustrated in figure 3.1. Given image points for sufficiently many vertices[1], the canonic projection matrix can be computed, even in the absence of prior knowledge on intrinsic or extrinsic parameters. Our calibration and pose estimation algorithms are based on the link between the canonic projection matrix (which we suppose given from now on) and the camera's and parallelepiped's intrinsic and extrinsic parameters.

Let us consider this in more detail. First, we may identify the *relative pose* between camera and parallelepiped in (3.3), represented by the following 3×4 matrix:

$$\begin{pmatrix} \mathsf{R} & \mathbf{t} \end{pmatrix} \begin{pmatrix} \mathsf{S} & \mathbf{v} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{pmatrix} = \begin{pmatrix} \mathsf{RS} & \mathsf{R}\mathbf{v}+\mathbf{t} \end{pmatrix}$$

[1]Five image points and one image direction are in general sufficient to solve for the 11 degrees of freedom of the parallelepiped's image projection. Additional points make the computation more stable.

Second, let us consider the leading 3×3 sub-matrix X of the canonic projection matrix $\tilde{\mathsf{X}}$, which is given by:

$$\mathsf{X} \sim \mathsf{K}\,(\mathsf{RS})\,\mathsf{L}. \tag{3.4}$$

This matrix will be called in the following the *reduced canonical projection matrix*.

Due to the orthogonality of the rotation matrices R and S, it is simple to derive the following relation between the camera's IAC ω and the corresponding entity μ of the parallelepiped:

$$\mathsf{X}^\mathsf{T}\omega\mathsf{X} \sim \mu, \text{ or, dually: } \mathsf{Y}^\mathsf{T}\mu\mathsf{Y} \sim \omega, \text{ for } \mathsf{Y} = \mathsf{X}^{-1}. \tag{3.5}$$

This equation establishes an interesting duality between the intrinsic parameters of a camera and those of a parallelepiped. It shows (unsurprisingly) that knowing the parallelepiped's shape μ allows to calibrate the camera from a single image. Conversely, knowing the camera's intrinsic parameters allows to directly compute the parallelepiped's metric shape, also from a single image.

In the next chapters, we generalize the use of this duality for calibration and pose estimation to the case of multiple parallelepipeds seen in multiple cameras and to the use of *partial* knowledge about the camera's or parallelepiped's intrinsic parameters. Before doing so, let us describe a few interesting links between our and other (self-) calibration scenarios.

Classical self-calibration proceeds usually in two main steps: first, a projective 3D reconstruction of the scene is obtained from correspondences across two or more images. Then, the projective reconstruction is transformed to a metric one using the available prior knowledge on intrinsic parameters. This upgrade is sometimes interlaced by an intermediate upgrade to an affine reconstruction.

In our scenario, we have a 3D reconstruction of the scene already from a *single* rather than multiple images, which is furthermore of *affine* rather than projective nature: we know that the observed parallelepiped's shape is that of a cube, up to some affine transformation. Analogously, our canonic projection matrix is equal to the true one up to an affine transformation. Hence, self-calibration in our scenario does not need to recover the plane at infinity, which is known to be the hardest ("most non-linear") part of classical self-calibration. Indeed, our calibration method is somewhat similar to the affine-to-Euclidean upgrade of stratified self-calibration approaches, e.g. [Hartley, 1993; Pollefeys and Van Gool, 1997].

Similarities also exist with (self-) calibration approaches based on special camera motions: calibrating a rotating camera [Hartley, 1997b; de Agapito et al., 1999] is more or less equivalent to self-calibrating a camera in general motion once affine structure is known. Other approaches recover the affine structure by first performing pure translations and then general motions or to approaches that consider special camera motions [Armstrong et al., 1994; Pollefeys et al., 1996].

Our scenario is similar to these. In the following chapters we show how it allows to efficiently combine the usual self-calibration constraints with constraints on scene structure. This enables to perform calibration (and 3D reconstruction) from very few images; one image may actually be sufficient.

3.3 n Parallelepipeds in m Views

The main motivation for the work described in this part of the thesis is to generalize the use of the duality introduced in the previous section: we consider the general case where multiple parallelepipeds are seen by multiple cameras (not all parallelepipeds need to be seen by all cameras). Furthermore, we do not in general suppose that some cameras or parallelepipeds are fully calibrated. We rather want to make efficient and complete use of any kind of partial calibration information. As for the cameras, this amounts to partial knowledge on their intrinsic parameters that is routinely used for self-calibration. As for parallelepipeds, we rather consider them as "vehicles" to jointly express simple yet useful geometric scene constraints. Defining orthogonality or parallelism constraints between for example "only" pairs of lines, amounts to providing information about the structure of individual *planes* in the scene. Parallelepipeds however allow to directly express richer couplings of constraints on *3D* scene structure.

Let us now consider the general case where n parallelepipeds are seen by m cameras. Let $\tilde{\mathsf{X}}_{ik}$ be the canonic projection matrix associated with the projection of the kth parallelepiped in the ith camera:

$$\tilde{\mathsf{X}}_{ik} \sim \mathsf{K}_i \begin{pmatrix} \mathsf{R}_i & \mathbf{t}_i \end{pmatrix} \begin{pmatrix} \mathsf{S}_k & \mathbf{v}_k \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \tilde{\mathsf{L}}_k$$

Let us explicitly introduce scale factors λ_{ik} such that the equality up to scale in the above equation can be turned into a component-wise equality:

$$\lambda_{ik}\tilde{\mathsf{X}}_{ik} = \mathsf{K}_i \begin{pmatrix} \mathsf{R}_i & \mathbf{t}_i \end{pmatrix} \begin{pmatrix} \mathsf{S}_k & \mathbf{v}_k \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \tilde{\mathsf{L}}_k \tag{3.6}$$

Just as a sidenote, observe that, for two views i and j, and a parallelepiped k, the infinite homography between the two views is given by the product $\mathsf{X}_{ik}\mathsf{X}_{jk}^{-1}$.

We may group together these equations for all m cameras and n parallelepipeds, into the following single matrix equation:

$$\underbrace{\begin{bmatrix} \lambda_{11}\tilde{\mathsf{X}}_{11} & \cdots & \lambda_{1n}\tilde{\mathsf{X}}_{1n} \\ \vdots & \ddots & \vdots \\ \lambda_{m1}\tilde{\mathsf{X}}_{m1} & \cdots & \lambda_{mn}\tilde{\mathsf{X}}_{mn} \end{bmatrix}}_{\mathcal{X}_{3m\times 4n}} = \underbrace{\begin{bmatrix} \mathsf{K}_1 \begin{pmatrix} \mathsf{R}_1 & \mathbf{t}_1 \end{pmatrix} \\ \vdots \\ \mathsf{K}_m \begin{pmatrix} \mathsf{R}_m & \mathbf{t}_m \end{pmatrix} \end{bmatrix}}_{\tilde{\mathcal{M}}_{3m\times 4}} \underbrace{\begin{bmatrix} \begin{pmatrix} \mathsf{S}_1 & \mathbf{v}_1 \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \tilde{\mathsf{L}}_1 & \cdots & \begin{pmatrix} \mathsf{S}_n & \mathbf{v}_n \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \tilde{\mathsf{L}}_n \end{bmatrix}}_{\tilde{\mathcal{S}}_{4\times 4n}} \tag{3.7}$$

The matrix $\tilde{\mathcal{X}}$ contains all information that can be recovered from the parallelepipeds' image points alone (in section 4.3, we discuss the issue of computing the scale factors λ_{ik}). In analogy with [Tomasi and Kanade, 1992], we call it the *measurement matrix*.

In analogy to the reduced canonical projection matrix we will consider also a *reduced measurement matrix* $\mathcal{X}$ consisting of the matrices X_{ik}. The reduced measurement matrix encodes the intrinsic parameters and orientation of all the cameras and parallelepipeds in the system. It will be used in chapter 4 for the estimation of those parameters. The remaining part, consisting of the fourth columns $\mathbf{x}_{ik}$ of matrices $\tilde{\mathsf{X}}_{ik}$ encodes the relative positions of cameras and parallelepipeds, as well as the parallelepipeds' sizes and will be discussed in chapter 5.

Chapter 4

Intrinsic and Orientation Parameters

In this chapter, we explain in detail, how the parallelepiped projection matrices can be used to incorporate scene constraints into the calibration process. Using parallelepipeds as natural calibration objects offers several advantages over standard self-calibration approaches. Firstly, fewer correspondences are needed; five and a half points extracted *per image* are sufficient, and even fewer inter-image correspondences are needed. For instance, the joint calibration of two cameras that view a parallelepiped from opposite viewpoints, is possible. Secondly, as mentioned in section 3, the reduced canonical parallelepiped projection matrices encode the affine properties of the scene, i.e., the infinite homography. In consequence, the calibration problem is reduced to a self-calibration problem where the plane at infinity is already localized [Hartley, 1993; Pollefeys and Van Gool, 1997] or where the cameras are stationary [Armstrong et al., 1994; Hartley, 1997b; de Agapito et al., 1999]. This observation is the foundation of the two calibration algorithms proposed in this chapter. The first algorithm, described in section 4.2, was introduced originally in [Wilczkowiak et al., 2002]and is conceptually close to the algorithms proposed for rotating cameras [Hartley, 1997b; de Agapito et al., 1999]. Using the inter-image homographies formed by reduced canonical parallelepiped projection matrices, the matrices representing intrinsic parameters of cameras and parallelepiped are parameterized in terms of one, reference object. Thus, the calibration of all the objects is reduced to the computation of five independent parameters of a 3×3 symmetric matrix defined up to a scalar factor. In the following step, the metric information provided by the cameras and parallelepipeds is used to form equations on the reference objects. When the intrinsic calibration is computed, the rotation parameters can be estimated in a factorization framework, extending plane-based approach described in [Sturm, 2000]. The intrinsic calibration algorithm is very fast, as the solution is provided by the resolution of a small linear equation system. However, due to the parameterization step, which requires distinguishing one object in the scene and the fact that rotation parameters are computed separately, this algorithms has worst properties that algorithm proposed later and described in section 4.3. Here, we a give factorization-based algorithm for simultaneous recovery of parallelepiped and camera intrinsic and orientation parameters. Such an estimation is only up to an affine transformation. However, as in the first approach, the metric information can be used to compute the transformation updating the affine reconstruction to the metric one. This transformation is represented by a 3×3 triangular matrix defined up to a scalar factor, thus reducing the calibration problem to the estimation of five independent parameters, as in the first approach. Interestingly, the use of the three-dimensional structure allows to deal very easily with the classical problems of the factorization methods: the unknown scale factors and the missing data.

In both the proposed algorithms, the way the metric information of a parallelepiped is used to stratify the reconstruction from affine to euclidean is similar to the vanishing point based methods [Caprile and Torre, 1990; Cipolla and Boyer, 1998; Chen et al., 1999; Kosecka and Zhang, 2002]. However, the constrained 3D structure of a parallelepiped improves the performance of the algorithm compared to the process of vanishing points computation and when registering images. Some properties of our algorithm are also common with plane-based approaches [Sturm and Maybank, 1999a; Zhang, 1999; Triggs, 1998; Malis and Cipolla, 2002; Rother et al., 2002; Sturm, 2000]. While more flexible than techniques based on calibration patterns [Tsai, 1986], plane-based approaches still require either euclidean information or, for self-calibration, many images in general position [Triggs, 1997], or at least one plane visible in all images [Rother et al., 2002]. In this sense, our approach is a generalization of plane-based methods with metric information to three-dimensional parallelepipedic patterns.

This chapter is organized as follows. In section 4.1 we resume the constraints provided by information on the metric structure of parallelepipeds and known intrinsic camera parameters. Next, in sections 4.2 and 4.3, we present the two calibration algorithms. Then we give some examples of minimal configurations for both the algorithms in section 4.4. We complete the chapter by a study of singular configurations in section 4.5.

4.1 Using Prior Knowledge

In this section we summarize the prior information on camera and parallelepiped intrinsic parameters which can be used for calibration. The parallelepiped-based calibration methods described in this part use the same constraints, and differ only in the problem parameterization.

4.1.1 Using Prior Information about Camera Intrinsics

Known values of camera intrinsics. Knowing respectively the aspect ratio and principal point coordinates of a camera i gives the following linear constraints on its IAC ω_i (based on equation (1.7)):

$$\tau_i^2 \omega_{i,11} - \omega_{i,22} = 0 \tag{4.1}$$
$$u_{i,0}\omega_{i,11} + \omega_{i,13} = 0 \tag{4.2}$$
$$v_{i,0}\omega_{i,22} + \omega_{i,23} = 0 \tag{4.3}$$

A known value of the focal length α_v can only be used to formulate linear equations if the other intrinsics are also known [Sturm and Maybank, 1999a]. In such a fully calibrated case, other algorithms might be better suited, so we neglect that case in the following, i.e we assume unknown focal lengths.

Constant camera intrinsics. In the case that two cameras i and j are known to have the same, yet unknown value for one intrinsic parameter, we in general obtain *quadratic* equations on ω_i and ω_j. For example, the assumption of equal aspect ratios leads to the equation:

$$\omega_{i,11}\omega_{j,22} = \omega_{j,11}\omega_{i,22}$$

The situation is different if *all* intrinsic parameters of two (or more) views are known to be identical. In that case, we can obtain linear equations instead of quadratic ones, as shown in [Hartley, 1997b]. It will be detailed in sections 4.2 and 4.3.

In practice, we only use available linear equations. In some minimal cases, quadratic equations as above might be useful to find a unique solution or a finite set of solutions, if the available linear constraints are insufficient.

4.1.2 Using Prior Information about Parallelepiped Intrinsics

Knowledge on parallelepiped intrinsics can be used in an analogous way as knowledge about camera parameters.

Known values of parallelepiped intrinsics. Suppose we know the length ratio of two edges of a parallelepiped k: $r_{k,uv} = \frac{l_{k,u}}{l_{k,v}}$. Referring to (3.1), we get the following *linear* equation:

$$r_{k,uv}^2 \mu_{k,vv} - \mu_{k,uu} = 0. \tag{4.4}$$

Similarly, the assumption that $\theta_{k,uv}$ is a right angle, i.e. $\cos\theta_{k,uv} = 0$, leads also to a *linear equation*:

$$\mu_{k,uv} = 0. \tag{4.5}$$

A known angle of a parallelepiped $\theta_{k,uv} \neq 90°$ gives a *quadratic* constraint on the associated matrix μ_k:

$$\mu_{k,uv}^2 = \cos^2\theta_{k,uv}\mu_{k,uu}\mu_{k,vv}.$$

Constant parallelepiped intrinsics. As for cameras, *quadratic* equations may be derived from assumptions about two or more parallelepiped having the same, yet unknown value for some intrinsic parameter. Furthermore, information about two parallelepipeds k, l having the same shape, leads to a set of 4 *linear* equations on μ_k and μ_l. This holds even if the parallelepipeds are of a different size. Knowing in addition that they are of the same size, gives an additional *linear* equation, but constraining rather their positions and sizes than intrinsic parameters.

Currently, we only exploit constraints on individual parallelepipeds (right angles and length ratios), since they are easier to provide for the user.

4.2 Reference Primitive Method

The calibration algorithm described in this section is the first approach we proposed for the multi-view calibration problem in [Wilczkowiak et al., 2002]. It is based on the parameterization of the intrinsic matrices of all the objects (cameras and parallelepipeds) in terms of the intrinsic matrix of a reference primitive. This is done using parallelepiped projection matrices. Due to the necessity of choice of reference object the properties of this algorithm are less interesting than properties of the factorization algorithm given in the following section, but we sketch this method to keep this thesis self-contained.

4.2.1 Calibration of the Reference Primitive

We assume that n parallelepipeds are seen by m cameras. The geometric information that can be computed from the projection of a parallelepiped k in image i is enclosed in the reduced measurement matrix $\mathsf{X}_{ik} = \mathsf{Y}_{ik}^{-1}$. From such matrices and prior information, we can derive constraints on the calibration by using the duality equations. In this section, we will show how the constraints arising from the prior knowledge on the elements of ω_i or μ_k given is section 4.1 can be used in practice.

The duality equations between μ_0 and image i, respectively parallelepiped k, are:

$$\begin{cases} \omega_i \sim \mathsf{Y}_{i0}^\top \mu_0 \mathsf{Y}_{i0}, \\ \mu_k \sim \mathsf{X}_{ik}^\top \mathsf{Y}_{i0}^\top \mu_0 \mathsf{Y}_{i0} \mathsf{X}_{ik} \quad i \in [0 \ldots m-1]. \end{cases} \tag{4.6}$$

which may be rewritten in several different forms, by nested applications of the duality equations, e.g.:

$$\begin{cases} \omega_i \sim \mathsf{Y}_{il}^\top \mathsf{X}_{jl}^\top \mathsf{Y}_{j0}^\top \mu_0 \underbrace{\mathsf{Y}_{j0} \mathsf{X}_{jl} \mathsf{Y}_{il}}_{\mathsf{G}_i} \sim \mathsf{G}_i^\top \mu_0 \; \mathsf{G}_i, \\ \mu_k \sim \mathsf{X}_{jk}^\top \mathsf{Y}_{jl}^\top \mathsf{X}_{0l}^\top \mathsf{Y}_{00}^\top \mu_0 \underbrace{\mathsf{Y}_{00} \mathsf{X}_{0l} \mathsf{Y}_{jl} \mathsf{X}_{jk}}_{\mathsf{G}_k} \sim \mathsf{G}_k^\top \mu_0 \; \mathsf{G}_k. \end{cases} \tag{4.7}$$

where $j \in [0 \ldots m-1]$, $l \in [0 \ldots n-1]$. Thus for every parallelepiped and camera in the scene there exist a transformation G parameterizing its internal parameters in terms of parameters of an arbitrary μ_0. Such forms of the duality equations do in principle not provide additional independent equations on the elements of μ_0. However, they are useful for example in the case where the parallelepiped associated to μ_0 is occluded in view i (e.g. due to occlusion), thus X_{i0} is not available. Using extensions of the duality equations such as (4.7), it is possible to express the information on any of the parallelepipeds or cameras in terms of μ_0.

We therefore derive the statements given in the following two paragraphs. When we speak of independent equations, we mean that they are independent in general, i.e. non-singular, situations.

Known values of camera and parallelepiped intrinsics. Table 4.1 summarizes the prior knowledge on intrinsic parameters which results in linear equations on μ_0.

Known parameters	Constraints
parallelepiped	
right angle $\theta_{k,uv}$	$\left(\mathsf{G}_k^\top \mu_0 \mathsf{G}_k\right)_{uv} = 0$
length ratio $r_{k,uv} = \frac{l_{k,u}}{l_{k,v}}$	$r_{k,uv}^2 \left(\mathsf{G}_k^T \mu_0 \mathsf{G}_k\right)_{vv} - \left(\mathsf{G}_k^T \mu_0 \mathsf{G}_k\right)_{uu} = 0$
camera	
skew $s = 0$	$\left(\mathsf{G}_i^\top \mu_0 \mathsf{G}_i\right)_{12} = 0$
aspect ratio τ_i^2	$\tau_i^2 \left(\mathsf{G}_i^T \mu_0 \mathsf{G}_i\right)_{11} - \left(\mathsf{G}_i^T \mu_0 \mathsf{G}_i\right)_{22} = 0$
principal point $u_{i,0}$	$u_{i,0} \left(\mathsf{G}_i^\top \mu_0 \mathsf{G}_i\right)_{11} + \left(\mathsf{G}_i^\top \mu_0 \mathsf{G}_i\right)_{13} = 0$
$v_{i,0}$	$v_{i,0} \left(\mathsf{G}_i^\top \mu_0 \mathsf{G}_i\right)_{22} + \left(\mathsf{G}_i^\top \mu_0 \mathsf{G}_i\right)_{23} = 0$

Table 4.1: Exemplary equations on matrix μ_0 engendered by prior knowledge on camera and parallelepiped parameters.

As mentioned before, any of these constraints can be enforced using one or several redundant equations of the types (4.6) and (4.7) for example. Note that due to the above facts, an acquisition system with five different cameras viewing an arbitrary parallelepiped can be fully calibrated under the assumption of the skew parameters being zero. Equivalently, a system with one camera viewing five parallelepipeds with one right angle, or two parallelepipeds with three right angles can be fully calibrated. An extended list of the minimal cases for our calibration algorithm is given in section 4.4.

Constant values of camera and parallelepiped intrinsics. As shown in section 4.1 in a case when two cameras i and j are known to have the same, yet unknown value for one intrinsic parameter, we in general obtain *quadratic* equations on $\omega_i$and ω_j. For example, the assumption of equal aspect ratios leads to the equation:

$$\left(\mathsf{G}_i^\top \mu_0 G_i\right)_{11} \left(\mathsf{G}_j^\top \mu_0 G_j\right)_{22} = \left(\mathsf{G}_j^\top \mu_0 G_j\right)_{11} \left(\mathsf{G}_i^\top \mu_0 G_i\right)_{22}$$

In analogy to cameras cameras, *quadratic* equations may be derived from assumptions about two or more parallelepiped having the same, yet unknown value for some intrinsic parameter.

The situation is different if *all* intrinsic parameters of two (or more) views are known to be identical. In that case, we can obtain linear equations instead of quadratic ones, as shown in [Hartley, 1997b]. When all the matrices G_i are scaled such as to have unit determinants, we can write the following component-wise matrix equality between any pair (i, j) of views:

$$\mathsf{G}_i^\top \mu_0 \mathsf{G}_i - \mathsf{G}_j^\top \mu_0 \mathsf{G}_j = 0_{3\times 3}$$

This represents 6 *linear* equations on μ_0 for each pair of views, among which 4 are independent.

In a similar way, two parallelepipeds having the same shape, leads to a set of 4 *linear* independent equations on μ_0.

4.2.2 Rotational Part of the Pose

When the intrinsic parameters of the cameras ($\mathsf{K}_{i,i\in[0..m-1]}$) and the parallelepipeds ($\Lambda_{k,k\in[0..n-1]}$) are computed, it turns out to be easy to compute the orientation of all the objects in a common frame. Indeed, from every matrix X_{ik} (see section 3.3) we can compute the matrix $\tilde{\mathsf{X}}'_{ik}$, which represents a relative rotation between a camera i and a parallelepiped k:

$$\mathsf{X}'_{ik} = \mathsf{K}_i^{-1} \mathsf{X}_{ik} {\Lambda_k}^{-1} \sim \mathsf{R}_i \mathsf{S}_k. \tag{4.8}$$

In practice, X'_{ik} will not be a perfect rotation matrix, but this can easily be corrected using SVD [Kanatani, 1996].

Let us first consider the case where all parallelepipeds are seen in all views. Then, all matrices X'_{ik} can be grouped and written as:

$$\underbrace{\begin{pmatrix} \mathsf{X}'_{0,0} & \mathsf{X}'_{0,1} & s & \mathsf{X}'_{0,n-1} \\ \mathsf{X}'_{1,0} & \mathsf{X}'_{1,1} & s & \mathsf{X}'_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{X}'_{m-1,0} & \mathsf{X}'_{m-1,1} & s & \mathsf{X}'_{m-1,n-1} \end{pmatrix}}_{\mathsf{X}'} = \underbrace{\begin{pmatrix} \mathsf{R}_0 \\ \mathsf{R}_1 \\ \vdots \\ \mathsf{R}_{m-1} \end{pmatrix}}_{\mathsf{R}} \underbrace{\begin{pmatrix} \mathsf{S}_0 & \mathsf{S}_1 & s & \mathsf{S}_{n-1} \end{pmatrix}}_{\mathsf{S}} \tag{4.9}$$

The matrices R_i and S_k can be extracted by factorizing X', due to the fact that its rank is 3. The factorization leads to solutions defined up to a global rotation. One might thus attach the reference frame to any camera or parallelepiped. All the issues concerning the unknown scale factors or missing data discussed in the following chapter 4.3 are valid also for the factorization problem presented in this section.

4.2.3 Complete Algorithm

In the current implementation, a scene is represented by a bi-partitioned graph, whose nodes are the cameras as well as parallelepipeds and the edges are the projections. We assume that the graph is connected and, consequently, for each object i the transformation G_i such that ${\mathsf{G}_i}^{\mathsf{T}}\mu_0\mathsf{G}_i$ can be computed. When this is not a case, all the connected parts of the graph have to be calibrated separately. Our calibration approach consists of two stages. First, all the available linear equations are used to determine μ_0 (the system is solved using SVD). If there is a unique solution, then we are done (from μ_0, all the camera and parallelepiped intrinsics can be computed using the G_{ik}). Note that in a general case(where no singularities occur) only five constraints of any of the type listed in table 4.1 are needed to recover the five unknown parameters of μ_0. If however the linear system is under-constrained, then the quadratic equations arising from constant but unknown parameters can be used to reduce the ambiguity in the solution. The decision if the system is under-constrained may be taken on the basis of a singular value analysis. This also gives the degree of the ambiguity (dimension of the solution space). In practice, this is usually two or lower. Hence, two quadratic equations are in general sufficient to obtain a finite number of solutions (if more than the minimum number of equations are available, then a best initial solution might be found using a RANSAC-type approach [Fischler and Bolles, 1981]). Once the matrices ω_i and μ_k are estimated, the matrices K_i and L_i can be computed via Cholesky decomposition.

Finally, we propose the following algorithm:

1. Construct the graph with cameras and parallelepipeds as nodes and projections as edges.
2. Estimate the canonical projection matrices $\tilde{\mathsf{X}}_{ik}$.
3. Choose a reference parallelepiped represented by μ_0 (choose the one projected to the bigest number of images).
4. Compute paths (shortest for example) connecting all the cameras i and parallelepipeds k to μ_0 and use them to compute transformations G_i, G_k.
5. Establish linear equation system on μ_0 based on prior knowledge of intrinsic parameters of cameras and parallelepipeds.
6. Solve the system to least squares.

7. If necessary, use the non-linear equations to resolve the remaining ambiguities on μ_0.

8. Compute the matrices ω_i, μ_k using μ_0 and transformations G_i, G_k.

9. Extract the K_i, L_k from the ω_i and the μ_k using e.g. QR-decomposition. Note that at this stage the L_k can only be recovered up to scale, i.e. the parallelepipeds' (relative) sizes remain undetermined.

10. Compute rotation matrices by factorization of a measurement matrix composed of matrices X'_{ik}.

4.3 Factorization Method

Equation (3.7) naturally leads to the idea of a factorization-based calibration algorithm, which will be developed in this section. It is based on the following observation. The measurement matrix $\mathcal{X}$ (see equation (3.7)) contains all information that can be recovered from the parallelepipeds' image points alone (below, we discuss the issue of computing the scale factors λ_{ik}). Since the measurement matrix is the product of a "motion matrix" $\tilde{\mathcal{M}}$ of 4 columns, with a "shape matrix" $\tilde{\mathcal{S}}$ of 4 rows, its rank can be 4 at most (in the absence of noise).

We might aim at extracting intrinsic and extrinsic parameters of cameras and parallelepipeds directly from a rank-4-factorization of $\tilde{\mathcal{X}}$. One step of many factorization methods for structure and motion recovery is to disambiguate the result of the factorization: in general, for a rank-r-factorization, motion and shape are recovered up to a transformation represented by an $r \times r$ matrix (in our case, this would be a 3D projective transformation). The ambiguity can be reduced using e.g. constraints on intrinsic camera parameters. In our case, we observe that the 4×4 sub-blocks of the shape matrix $\tilde{\mathcal{S}}$ are affine transformations (last row consists of three zeroes and a one). We would have to include this constraint into the disambiguation, but nevertheless, the result would not in general exactly satisfy the affine form of these sub-blocks. We thus cut the problem into two steps, which allows to easily guarantee that the sub-blocks of the shape matrix be affine transformations. In the first step described in this section, we consider the *reduced measurement matrix* $\mathcal{X}_{ik}$, defined in section 3.3. We extract *intrinsic parameters* and *orientation* of our cameras and parallelepipeds based on a rank-3-factorization and a disambiguation stage using calibration and scene constraints. The second step, consisting of the computation of position parameters and parallelepiped size is described in section 5.

As mentioned previously, we first restrict our attention to the leading 3×3 submatrices of the $\tilde{\mathsf{X}}_{ik}$, like we did in section 3.2 for the establishment of the duality between intrinsic parameters of cameras and parallelepipeds. We thus deal with the corresponding subpart of equation (3.7):

$$\underbrace{\begin{bmatrix} \lambda_{11}\mathsf{X}_{11} & \cdots & \lambda_{1n}\mathsf{X}_{1n} \\ \vdots & \ddots & \vdots \\ \lambda_{m1}\mathsf{X}_{m1} & \cdots & \lambda_{mn}\mathsf{X}_{mn} \end{bmatrix}}_{\mathcal{X}_{3m\times 3n}} = \underbrace{\begin{bmatrix} \mathsf{K}_1\mathsf{R}_1 \\ \vdots \\ \mathsf{K}_m\mathsf{R}_m \end{bmatrix}}_{\mathcal{M}_{3m\times 3}} \underbrace{\begin{bmatrix} \mathsf{S}_1\mathsf{L}_1 & \cdots & \mathsf{S}_n\mathsf{L}_n \end{bmatrix}}_{\mathcal{S}_{3\times 3n}} \tag{4.10}$$

In the following sections, we describe the different steps of our factorization-based method. We first deal with the important problem of missing data. Then we describe how the scale factors λ_{ik} needed to construct the measurement matrix $\tilde{\mathcal{X}}$, can be excluded from the intrinsic and rotation parameters estimation process. The factorization itself is described in section 4.3.3, followed by the most important aspect: how to disambiguate the factorization's result in order to extract intrinsic and orientation parameters.

4.3.1 Missing Data

As with all factorization problems, our method might suffer from the problem of missing data, i.e. missing X_{ik}. Indeed, in practice, the condition that all parallelepipeds are seen in all views can not always be satisfied. However, each missing matrix X_{ik} can be deduced from others if there is at least one camera j and one parallelepiped l such that the transformations X_{jl}, X_{jk} and X_{il} are known. The missing matrix can then be computed using:

$$\mathsf{X}_{ik} \sim \mathsf{X}_{il}\ (\mathsf{X}_{jl})^{-1}\ \mathsf{X}_{jk}. \tag{4.11}$$

Several equations of this type may be used simultaneously to increase the accuracy. In that case, care has to be taken since equation (4.11) is defined up to scale only. This problem can be circumvented very simply though, by normalizing all X_{ik} to unit determinant.

These observations motivate a simple recursive method[1] to compute missing matrices X_{ik}: at each iteration, we compute the one for which most equations of type (4.11) are available. Previously computed matrices X_{ik} can be involved at every successive iteration of this procedure.

4.3.2 Recovery of Scale Factors

The reduced measurement matrix $\mathcal{X}$ in (4.10) is, in the absence of noise, of rank 3, being the product of a matrix of 3 columns and a matrix of 3 rows. This however only holds if a correct set of scale factors λ_{ik} is used. For other problems, these are often non trivial to compute, see e.g. [Malis and Cipolla, 2002; Sturm and Triggs, 1996; Zelnik-Manor and Irani, 2002]. In our case however, this turns out to be rather simple.

Let us first write $\mathsf{A}_i = \mathsf{K}_i\mathsf{R}_i$ and $\mathsf{B}_k = \mathsf{S}_k\mathsf{L}_k$. What we know is that (in the absence of noise), there exist matrices $\mathsf{A}_i, i = 1..m$ and $\mathsf{B}_k, k = 1..n$ such that:

$$\forall i,k : \mathsf{X}_{ik} \sim \mathsf{A}_i\mathsf{B}_k$$

Since this equation is valid up to scale only, we also have:

$$\forall i,k : \mathsf{X}_{ik} \sim (a_i\mathsf{A}_i)\,(b_k\mathsf{B}_k)$$

for any non-zero scale factors $a_i, i = 1..m$ and $b_k, k = 1..n$. Consequently, this is also true for the scale factors with:

$$\begin{aligned}\det(a_i\mathsf{A}_i) &= 1\\ \det(b_k\mathsf{B}_k) &= 1\end{aligned}$$

Note that we do not need to know these scale factors; it is sufficient to know they exist.

Hence:

$$\forall i,k : \mathsf{X}_{ik} \sim a_ib_k\mathsf{A}_i\mathsf{B}_k,$$

with $\det(a_ib_k\mathsf{A}_i\mathsf{B}_k) = \det(a_i\mathsf{A}_i)\det(b_k\mathsf{B}_k) = 1$.

To achieve a component-wise equality $\lambda_{ik}\mathsf{X}_{ik} = (a_i\mathsf{A}_i)(b_k\mathsf{B}_k)$, we need to use scale factors[2] λ_{ik} such that $\det(\lambda_{ik}\mathsf{X}_{ik}) = 1$. Hence:

$$\lambda_{ik} = (\det\mathsf{X}_{ik})^{-1/3}$$

[1]Compare with the analogous method in [Sturm, 2000].

[2]It is well known that two non-singular 3×3 matrices that are equal up to scale and whose determinants are equal, are also equal component-wise.

In the following, we assume that the X_{ik} are already normalized to unit determinant, i.e. that $\lambda_{ik} = 1$. Equation (4.10) becomes:

$$\underbrace{\begin{bmatrix} \mathsf{X}_{11} & \cdots & \mathsf{X}_{1n} \\ \vdots & \ddots & \vdots \\ \mathsf{X}_{m1} & \cdots & \mathsf{X}_{mn} \end{bmatrix}}_{\mathcal{X}_{3m\times 3n}} = \underbrace{\begin{bmatrix} a_1 \mathsf{K}_1 \mathsf{R}_1 \\ \vdots \\ a_m \mathsf{K}_m \mathsf{R}_m \end{bmatrix}}_{\mathcal{M}_{3m\times 3}} \underbrace{\begin{bmatrix} b_1 \mathsf{S}_1 \mathsf{L}_1 & \cdots & b_n \mathsf{S}_n \mathsf{L}_n \end{bmatrix}}_{\mathcal{S}_{3\times 3n}} \tag{4.12}$$

The scale factors a_i and b_k do not matter for now; all that counts is that the measurement matrix $\mathcal{X}$ containing the normalized X_{ik}, is of rank 3 at most, and can thus be factorized as shown below.

4.3.3 Factorization

As usual, we use the SVD (Singular Value Decomposition) to obtain the low-rank factorization of the measurement matrix. Let the SVD of $\mathcal{X}$ be given as:

$$\mathcal{X}_{3m\times 3n} = \mathsf{U}_{3m\times 3n} \mathsf{\Sigma}_{3n\times 3n} \mathsf{V}^{\mathsf{T}}_{3n\times 3n}$$

The diagonal matrix Σ contains the singular values of $\mathcal{X}$. Let them be ordered:
$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{3n}$. In the absence of noise, $\mathcal{X}$ is of rank 3 at most and $\sigma_4 = \cdots = \sigma_{3n} = 0$. If noise is present, $\mathcal{X}$ is of full rank in general. Setting all singular values to zero, besides the three largest ones, leads to the best rank-3 approximation of $\mathcal{X}$ (in the sense of the Frobenius norm [Reid and Murray, 1996]).

In the following, we consider the decomposition of the rank-3 approximation of $\mathcal{X}$ (for ease of notation, we denote this also as $\mathcal{X}$):

$$\mathcal{X} = \mathsf{U}_{3m\times 3n} \quad \mathrm{diag}(\sigma_1, \sigma_2, \sigma_3, 0, \ldots, 0) \quad \mathsf{V}^{\mathsf{T}}_{3n\times 3n}$$

In the matrix product on the right, only columns of U and rows of V^{T} that correspond to non-zero singular values, contribute. Hence:

$$\mathcal{X} = \mathsf{U}'_{3m\times 3} \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix} \left(\mathsf{V}'^{\mathsf{T}}\right)_{3\times 3n}$$

where U' (resp. V') consists of the first three columns of U (resp. V). Let us define

$$\begin{aligned} \mathsf{U}'' &= \mathsf{U}' \begin{pmatrix} \sqrt{\sigma_1} & & \\ & \sqrt{\sigma_2} & \\ & & \sqrt{\sigma_3} \end{pmatrix} \\ \mathsf{V}'' &= \mathsf{V}' \begin{pmatrix} \sqrt{\sigma_1} & & \\ & \sqrt{\sigma_2} & \\ & & \sqrt{\sigma_3} \end{pmatrix} \end{aligned}$$

Thus:

$$\mathcal{X} = \mathsf{U}''\mathsf{V}''^{\mathsf{T}}$$

This represents a decomposition of the measurement matrix $\mathcal{X}$ into a product of a matrix of 3 columns with a matrix of 3 rows. Note however, that this decomposition is not unique. For any non-singular 3×3 matrix T, the following is also a valid decomposition:

$$\mathcal{X} = \left(\mathsf{U}''\mathsf{T}^{-1}\right) \left(\mathsf{T}\mathsf{V}''^{\mathsf{T}}\right)$$

Making the link with equation (4.12), we obtain:

$$\begin{bmatrix} a_1\mathsf{K}_1\mathsf{R}_1 \\ \vdots \\ a_m\mathsf{K}_m\mathsf{R}_m \end{bmatrix} \begin{bmatrix} b_1\mathsf{S}_1\mathsf{L}_1 & \cdots & b_n\mathsf{S}_n\mathsf{L}_n \end{bmatrix} = \left(\mathsf{U}''\mathsf{T}^{-1}\right)\left(\mathsf{T}\mathsf{V}''^{\mathsf{T}}\right) \tag{4.13}$$

Let us write U'' and V'' as follows as a composition of 3×3 submatrices:

$$\mathsf{U}'' = \begin{bmatrix} \mathsf{U}_1 \\ \vdots \\ \mathsf{U}_m \end{bmatrix} \qquad \mathsf{V}'' = \begin{bmatrix} \mathsf{V}_1 \\ \vdots \\ \mathsf{V}_n \end{bmatrix}$$

Equation (4.13) thus becomes:

$$\begin{bmatrix} a_1\mathsf{K}_1\mathsf{R}_1 \\ \vdots \\ a_m\mathsf{K}_m\mathsf{R}_m \end{bmatrix} \begin{bmatrix} b_1\mathsf{S}_1\mathsf{L}_1 \cdots b_n\mathsf{S}_n\mathsf{L}_n \end{bmatrix} = \begin{bmatrix} \mathsf{U}_1\mathsf{T}^{-1} \\ \vdots \\ \mathsf{U}_m\mathsf{T}^{-1} \end{bmatrix} \begin{bmatrix} \mathsf{T}\mathsf{V}_1^{\mathsf{T}} \cdots \mathsf{T}\mathsf{V}_n^{\mathsf{T}} \end{bmatrix} \tag{4.14}$$

How to estimate T is explained in section 4.3.4. Once a correct estimate is given, we can directly extract the matrices $\mathsf{A}_i = a_i\mathsf{K}_i\mathsf{R}_i$ and $\mathsf{B}_k = b_k\mathsf{S}_k\mathsf{L}_k$, from which in turn the individual rotation matrices and calibration matrices can be recovered using a Cholesky or QR-decomposition. The Cholesky decomposition of $\mathsf{A}_i\mathsf{A}_i^{\mathsf{T}}$ for example, results in an upper triangular matrix $\mathsf{M}_i = a_i\mathsf{K}_i$. Based on the requirement $K_{i,33} = 1$, we can compute the unknown scale factor a_i as $a_i = M_{i,33}$. The calibration matrix is finally obtained as $\mathsf{K}_i = \frac{1}{a_i}\mathsf{M}_i$.

As for the parallelepipeds, we do not have any global constraint on the entries of their calibration matrices L_k. Hence, we can compute them only up to the unknown scale factors b_k. This means that we can compute the *shape* of each parallelepiped, but not (yet) their *size* (or, volume). In section 5, we explain how to compute their (relative) size.

We now briefly discuss the structure and geometric signification of the matrix T. Note that T actually represents the non-translational part of a 3D affine transformation (its upper left 3×3 submatrix). This is just another expression of the previously mentioned fact that due to the observation of parallelepipeds, we directly have an affine reconstruction (of scene and cameras).

The matrix T can only be computed up to an arbitrary rotation and scale. Indeed, consider expression $\mathsf{R}\mathsf{T}\mathbf{V}_k \sim \mathsf{S}_k\mathsf{L}_k$. For given T right side of the expression can be decomposed into rotation matrix S_k and upper triangular matrix L_k. However, decomposition of expression $\mathsf{T}'\mathbf{V}_k \sim \mathsf{R}\mathsf{T}\mathbf{V}_k \sim (\mathsf{R}\mathsf{S}_k)\mathsf{L}_k$ would result in the same triangular matrix L_k. This ambiguity is natural and expresses the fact that the global Euclidean reference frame for the reconstruction of our parallelepipeds and cameras can be chosen arbitrarily. Thus without loss of generality, we may assume that it is chosen in a way that T is upper triangular. This highlights the fact that our estimation problem has only 5 degrees of freedom (6 parameters for an upper triangular 3×3 matrix minus one for the arbitrary scale) which can also be explained in more geometric terms: as explained previously, our problem is somewhat equivalent to self-calibration with known affine structure. The 5 degrees of the problem may be interpreted as the coefficients of the absolute conic on the plane at infinity.

4.3.4 Disambiguating the Factorization

We now deal with the estimation of the unknown transformation T appearing in equation (4.14). As will be seen below, and as is often the case in self-calibration problems, it is simpler to not directly estimate T, but the symmetric and positive definite 3×3 matrix Z defined as:

$$\mathsf{Z} = \mathsf{T}^{\mathsf{T}}\ \mathsf{T} \tag{4.15}$$

We may observe that this matrix represents the absolute conic on the plane at infinity. Once Z is estimated, T may be extracted from it using Cholesky decomposition. As described above, T is defined up to a rotation and scale, so the upper triangular Cholesky factor of Z can directly be used as the estimate for T.

The matrix Z (and thus T), can be estimated in various ways, using any information about the cameras or the parallelepipeds, e.g. prior knowledge on relative positioning of some entities. Here, we concentrate on exploiting prior information on intrinsic parameters, of both, cameras and parallelepipeds. As explained in section 4.1, there are two types of information that we consider:

- knowledge of the actual value of some intrinsic parameter for some camera or parallelepiped.
- knowledge that two or more cameras (or parallelepipeds) have the same value for some intrinsic parameter. We also sometimes speak of "constant" intrinsic parameters.

4.3.4.1 Using Knowledge on Camera and Parallelepiped Intrinsics

Before considering how the knowledge on camera and parallelepiped intrinsics can be used to estimate the transformation Z, let us show how the matrices ω and μ representing these intrinsics and the transformation Z can be related to each other.

Let us first consider relation between Z and camera intrinsic parameters.

From equation (4.14), we have:

$$a_i \mathsf{K}_i \mathsf{R}_i = \mathsf{U}_i \mathsf{T}^{-1}$$

Due to the orthogonality of R_i, we get:

$$a_i^2 \underbrace{\mathsf{K}_i \mathsf{K}_i^{\mathsf{T}}}_{\omega_i^{-1}} = \mathsf{U}_i \underbrace{\mathsf{T}^{-1}\mathsf{T}^{-\mathsf{T}}}_{\mathsf{Z}^{-1}} \mathsf{U}_i^{\mathsf{T}}$$

Neglecting the unknown scale factor a_i and taking the inverse of both sides of the equation, we obtain (note that the U_i are not orthogonal in general):

$$\omega_i \sim \mathsf{U}_i^{-\mathsf{T}} \mathsf{Z} \mathsf{U}_i^{-1}. \tag{4.16}$$

In similar way, a relation between Z and parallelepiped intrinsic parameters can be established. From equation (4.14), we have:

$$b_k \mathsf{S}_k \mathsf{L}_k = \mathsf{T} \mathsf{V}_k^{\mathsf{T}}$$

Due to the orthogonality of S_k, we get:

$$b_k^2 \underbrace{\mathsf{L}_k^{\mathsf{T}} \mathsf{L}_k}_{\mu_k} = \mathsf{V}_k \underbrace{\mathsf{T}^{\mathsf{T}}\mathsf{T}}_{\mathsf{Z}} \mathsf{V}_k^{\mathsf{T}}$$

Neglecting the unknown scale factor b_k, we obtain:

$$\mu_k \sim \mathsf{V}_k \mathsf{Z} \mathsf{V}_k^{\mathsf{T}}. \tag{4.17}$$

We are now ready to formulate constraints on Z based on prior knowledge on the cameras' and parallelepipeds intrinsics in a way similar as in section 4.2.

Known values of camera and parallelepiped intrinsics. Table 4.2 summarizes the prior knowledge on intrinsic parameters which results in linear equations on Z.

Known parameters	Constraints
parallelepiped	
right angle $\theta_{k,uv}$	$\left(\mathsf{V}_k\mathsf{Z}\mathsf{V}_k^\mathsf{T}\right)_{uv} = 0$
length ratio $r_{k,uv} = \frac{l_{k,u}}{l_{k,v}}$	$r_{k,uv}^2\left(\mathsf{V}_k\mathsf{Z}\mathsf{V}_k^\mathsf{T}\right)_{vv} - \left(\mathsf{V}_k\mathsf{Z}\mathsf{V}_k^\mathsf{T}\right)_{uu} = 0$
camera	
skew $s = 0$	$\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{12} = 0$
aspect ratio τ_i^2	$\tau_i^2\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{11} - \left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{22} = 0$
$u_{i,0}$	$u_{i,0}\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{11} + \left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{13} = 0$
$v_{i,0}$	$v_{i,0}\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{22} + \left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{23} = 0$

Table 4.2: Examples of linear equations on the matrix Z engendered by prior knowledge on camera and parallelepiped parameters.

Constant values of camera and parallelepiped intrinsics. As shown in section 4.1 in the case when two cameras i and j are known to have the same, yet unknown value for one intrinsic parameter, we in general obtain *quadratic* equations on Z. For example, the assumption of equal aspect ratios leads to the equation:

$$\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{11}\left(\mathsf{U}_j^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_j^{-1}\right)_{22} = \left(\mathsf{U}_j^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_j^{-1}\right)_{11}\left(\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1}\right)_{22}.$$

In analogy to cameras, *quadratic* equations may be derived from assumptions about two or more parallelepiped having the same, yet unknown value for some intrinsic parameter.

In practice, we only use available linear equations. In some minimal cases, quadratic equations as above might be useful to find a unique solution or a finite set of solutions, if the available linear constraints are insufficient.

The situation is different if *all* intrinsic parameters of two (or more) views or parallelepipeds are known to be identical. In that case, we can obtain linear equations instead of quadratic ones, as shown in [Hartley, 1997b]. When the matrices U^i and V^k are scaled such as to have unit determinant, we can write the following component-wise matrix equality between any pair (i, j) of views:

$$\mathsf{U}_i^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_i^{-1} - \mathsf{U}_j^{-\mathsf{T}}\mathsf{Z}\mathsf{U}_j^{-1} = 0_{3\times 3}$$

This represents 6 *linear* equations on Z for each pair of views, among which 4 are independent [Hartley and Zisserman, 2000].

In a similar way, two parallelepipeds having the same shape, leads to a set of 4 independent *linear* equations on Z. This holds even if the parallelepipeds are of different size. Knowing in addition that they are of the same size, gives an additional *linear* equation, which constrain the objects position in space rather than their intrinsic parameters. Currently, we only exploit constraints on individual parallelepipeds (right angles and length ratios), since they are easier to provide for the user.

4.3.5 Complete Algorithm

1. Estimate the canonical projection matrices $\tilde{\mathsf{X}}_{ik}$.
2. Compute missing X_{ik}.
3. Normalize the X_{ik} to unit determinant.
4. Construct the measurement matrix and compute its SVD.

5. From the SVD, extract the matrices U_i and V_k.
6. Establish a linear equation system on Z based on prior knowledge of intrinsic parameters of cameras and parallelepipeds.
7. Solve the system to least squares.
8. If necessary, use additional quadratic equations to resolve the remaining ambiguities on Z.
9. Extract T from Z using Cholesky decomposition.
10. Extract the $\mathsf{K}_i, \mathsf{R}_i, \mathsf{L}_k, \mathsf{S}_k$ from the $\mathsf{U}_i\mathsf{T}^{-1}$ and the $\mathsf{T}\mathsf{V}_k^{\mathsf{T}}$ using e.g. QR-decomposition. Note that at this stage the L_k can only be recovered up to scale, i.e. the parallelepipeds' (relative) sizes remain undetermined.

4.4 Some Minimal Cases

parallelepipeds		cameras	
#	constraint(s)	#	constraint(s)
0		5	• K_1: $\{s, \tau, u_0, v_0\}$ known; K_2: s known • 5 cameras with known s
1	• 1 known length ratio • 1 right angle	4	• 2 cameras with known s and τ • 4 cameras with known s
2	• 2 right angles • 1 right angle and 1 known length ratio	3	• 1 camera with $\{s, u_0, v_0\}$ known • 3 cameras with known s
3	• 3 right angles • 2 right angles and 1 known length ratio • 1 right angle and 2 known length ratios	2	• 1 camera with known s and τ • 1 camera with known u_0 and v_0 • 2 cameras with known s
4	• 3 right angles and 1 known length ratio • 2 right angles and 2 known length ratios	1	• 1 camera with known τ • 1 camera with known s
5	• 3 right angles and 2 length ratios • L_0: 2 and L_1: 3 right angles	0	

Table 4.3: Some minimal cases for the linear calibration algorithm. The problem has five degree of freedom (see text). The table contains a non-exhaustive list of cases where the number of constraints on parallelepipeds and on cameras, sum up to five.

As mentioned in the last section, all constraints provided by knowledge on the cameras and parallelepipeds can be expressed in terms of the 5 independent parameters of the matrix Z. Thus, information about a total of only five intrinsic parameters of cameras or parallelepipeds is in general sufficient to calibrate the whole system. In table 4.3 we give a non-exhaustive list of practical minimal cases. Note however that certain configurations, i.e. relative positioning of cameras and parallelepipeds, represent singularities, depending on the amount of prior information available. Such singularities are discussed in section 4.5.

4.5 Singularities

Many self-calibration algorithms are subject to more or less severe singularities, i.e. there exist situations, where the algorithm is bound to fail. Furthermore, even in situations that are not exactly singular, but close to a singularity, the results become usually very unstable. In this section, we examine the singularities for the linear calibration algorithms described above. First, we study the singularities in the case of one parallelepiped being seen by one camera. We then study some multi-view cases, where we exploit results on critical motions for classical self-calibration.

4.5.1 One Parallelepiped in a Single View

We have studied all possible combinations of *a priori* knowledge, on both camera and parallelepiped intrinsic parameters leading to the linear equations (see section 4.1). In the following we will sketch the methodology followed, give a proof for one sample configuration and provide the results for all configurations studied.

We first formulate the meaning of a singularity in terms of the ingredients of the calibration algorithm. The existence of a singularity in our case means exactly that equation (3.5) has more than one solution for ω and μ that conform to all available *a priori* information, i.e. that there is at least one solution that is different from the true one. It is easy to show that the existence of a singularity does not depend on the relative *position* of the camera and the parallelepiped, only on the relative *orientation* and the *a priori* knowledge on camera and parallelepiped intrinsic parameters.

Let $\mathsf{K} = \mathsf{K}_k\mathsf{K}_u$ be the true calibration matrix, and $\mathsf{K}' = \mathsf{K}_k\mathsf{K}'_u$ the estimated one (we decompose in known and unknown parts, so K' and K share of course the known part K_k). As for parallelepipeds, a similar decomposition into known and unknown parts is not always possible. If however, we only consider constraints arising from prior knowledge leading to linear equations (see section 4.1.2, then we can decompose as above: $\mathsf{L} = \mathsf{L}_u\mathsf{L}_k$ and $\mathsf{L}' = \mathsf{L}'_u\mathsf{L}_k$ are respectively the true and estimated intrinsic parameters of a parallelepiped.

With these definitions, a singularity exists if there are solutions for (3.5) with $\mathsf{K}'_u \neq \mathsf{K}_u$ and $\mathsf{L}'_u \neq \mathsf{L}_u$. From (3.5), it is easy to derive the following equality (using $\mathsf{X} \sim \mathsf{K}_k\mathsf{K}_u\mathsf{R}\mathsf{L}_u\mathsf{L}_k$, $\omega' \sim \mathsf{K}'^{-\mathsf{T}}\mathsf{K}'^{-1}$ and $\mu' \sim \mathsf{L}'^{\mathsf{T}}\mathsf{L}'$):

$$\mathsf{R}^{\mathsf{T}}\mathsf{K}_u^{\mathsf{T}}\mathsf{K'}_u^{-\mathsf{T}}\mathsf{K'}_u^{-1}\mathsf{K}_u\mathsf{R} \sim \mathsf{L}_u^{-\mathsf{T}}\mathsf{L'}_u^{\mathsf{T}}\mathsf{L}'_u\mathsf{L}_u^{-1}.$$

A singularity, as defined above, is then equivalent to the existence of matrices $\omega'' = \mathsf{K}_u^{\mathsf{T}}\mathsf{K'}_u^{-\mathsf{T}}\mathsf{K'}_u^{-1}\mathsf{K}_u$ and $\mu'' = \mathsf{L}_u^{-\mathsf{T}}\mathsf{L'}_u^{\mathsf{T}}\mathsf{L}'_u\mathsf{L}_u^{-1}$, with K'_u and L'_u of the desired form (given by the constraints), but which are different from the identity (otherwise, $\omega' \sim \omega$ and $\mu' \sim \mu$, i.e. we would look at the true solution).

Depending on the *a priori* knowledge, ω'' and μ'' have special forms (as shown in table 4.4 for ω''), independently of the actual values of the known or unknown intrinsic parameters. Hence, the configuration is singular for calibration if the relative orientation R between parallelepiped and camera is such that there are solutions ω'' and μ'' of the required special form and different from the identity, satisfying:

$$\exists(\omega'' \neq \mathbf{I_3},\ \mu'' \neq \mathbf{I_3}) : \mathsf{R}^{\mathsf{T}}\omega''\mathsf{R} \sim \mu'' \qquad (4.18)$$

Based on this definition, it is a rather mechanical, though sometimes tricky, task, to derive singular relative orientations. Table 4.5 shows all singularities for nearly all combinations of the above cases. We explain the singularities in geometrical terms, by describing the relative orientation of the parallelepiped with respect to the camera. In the following paragraphs, we give a few comments on different cases of prior knowledge on the parallelepiped.

Three right angles, two length ratios. In this case, the metric structure of the parallelepiped is completely given (up to scale), and it can be used as a classical calibration object. There are singularities proper to the use of a parallelepiped, but of course the generic singularities described in [Buchanan, 1988] apply here too.

Known camera parameters			
(A) None	(B) τ	(C) u_0, v_0	(D) τ, u_0, v_0
$\begin{pmatrix} a & 0 & d \\ 0 & b & e \\ d & e & c \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & d \\ 0 & 1 & e \\ d & e & c \end{pmatrix}$	$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & c \end{pmatrix}$

Table 4.4: Structure of ω'' depending on prior knowledge on intrinsic camera parameters. Structure of μ'' is similar.

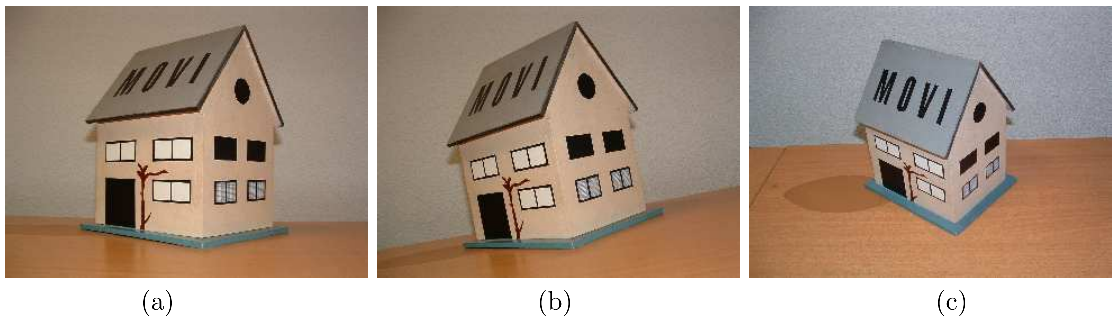


Figure 4.1: Examples of singular and non-singular configurations for calibration based on a parallelepiped with three right angles; (a) The parallelepiped's vertical edge is parallel to the camera's **y**-axis; this configuration is singular if the camera aspect ratio and principal point are not given (cf. cases B-3-0 and C-3-0 in table 4.5). (b) The parallelepiped's vertical edge is parallel to the image plane; this configuration is singular if the camera's principal point is not given (case B-3-0 in table 4.5). (c) A rotation of the camera as shown here removes the singularities of (a) and (b).

Three right angles, one length ratio (cases *-3-1 in table 4.5). In table 4.5, **v** represents the direction of the parallelepiped's edges which are not "involved" in the known length ratio.

Two right angles (cases *-2-* in table 4.5). In this case, the parallelepiped can be visualized as built around two rectangles sharing an edge **v** (see figure 4.2). The role of **w** can be played by one of the two rectangles' edges not parallel to **v**.

An example of the derivation of the singularities for the case C-3-0 is described in the appendix B.

Several singular situations that might occur in practice, are illustrated in figure 4.1.

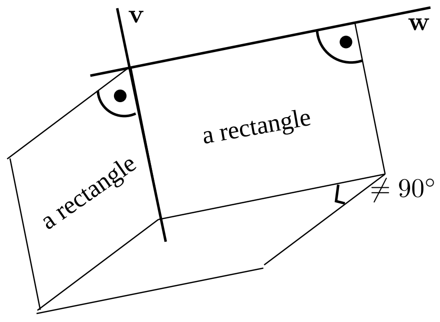


Figure 4.2: Illustration of the case of two right angles.

Case	Conditions for singularity
A-3-1	$\mathbf{v}$ is orthogonal to the $\mathbf{x}$ or $\mathbf{y}$ camera axis
B-3-1	$\mathbf{v}$ is parallel to the optical axis
C-3-1	$\mathbf{v}$ is parallel to any of the three camera axes
D-3-1	$\mathbf{v}$ is parallel to the optical axis
A-3-0	always (3 constraints for 4 camera intrinsics)
B-3-0	any edge is parallel to the image plane
C-3-0	any edge is parallel to a camera axis
D-3-0	any edge is parallel to the optical axis
A-2-2	too difficult to describe
B-2-2	$\mathbf{v}$ $\parallel$ image plane and $\mathbf{w}$ $\parallel$ optical axis or image plane
C-2-2	$\mathbf{v}$ $\parallel$ $\mathbf{x}$ or $\mathbf{y}$ axis and $\mathbf{w}$ at 45° angle with image plane
	$\mathbf{v} \parallel \mathbf{z}$ and $\mathbf{w}$ $\parallel$ image plane and at 45° to both $\mathbf{x}$ and $\mathbf{y}$
D-2-2	never!
A-2-1	always (three constraints for four camera intrinsics)
B-2-1	$\mathbf{v}$ is parallel to the image plane
C-2-1	$\mathbf{v}$ parallel to either camera axis
	$\mathbf{v}$ and $\mathbf{w}$ are both orthogonal to the $\mathbf{x}$ camera axis
	$\mathbf{v}$ and $\mathbf{w}$ are both orthogonal to the $\mathbf{y}$ camera axis
	$\mathbf{v}$ and $\mathbf{w}$ are parallel to the image plane
D-2-1	$\mathbf{v}$ and $\mathbf{w}$ are parallel to the image plane
A-2-0	always (two constraints for four camera intrinsics)
B-2-0	always (two constraints for three camera intrinsics)
C-2-0	$\mathbf{v}$ orthogonal to the $\mathbf{x}$ or $\mathbf{y}$ camera axis or $\parallel$ image plane
D-2-0	$\mathbf{v}$ parallel to image plane or to optical axis

Table 4.5: Singular relative orientations for the case of one parallelepiped seen in one camera, for various combinations of prior knowledge. Cases are denoted X-Y-Z, where X $\in$ {A,B,C,D} refers to table 4.4 and Y and Z are the number of known right angles respectively length ratios. For further explanations, see text.

4.5.2 One Parallelepiped in Multiple Views

Two observations are useful to characterize the singularities in the case when one parallelepiped is seen in multiple images:

- The way the canonic parallelepiped projection matrix X is computed implies that there is only an affine ambiguity in the calibration process. Thus the singularities of calibration of images viewing one parallelepiped are equivalent to singularities of generic self-calibration when the plane at infinity is known;
- It is natural to suppose that the nature of prior knowledge on the intrinsic parameters is the same for all the cameras used, thus all the matrices ω_i belong to only one of the groups defined in table 4.4.

These two observations make it possible to adapt the studies on critical motions for self-calibration. In particular, we use the results presented in [Kahl, 1999] and [Kahl et al., 2000] for scenarios with known plane at infinity. Depending on the type of the knowledge about the camera (cf. table 4.4), the following rotations are singular for the calibration:

A Always critical with fewer than 5 cameras. Critical motions for 5 or more cameras are difficult to explain.

B Critical if the cameras' optical axes point into at most two different directions.

C Critical if one axis of each camera is pointing in some common direction (see figure 4.3).

D Critical if the optical axes of all cameras point into the same direction.

Results for the cases A, B, D were given in [Kahl et al., 2000] and a proof of the case C is given in the appendix B.

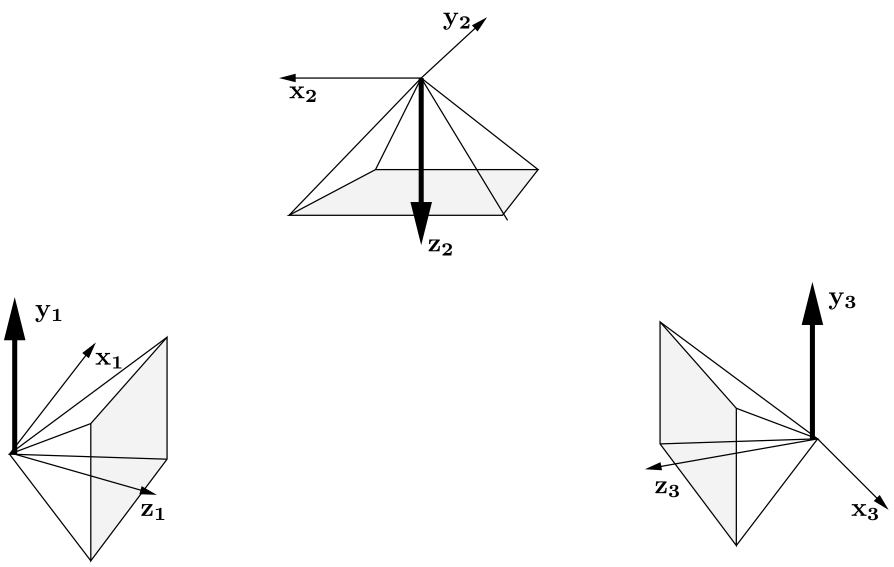


Figure 4.3: Illustration of critical motion for cameras of type C (only principal point is known): one axis per camera (axes $\mathbf{y}_1$, $\mathbf{z}_2$ and $\mathbf{y}_3$) is pointing in some common direction.

Chapter 5

Position and Size

In this chapter we assume that the intrinsic and orientation parameters were already computed, as described in chapter 4, using the leading 3×3 matrices of the canonical projection matrices. In the following we propose a linear framework for the estimation of the positions of the cameras and parallelepipeds, as well as the sizes of the parallelepipeds using the remaining, 4th columns of the canonical projection matrices. In chapter 4 we showed, that when estimating the intrinsic and orientation parameters, it is possible to neglect the scale ambiguity in estimation of the canonical projection matrices. However, when estimating the relative pose and size parameters, it is necessary to take them into consideration. Consequently, there exist configurations, when the prior information on the scene, even if sufficient to calibrate the intrinsic and orientation parameters is not sufficient to estimate the relative pose and size parameters.

5.1 The Algorithm

Consider equation (3.6):

$$\lambda_{ik}\tilde{\mathsf{X}}_{ik} = \mathsf{K}_i \begin{pmatrix}\mathsf{R}_i & \mathbf{t}_i\end{pmatrix} \begin{pmatrix}\mathsf{S}_k & \mathbf{v}_k \\ \mathbf{0}^\mathsf{T} & 1\end{pmatrix} \tilde{\mathsf{L}}_k$$

The leading 3×3 sub-part of the two sides of the equation were used in chapter 4 to compute the intrinsic camera parameters K_i and the rotation matrices R_i and S_k. As for the parallelepipeds' intrinsic parameters L_k, they were computed up to scale only, i.e. up to the "size" of the parallelepipeds.

Let us consider this in detail. In the following, we suppose that the matrices $\tilde{\mathsf{X}}_{ik}$ are already scaled such that the sub-matrices X_{ik} have unit determinant, as in section 4.3, i.e. $\lambda_{ik} = 1$. Let $\bar{\mathsf{K}}_i$ and $\bar{\mathsf{L}}_k$ be the calibration matrices scaled to unit determinant. We know all matrices in the following equation:

$$\mathsf{X}_{ik} = (\mathsf{X}_{ik}\mathbf{x}_{ik})\, \bar{\mathsf{K}}_i \mathsf{R}_i \mathsf{S}_k \bar{\mathsf{L}}_k$$

What we don't know is the size s_k of the parallelepipeds (see section 3.1 for definition of parallelepiped's size). Let us observe the following:

$$\tilde{\mathsf{L}}_k = \begin{pmatrix} s_k\bar{\mathsf{L}}_k & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1\end{pmatrix} \sim \begin{pmatrix} \bar{\mathsf{L}}_k & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1/s_k\end{pmatrix}$$

We may now rewrite equation (3.6):

$$\tilde{\mathsf{X}}_{ik} = \bar{\mathsf{K}}_i \begin{pmatrix}\mathsf{R}_i & \mathbf{t}_i\end{pmatrix} \begin{pmatrix}\mathsf{S}_k & \mathbf{v}_k \\ \mathbf{0}^\mathsf{T} & 1\end{pmatrix} \begin{pmatrix} \bar{\mathsf{L}}_k & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1/s_k\end{pmatrix}$$

Let $\mathbf{x}_{ik}$ be the fourth column of $\tilde{\mathsf{X}}_{ik}$. We have the following equation:

$$\mathbf{x}_{ik} = \bar{\mathsf{K}}_i \begin{pmatrix}\mathsf{R}_i & \mathbf{t}_i\end{pmatrix} \begin{pmatrix}\mathsf{S}_k & \mathbf{v}_k \\ \mathbf{0}^\mathsf{T} & 1\end{pmatrix} \begin{pmatrix} \mathbf{0} \\ 1/s_k\end{pmatrix} = \frac{1}{s_k}\bar{\mathsf{K}}_i(\mathsf{R}_i\mathbf{v}_k + \mathbf{t}_i)$$

From this, we get equations that are linear in all unknowns (s_k, $\mathbf{t}_i$ and $\mathbf{v}_k$):

$$s_k\mathbf{x}_{ik} - \bar{\mathsf{K}}_i\mathsf{R}_i\mathbf{v}_k - \bar{\mathsf{K}}_i\mathbf{t}_i = \mathbf{0}$$

We may now compute the unknowns by solving a straightforward linear least squares problem: minimizing the sum of the squared L_2 norms of the vectors on the left hand side of the above equation, over all camera-parallelepiped pairs.

Note, that there exist configurations, when the prior information on the scene is sufficient to calibrate the intrinsic and orientation parameters but is not sufficient to estimate the relative pose and size parameters. Indeed, in chapter 4 we showed, that when estimating the intrinsic and orientation parameters, it is possible to neglect the scale ambiguity in estimation of the canonical projection matrices. However, when estimating the relative pose and size parameters, it is necessary to take them into consideration. Thus missing information about the scalar factors or insufficient number of parallelepipeds projections causes ambiguities.

Such a configuration is shown in figure 5.1. Even if cameras and parallelepipeds are calibrated, unless the ratio of parallelepipeds' sizes $\frac{s_1}{s_2}$ is known, the depth of parallelepiped 1 in the first image is arbitrary. Consequently there exist one parameter family of solutions for its size and position.

The underconstrained features can be detected using the algorithm presented in the section 6. After chapter 4, this concludes our complete method for intrinsic and extrinsic calibration.

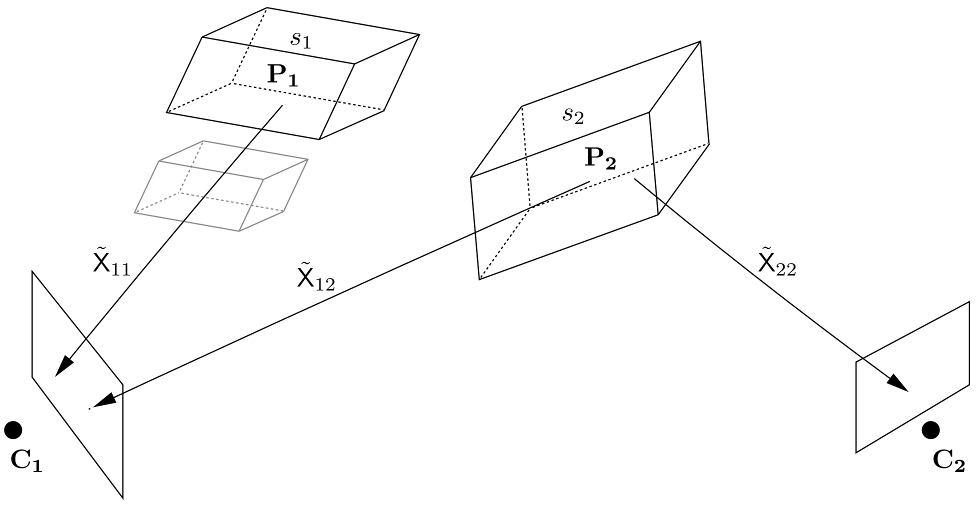

s_1
$\mathbf{P_1}$
s_2
$\mathbf{P_2}$
$\tilde{\mathsf{X}}_{11}$
$\tilde{\mathsf{X}}_{12}$
$\tilde{\mathsf{X}}_{22}$
$\mathbf{C_1}$
$\mathbf{C_2}$

Figure 5.1: Example of a camera-parallelepiped configuration, where the relative pose and size remains ambiguous even when intrinsic and orientation parameters are known. Indeed, the depth of parallelepiped $\mathbf{P}_1$ is arbitrary unless the ratio of parallelepipeds' sizes $\frac{s_1}{s_2}$ is given.

Chapter 6

3D Reconstruction

The calibration approach presented in chapters 3-5 is well adapted to interactive 3D modeling from a few images. It has a major advantage over other methods: simplicity. Indeed, only a small amount of user interaction is needed for both calibration and reconstruction: a few points must be picked in the image to define the primitives' image positions. It thus seems to be an efficient and intuitive way to build models from images of any type, in particular from images taken from the Internet for which no information about the camera is known.

To reconstruct scene elements not belonging to the parallelepiped, constrained by bilinear relations such as collinearity, coplanarity or parallelism, we have implemented a multi-linear reconstruction method, introduced originally in [Wilczkowiak et al., 2003d]. The reconstruction step is actually independent from the calibration method, although it uses the same input in the first step. Interestingly, it allows 3D models to be computed from non-overlapping photographs (see e.g. figure 7.16). The global scheme of our system is presented in figure 6.1.

In this chapter we resume briefly the proposed reconstruction method. First, we propose a method for extraction of uniquely defined variables in linear systems, which is the basis of the

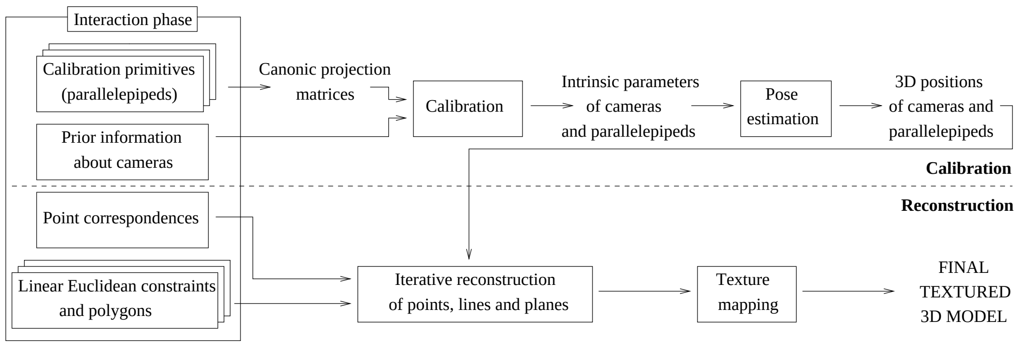

Interaction phase
Calibration primitives (parallelepipeds)
Prior information about cameras
Canonic projection matrices
Calibration
Intrinsic parameters of cameras and parallelepipeds
Pose estimation
3D positions of cameras and parallelepipeds
Calibration
Reconstruction
Point correspondences
Linear Euclidean constraints and polygons
Iterative reconstruction of points, lines and planes
Texture mapping
FINAL TEXTURED 3D MODEL

Figure 6.1: The calibration and reconstruction algorithms.

reconstruction method. Then we describe shortly the algorithm and certain related practical issues. The results are presented in the next chapter.

6.1 Extraction of Uniquely Defined Variables in Linear Systems

Consider the following linear equation system:

$$\mathsf{A}_{m\times n}\mathbf{X}_n = \mathbf{B}_m, \tag{6.1}$$

and assume that the solution for $\mathbf{X}$ is not unique. We then want to determine if there exists a subset of coefficients of $\mathbf{X}$ which can nevertheless be unambiguously estimated. This proves to be very useful in many approaches based on linear constraints, such as intrinsic and extrinsic camera calibration or 3D reconstruction, as described in the following sections of this chapter.

Our approach is based on the analysis of the nullspace of matrix A . Using Singular Value Decomposition (Appendix A), matrix A can be decomposed as follows:

$$\mathsf{A}_{m\times n} = \mathsf{U}_{m\times n}\mathsf{W}_{n\times n}{\mathsf{V}_{n\times n}}^{\mathsf{T}},$$

where the matrices U and V are column-orthogonal and W is diagonal, with the singular values w_i of $\mathbf{A}$ on its diagonal, in decreasing order. Let $\mathbf{X}_0$ be any vector satisfying the equation system. From (A.4), any vector $\mathbf{X}$ satisfying (6.1) must be of the form:

$$\mathbf{X} = \mathbf{X_0} + \sum_{i=r+1}^{n} \lambda_i \mathbf{v}^i, \lambda_i \in \mathbb{R}. \tag{6.2}$$

where vectors $\mathbf{v}^i$ are the columns of V corresponding to zero singular values w_i, constituting the nullspace of A and λ_i are arbitrary scalar factors, and r state for rank of matrix A.

The solution for a coefficient of $\mathbf{X}$, say $\mathbf{X}(k)$, is unique, if

$$\forall \lambda_i : \mathbf{X}(k) = \mathbf{X_0}(k),$$

which implies that

$$\forall \lambda_i : \sum_{i=r+1}^{n} \lambda_i \mathbf{v}^i(k) = 0$$

and is equivalent to:

$$\forall i \in \{r+1, \ldots, n\} : \mathbf{v}^i(k) = 0. \tag{6.3}$$

Hence, all variables $x_k = \mathbf{X}(k)$ corresponding to rows $\mathbf{r}_k$ of matrix $\mathsf{V}[1\ldots n, r+1\ldots n]$, such that $\|\mathbf{r}_k\| = 0$, have unique values $x_k = \mathbf{X}_0(k)$. Geometrically, this means that the axis represented by vector $\mathbf{e}_k^n$ of the solution space $\mathbb{R}^n$ corresponding to such a sufficiently constrained variable is orthogonal to the nullspace of matrix A.

Choice of thresholds. Using equation (6.3), it is in principle straightforward to split the unknowns of the system into well-defined and ambiguous ones. Note that this requires deciding if certain numerical values (singular values and coefficients $\mathbf{v}^i(k)$) are equal to zero. It is well known that due to noise and round-off errors, the numerically computed singular values of a matrix are never exactly zero. We thus use the approach proposed in [Press et al., 1988; Golub and van Loan, 1989; Bjorck, 1990] where singular values w_i are set to 0 when they satisfy the following condition: $w_i < \epsilon\, w_1$), for a given threshold ϵ. Similarly, the detection of the well-constrained set of variables is based on the comparison of elements $\mathbf{v}^i(k)$ with a given threshold ϵ_1. Of course, the results of the method depend on the choice of the thresholds ϵ and ϵ_1, which may depend themselves on the underlying application[1]. If thresholds are too large, then there is a possibility that some variables which are sufficiently constrained, will be classified as underconstrained. On the other hand, if they are too small, some underconstrained variables will be classified as having been well estimated, disadvantageously influencing the overall results of an underlying algorithm.

Applications and extensions. The application domain of the proposed approach covers all computer vision algorithms based on linear algebra. In particular, it can be useful in any calibration algorithm based on linear equations (an example for plane-based calibration is given in [Wilczkowiak et al., 2003a]), as well as for the reconstruction methods, as explained on an example described in the next section.

A main advantage of the proposed approach is simplicity. The test for underconstrained variables needs very small additional computation effort. Indeed, the necessary SVD of the constraint matrix, is usually computed anyway to solve the linear problems. A main drawback is the reliance of the approach on the predefined thresholds. It would be advantageous to incorporate to the method a statistical analysis and uncertainty propagation of the data.

6.2 Multi-linear Reconstruction System

In this section we propose an approach for interactive scene modelling. First, in section 6.2.1 a brief overview of available objects and constraints is given, followed by a general study of how thay can be exploited in the system. Then in section 6.2.2 the algorithm implemented in our system is detailed and illustrated by a detailed example in section 6.2.3. Finally, an approach for incorporation of soft and hard constraints into the system is given in section 6.2.4.

6.2.1 Overview

The scene is modelled using points, lines and planes, represented like described in section 1.4.1. Three general types of constraints (described in details in section 1.4.1) between objects are considered:

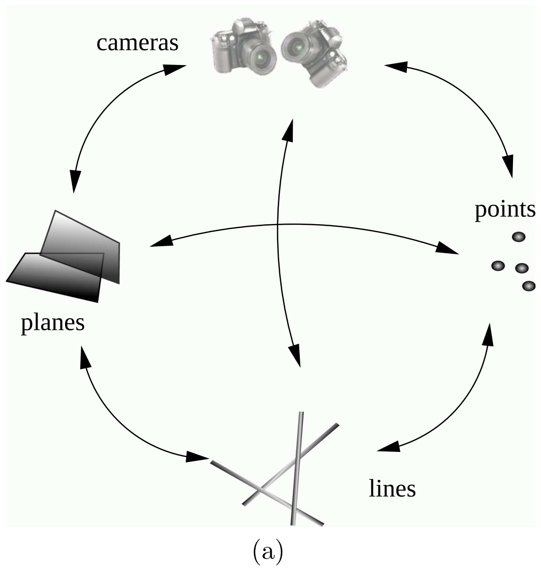
Projections. Every known projection of a point or a line gives linear constraints on the 3D coordinates of the corresponding object. Reciprocally, known 3D points or lines give linear constraints on the camera projection matrices. Moreover, geometrical constraints or known plane homographies allow to constrain directly the intrinsic camera parameters.

Bilinear constraints. Incidence, parallelism and orthogonality between two objects $\mathbf{X}$, $\mathbf{Y}$ can be expressed as bilinear forms $\mathsf{F}(\mathbf{X}, \mathbf{Y}) = 0$ [Heuel, 2001; Poulin et al., 1998; Hartley and Zisserman, 2000]. Thus, knowing coordinates of one of the objects induces linear constraints on the other one.

Affine point configurations. Relations like points lying on a parallelogram or symmetry are useful in practice and are linear in terms of all the involved objects.

The scene is represented by a graph whose nodes are objects and whose edges are constraints. For example, four coplanar points will be represented by five nodes (4 points and 1 plane) and 4 incidence constraints (each point with the plane). Except for the affine point configurations, all the geometrical relations incorporated into the system involve two objects of different types and can be used to constrain any of the related objects (see figure 6.2–(a)). These relations are bilinear with respect to coordinates of the two related objects. Thus, they can be used in a linear framework

[1]However, we observed in our experiments that the choice for ϵ_1 was not very critical.



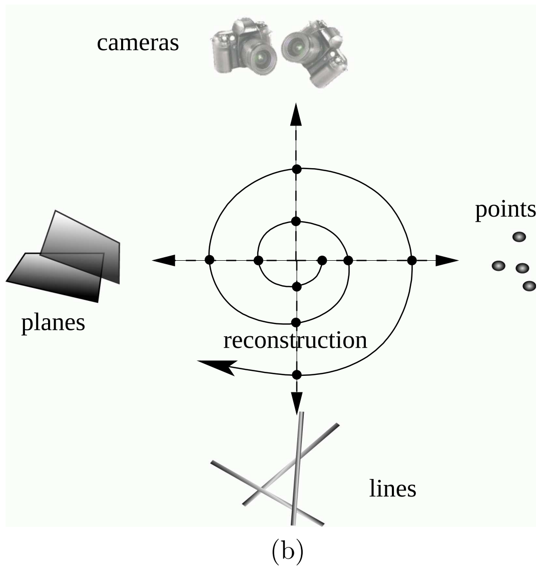


Figure 6.2: The concept of the multi-linear reconstruction algorithm; (a) The projections and scene constraints are bilinear and can be used to reconstruct any of the related objects. (b) The spiral form represents the reconstruction algorithm and corresponds to the iteratively growing set of reconstructed objects and, in consequence, accessible constraints. Every dot illustrates a single step of the algorithm, when the constraint matrix is formed and computed, solving the active constraints for one type of objects.

only when at least coordinates of one of the involved objects are known. For example, known 3D positions of points can be used simultaneously to constrain the camera projection matrices [Tsai, 1986] as well as calibrated cameras alone or together with some prior scene information can be used simultaneously to compute 3D points positions [Hartley and Zisserman, 2000; Shum et al., 1998; Robertson and Cipolla, 2000; Grossmann and Santos-Victor, 2002]. However, if a method proceeds in a single step, it is possible that not all the accessible data is used. For example, it is not possible to impose the orthogonality of two scene directions and use them in the same step to constrain the 3D scene points. Similarly, when a 3D line direction is not known, it is not possible to use the collinearity constraint on the associated points (a simple example of such a point configuration, which can be recovered only in three steps is shown in figure 2.5).

There are two reasons why the extraction of the sufficiently constrained variables in the system defined above is crucial for the efficiency of the algorithm.

Firstly, at each iteration underconstrained variables may exist. Especially at initial iterations, only few constraints are active: the coordinates of 3D lines and planes are still unknown, thus only the projection and symmetry/parallelogram constraints are active. Also, even when the reconstruction process is in an advanced stage, it is common that some objects are underconstrained due to missing or redundant data. By redundant data we mean that the result is very sensitive to noise (e.g. 2 projections available for a 3D point, but for 2 views with a very small baseline; or, a 3D point defined to be the intersection of a line and a plane, but when these two are near parallel).

Secondly, and contrary to existing approaches, our system allows constraints influencing several objects at once, which means that equation systems to be solved may contain simultaneously well constrained and underconstrained unknowns. Without selecting the solvable subset of unknowns, one would either propagate wrong values to subsequent iterations, or would have to stop the whole algorithm. In the following, we give details on the implementation of our algorithm and explain, how the introduced geometrical dependencies can be treated as soft or hard constraints. The experimental evaluation of the method can be found in [Wilczkowiak et al., 2003b,a]. All the scenes illustrating our calibration approach presented in part II, as well as initial values used for the reconstruction approach presented in part III, were computed using the above method.

6.2.2 Algorithm

In the previous section, we have detailed the constraints used in the system and sketched how they can be exploined for the reconstruction. Let us consider now some practical issues concerning the implemented version of the algorithm. Firstly, we use precalibrated cameras and do not update their parameters during the reconstruction process, but as suggested in the last section, it is easy to add the re-calibration step into the system.

Secondly, due to the fact that any of the linear constraints used in the system do not involve both points and lines or planes at the same time, point features and line and plane features are computed in two separated steps. Computing line and plane features together allows, if desired, to represent parallel line and plane normal directions by a single vector and use the constraints on lines and planes simultaneously.

Finally, we propose the following reconstruction algorithm:

```
1:  while !stop_condition do
2:    for objects=points,lines+planes: do
3:      N:=$\sum_{i=1}^{n}$ nb_of_coordinates(objects[$i$])
4:      initialize an empty linear equation system $\mathsf{A}_{0\times N} X_{N\times 1} = \mathsf{B}_{0\times 1}$
5:      compute the indexing function (bijection) $\mathsf{F} : \text{idx} \rightarrow (i, j); \text{idx} \in [1 \ldots \text{N}]$,
        where idx is the index in $X_{N\times 1}$ of the $j$-th coordinate of the $i$-th object.
6:      for all constraint $c[k]$: do
7:        compute
          $(\mathsf{A}^k_{m_k\times N}, B^k_{m_k\times 1})$ := equations($c[k]$.type, $c[k]$.objects)
8:        add equations to the system: $\mathsf{A} := \begin{bmatrix} \mathsf{A} \\ \mathsf{A}^k \end{bmatrix} \mathsf{B}^k := \begin{bmatrix} \mathsf{B} \\ \mathsf{B}^k \end{bmatrix}$
9:      end for
10:     solve $\mathsf{A}X = \mathsf{B}$
11:     for idx=1...$N$: do
12:       if variable_computed(idx) then
13:         set $(i, j) := \mathsf{F}$(idx)
14:         set objects[$i$].coords[$j$]:=$\mathbf{X}$(idx)
15:       end if
16:     end for
17:   end for
18: end while
```

Algorithm 1: Iterative Reconstruction Algorithm

6.2.3 Example

6.2.3.1 Numerical Analysis

The principle of our method is depicted in Fig. 6.3. The elements in this figure are parts of more complex model shown in section 6.2.3.2.

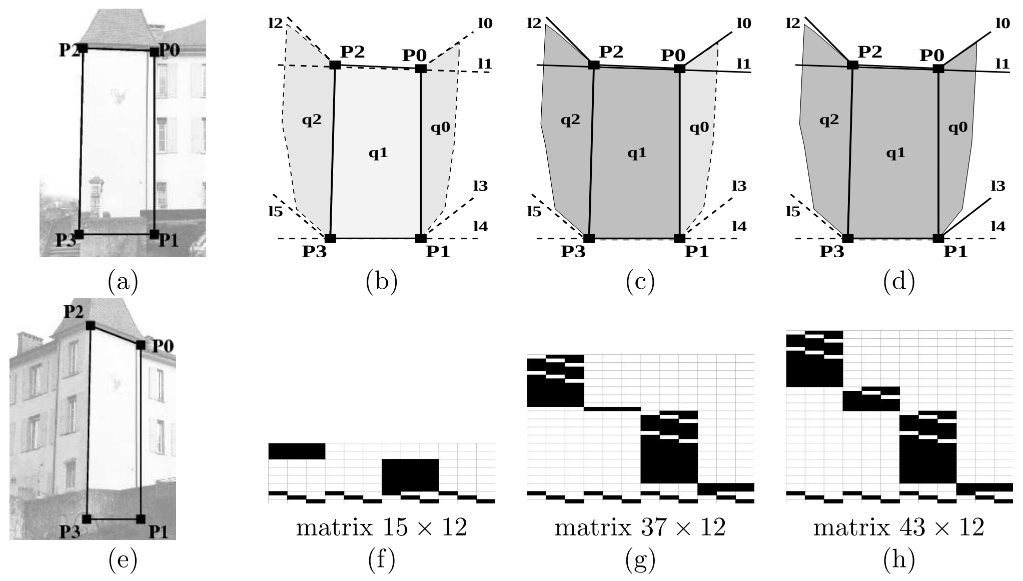


Figure 6.3: Illustration of the reconstruction process. 1st column: fragments of the original images with marked interest points. P1, P3 are occluded in all images. 2nd, 3rd, 4th column: the constraints influencing the interest points. 1st row: the relations between the considered objects. Lines and planes which can influence the interest points P1 - P3 (their coordinates are reconstructed at this step) are marked with black continuous line (lines) or in dark gray (planes); 2nd row: the corresponding constraint matrices. Black fields correspond to non-zero matrix elements.

The nullspace analysis and the associated variable splitting performed at each iteration make all the available information effective. The configuration shown in Fig. 6.3 consists of four points **P0-P3** related by a parallelogram constraint. The points **P0** and **P2** are visible in several images, while the points **P1** and **P3** are occluded by a fence. Points **P0-P3** are incident with 3 planes and 6 lines (See Fig. 6.3–(b) - 6.3–(d)), which are connected with other elements of the model by incidence and parallelism constraints. The columns 2-4 in Fig. 6.3 show the reconstruction achieved at the 1st, 2nd and 4th iteration. The corresponding numerical results are displayed in Tab. 6.1. At each reconstruction step (See Sec. 6.2.2) all the equations resulting from projection and geometrical constraints are added to the common equation system. In this particular case, the system contains 12 variables: the coordinates (x, y, z) of the points **P0-P3**.

1st iteration. The matrix $\mathbf{A}_{15\times12}$ contains only the equations corresponding to the projection and the parallelogram constraints. The resulting matrix structure is shown in Fig. 6.3–(b). A straightforward analysis of this matrix shows that the variables associated to the points **P1** and **P3** are underconstrained (only 3 equations are defined for 6 variables). The nullspace analysis confirms this result. The singular values of the constraint matrix are detailed in Fig. 6.1. The choice of the 3 singular values which should be zeroed is obvious. The vectors spanning the nullspace correspond to the zeroed singular values. The rows of the nullspace basis which should be zeroed are also easily detected. The process result confirms the intuition that only the values corresponding to coordinates of points **P0** and **P2** are well-constrained.

2nd iteration. The equations yielded by the planes **q1**, **q2** and the lines **l0**, **l1**, **l2** reconstructed in the first iteration are added to the new system matrix $\mathbf{A}_{37\times12}$. The structure of this matrix

(Fig. 6.3–(c)) suggests that there are enough constraints to solve the whole system. However, the nullspace analysis (see 4th column of Tab. 6.1) shows that the system is still degenerate. Indeed, the parallelogram constraint is partially redundant with the constraints coming from the incidences of points **P0-P3** with the plane **q1**.

3rd iteration. The plane **q0** and the line **l3** are reconstructed. The resulting constraints are sufficient to reconstruct the whole configuration.

iteration 0				iteration 1	
W	$\Phi(\mathbf{A})$			W	$\Phi(\mathbf{A})$
3.9	*-9.4e-17*	*1.9e-17*	*3.6e-18*	1.6e+02	*3.2e-17*
3.4	*1.5e-16*	*1.7e-17*	*-6.3e-17*	1.3e+02	*3.6e-17*
3.2	*-7.1e-17*	*2.5e-18*	*4.3e-17*	77	*2.8e-17*
2.7	-0.63	0.14	-0.29	77	0.71
2.3	-0.23	0.25	0.62	3.8	-0.046
1.5	-0.23	-0.65	0.18	3.2	-0.0041
1.3	*9.2e-17*	*-1.7e-17*	*3.8e-17*	1.7	*-3.2e-17*
1.2	*-8.3e-17*	*-2.7e-17*	*-1.7e-17*	1.6	*-2.1e-17*
0.77	*-4.1e-17*	*-3.0e-18*	*-3.3e-18*	1.4	*-7.3e-17*
2.1e-16	-0.63	0.14	-0.29	1.0	0.71
8.3e-17	-0.23	0.25	0.62	0.61	-0.046
3.2e-17	-0.23	-0.65	0.18	1.1e-15	-0.0041

Table 6.1: Numerical results obtained for example in Fig. 6.3. For each iteration the 1st column contains the singular values of the constraint matrix A. The horizontal line separates the values which were zeroed. Following columns contain vectors forming the matrix nullspace, with zeroed values in italic. Variables corresponding to rows containing only zero values were classified as reconstructed. These values correspond to coordinates of points P0 and P2 in Fig. 6.3.

6.2.3.2 Full Reconstruction

The full reconstruction of the castle is based on 7 images of a castle, one of which is shown in Fig. 6.4–(a). The cameras used were calibrated using mixed approaches [Sturm and Maybank, 1999a; Wilczkowiak et al., 2001]. This reconstruction raises several difficulties:

- the images overlap only slightly, decreasing the quality of the camera calibration;
- some of the model points are either not visible in any image or visible only in image regions where the camera distortion, which is not taken into account, is important;
- the geometrical constraints that can improve the reconstruction are not numerous: vertical edges of the castle are slightly pointing to the center, and its faces are not parallel (see Fig. 6.4–(b)). Thus geometric constraints are rather used to reconstruct castle elements which are occluded in images (e.g. see Sec. 6.2.3).

Fig. 6.4–(b) displays the map of the castle with the reprojected model points. Points marked with circles are those reconstructed from geometrical constraints only. Experiments were also conducted using a ground plane map of the castle as an additional image for the reconstruction. However it did not significantly change the results. The reconstructed model is shown in Fig. 6.4–(c).

The second row in Fig. 6.4 shows results for the first three iterations of the reconstruction. Again, at each iteration the model is enriched by new objects computed using the previously reconstructed set and the newly defined constraints.

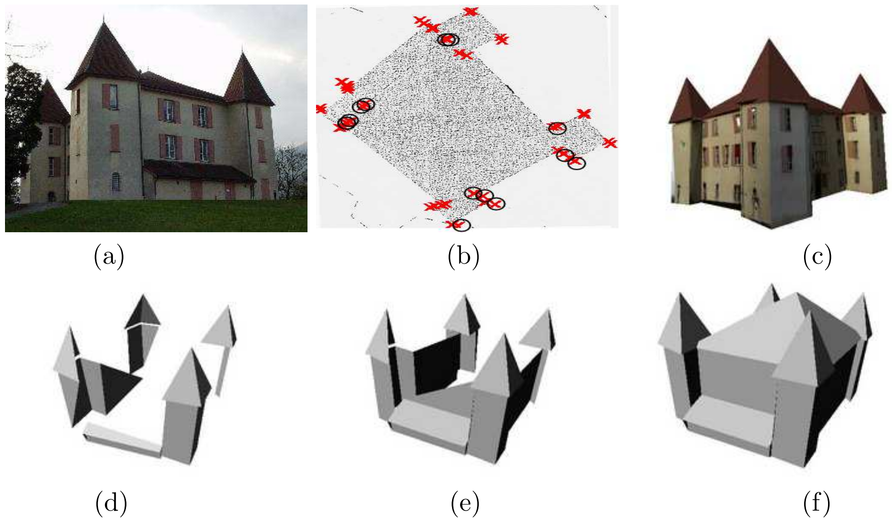


Figure 6.4: (a) One of the seven images used for the reconstruction; (b) The castle plane; (c) The textured 3D model; (d)-(f) Screen-shots of the model at different steps of the reconstruction process.

6.2.4 Soft and Hard Constraints

While defining the geometric constraints, the user might wish to enforce some "highly reliable" constraints in an exact manner, instead of incorporating them in a least squares computation. This of course is only possible if these constraints do not contradict one another.

Let us consider a system with m equations, where r of them are to be respected *exactly*. Without loss of generality, we can permute the rows of $\mathbf{A}$ and the coefficients of $\mathbf{B}$ and write:

$$\mathsf{A}_{m\times n} = \begin{bmatrix} \mathsf{A}_e \;\; {}_{r\times n} \\ \mathsf{A}'_{(m-r)\times n} \end{bmatrix};\; \mathsf{B}_{m\times n} = \begin{bmatrix} \mathsf{B}_e \;\; {}_{r\times n} \\ \mathsf{B}'_{(m-r)\times n} \end{bmatrix} \tag{6.4}$$

where A_e and B_e correspond to equations to be respected exactly and A' B' to equations whose residuals are to be minimised (subject to the exact constraints), i.e.:

find the vector $\mathbf{X}$ minimizing the function

$$f(\mathbf{X}) = \|\mathsf{A}'\mathbf{X} - \mathbf{B}'\|$$

and satisfying the linear constraints $\mathsf{A}_e\mathbf{X} = \mathbf{B}_e$.

The solution to this problem can be found using standard constrained optimization techniques, e.g. Langrange multipliers (see e.g. [Gill et al., 1981]). [Shum et al., 1998] proposes to use the properties of QR factorisation to solve this problem. We propose another method, based on the SVD (see also [Hartley and Zisserman, 2000], Appendix **5**).

Let us consider the system $\mathsf{A}_e\mathbf{X} = \mathbf{B}_e$. As mentioned in section A, the set of solutions can be expressed using the SVD of A_e:

$$\begin{array}{c} \mathbf{X}_e = \mathbf{X}_{0e} + \sum_{k=r+1}^{n} \lambda_k \mathbf{v}^k;\; \lambda_k \in \mathbb{R}; \\ \mathbf{X}_{0e} = \mathsf{A}_{\mathbf{e}}^{+}\mathbf{B}_e; \end{array} \tag{6.5}$$

All vectors $\mathbf{X}_e$ respect the equations $1 \ldots r$ exactly. Now the resolution of the system $\mathsf{A}\mathbf{X} = \mathbf{B}$ is reduced to finding coefficients λ_k such that $\mathbf{X}_e$ is satisfying the equation $\mathsf{A}'\mathbf{X}_e = \mathbf{B}'$ in the optimal way (usually, least squares).

Using equation (6.5) we can reformulate the problem:

$$\begin{aligned} \mathsf{A}'\mathbf{X}_e &= \mathbf{B}' \\ \Leftrightarrow \mathsf{A}'\mathsf{A}_{\mathbf{e}}^{+}\mathbf{B}_e + \mathsf{A}'(\sum_{k=r+1}^{n} \lambda_k \mathbf{v}^k) &= \mathbf{B}' \\ \Leftrightarrow \overbrace{\mathsf{A}' \begin{bmatrix} \mathbf{v}^{r+1} & \cdots & \mathbf{v}^n \end{bmatrix}}^{\mathsf{A}''} \begin{bmatrix} \lambda_{r+1} \\ \lambda_{r+2} \\ \vdots \\ \lambda_n \end{bmatrix} &= \overbrace{\mathbf{B}' - \mathbf{A}'\mathsf{A}_{\mathbf{e}}^{+}\mathbf{B}_e}^{\mathbf{B}''} \end{aligned}$$

This is again a linear minimisation problem wich can be solved using the SVD decomposition. The undetermined values can be detected like described in the last section. The advantage over using e.g. Lagrange multipliers, is that here, the equation system is of smaller size.

Chapter 7

Experimental Results

In this chapter we present experimental results for calibration of from both, synthetic and real images. The results of real image calibration are validated via subsequent reconstructions of full scenes, containing other primitives besides the parallelepipeds. The experiments on synthetic and real data are respectively described in sections 7.1 and 7.2.

7.1 Synthetic Scenes

This section presents results of evaluation of the performance of the calibration method described in section 4.3. The tests performed with 600×400 synthetic images, taken by cameras with the following intrinsic parameters: $(\alpha_u, \alpha_v, s, u_0, v_0) = (1000, 1000, 0, 300, 200)$. Parallelepiped parameters were varying over the different tests. The most important parameter of the experiments is the relative orientation between parallelepipeds and cameras. For a given orientation, parallelepiped vertices were projected into the images, and random Gaussian noise was added to image points. For a given setting (relative orientation, standard deviation of noise, etc.), 100 such data sets were created randomly, and used as input for calibration. Calibration was considered to have failed if any of the estimated matrices ω' or μ' was not positive definite (in that case, K' or L' can not be retrieved). In the following, we indicate the number of failures, as well as median values and standard deviations for estimated parameters (computed using valid calibration results only).

In the first experiment, we also compare our method with an approach based on vanishing points.

7.1.1 A minimal case: 1 camera and 1 parallelepiped

This test was performed mostly to compare the performance of the calibration method described in this part of the thesis and calibration based on vanishing points, especially in the proximity of singular configurations. To estimate vanishing points, the MLE estimator method described in [Liebowitz and Zisserman, 1998] was used. Then intrinsic camera parameters were computed using standard method [Caprile and Torre, 1990]. In all tests, only 6 parallelepiped vertices are used for the calibration, which is the most common case in practice.

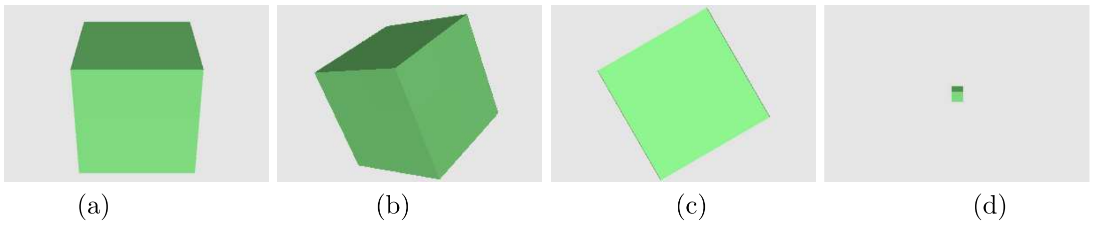


Figure 7.1: Parallelepiped and camera positions in the experiment 7.1.1. (a) 1st camera position; (b) intermediate camera position; (c) maximum parallelepiped rotation; (d) minimal parallelepiped size.

Setup. Prior information used were: the parallelepiped has only right angles and known camera parameters were $(s, \tau) = (0, 1)$ (corresponding to case B-3-0 in section 4.5). This is one of the minimal cases for calibration.

Tests were performed for different sizes and orientations of the parallelepiped and Gaussian noise with a standard deviation of σ=1. The relative parallelepiped-camera rotation varies from position P_0 (x axes of parallelepiped and camera are parallel, cf. figure 7.1–(a)) to position P_n (the x and y axes of the parallelepiped are parallel to the image plane, cf. figure 7.1–(c)). According to section 4.5, both these positions are singular for the calibration. The maximal and minimal sizes of parallelepiped are shown on figure 7.1–(a) and 7.1–(d), respectively.

Results. Figure 7.2 shows the number of successful calibrations, median calibration values for α_v and errors in rotation estimation as functions of the rotation angle and the size of the parallelepiped.

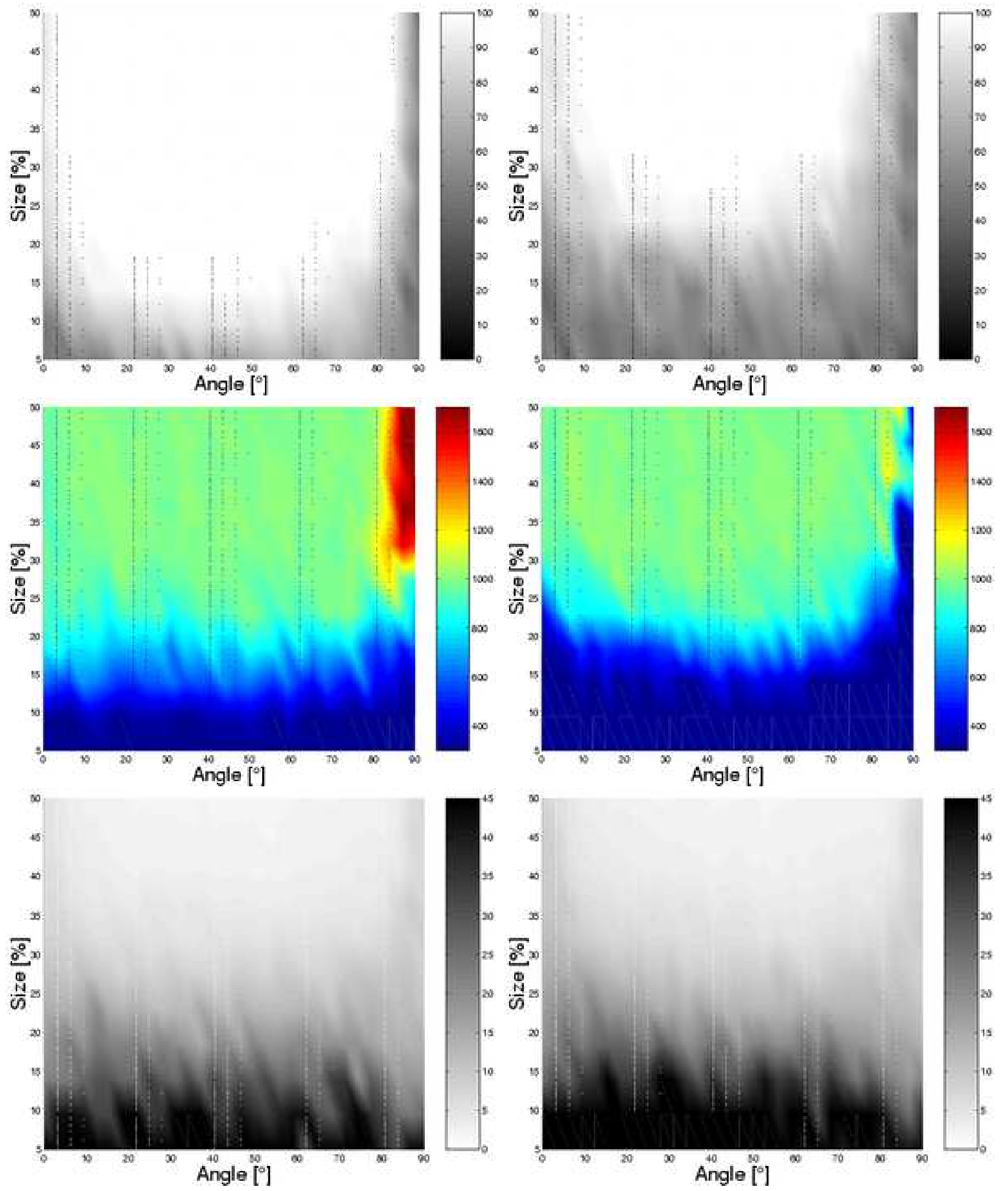


Figure 7.2: Calibration results as a function of the size and relative camera-parallelepiped rotation angle. First and second column are corresponding, respectively, to parallelepiped-based and vanishing point-based calibration; Rows are corresponding to (1) Number of successful calibrations; Black color correspond to 0% and white to 100% of successful calibrations;(2) Median values for α_v. Dark colors, i.e. red and blue, correspond to important calibration errors (estimated values are larger than 150% or smaller than 50% of the true values, respectively); (3) Median errors in relative camera-parallelepiped rotation estimation.

The values on x axes of figures correspond to angle between relative parallelepiped-camera rotation. The extremities correspond to the positions P_0 to P_n, which are singular for the calibration, thus results are expected to be instable at their proximity. Values on x axes of figures correspond to sizes of parallelepiped in image. Results are shown for both the parallelepiped based approach (first column) and the vanishing point approach (second column). It can be seen that the range of rotations where calibration succeeds, is about 5° larger on the rotation axis and 10% on the size axis for the parallelepiped-based method. Results of successful calibrations are also slightly more accurate for the parallelepiped method.

7.1.2 Scenario with two cameras

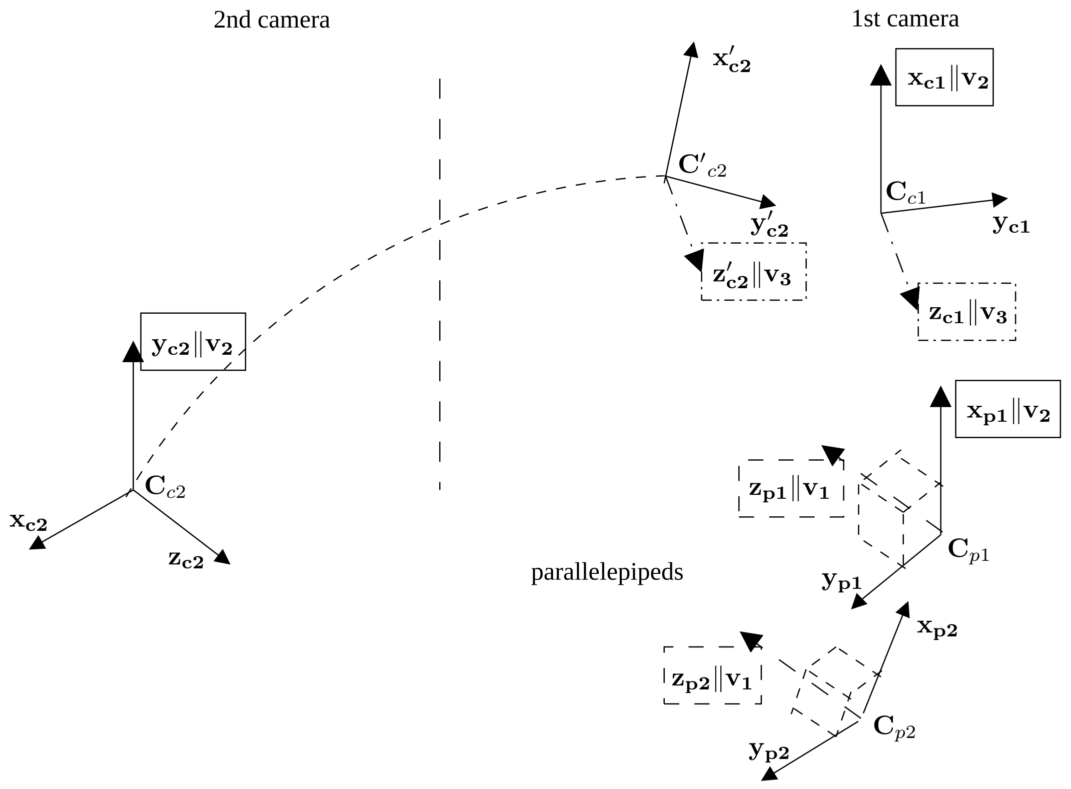


Figure 7.3: Experimental setup for tests on synthetic data: one camera is static ($\mathbf{C}_{c1}$) and the second one moving (from position $\mathbf{C}_{c2}$ to $\mathbf{C}'_{c2}$). They are watching two parallelepipeds (at $\mathbf{C}_{p1}$ and $\mathbf{C}_{p2}$). Parallel directions are drawn using the same line style and label $\mathbf{v}_{i\in\{1,2,3\}}$.

The goal of this test was to evaluate calibration results in the proximity of singular positions, and with different numbers of calibration primitives as well as different types of prior information. We also compare the results of tests obtained by the reference primitive method described in section 4.2 with the factorization method described in section 4.3.

7.1.2.1 Scene Configuration

Setup. The experimental setup is sketched in figure 7.3. Tests were performed using parallelepipeds with parameters $(l_1, l_2, l_3, \theta_1, \theta_2, \theta_3) = (1, 1, 1, 90°, 90°, 90°)$ and

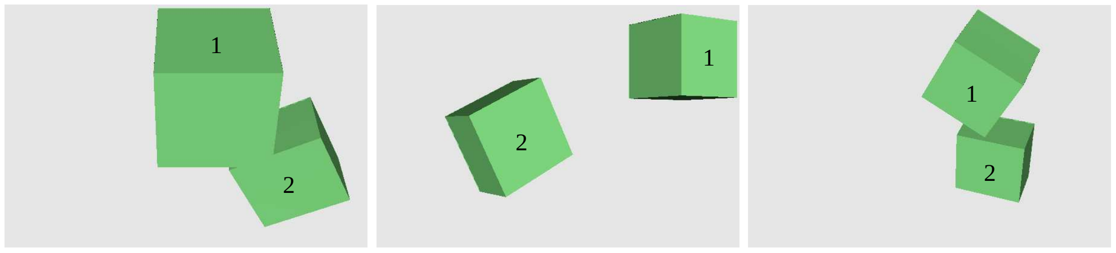


Figure 7.4: Three projections of parallelepipeds as seen from: (a) camera 1 ($\mathbf{O}_{c1}$); (b)-(c) camera 2 in initial and final positions ($\mathbf{O}_{c2}$ and $\mathbf{O}'_{c2}$) respectively.

$(l_1, l_2, l_3, \theta_1, \theta_2, \theta_3) = (4, 2, 3, 90°, 90°, 90°)$. Two cameras with zero skew and known principal points (case C, cf. section 4.5.2) are watching these parallelepipeds. The first camera is static and its x-axis is parallel to the first parallelepiped's x-axis. The second camera is rotating from the position P_0, where its y-axis is parallel to the first parallelepiped's x-axis (and thus parallel to the first camera's x-axis) to the position P_n where its z-axis is parallel to the first camera's z-axis. As explained below, both positions P_0 and P_n correspond to singular configurations.

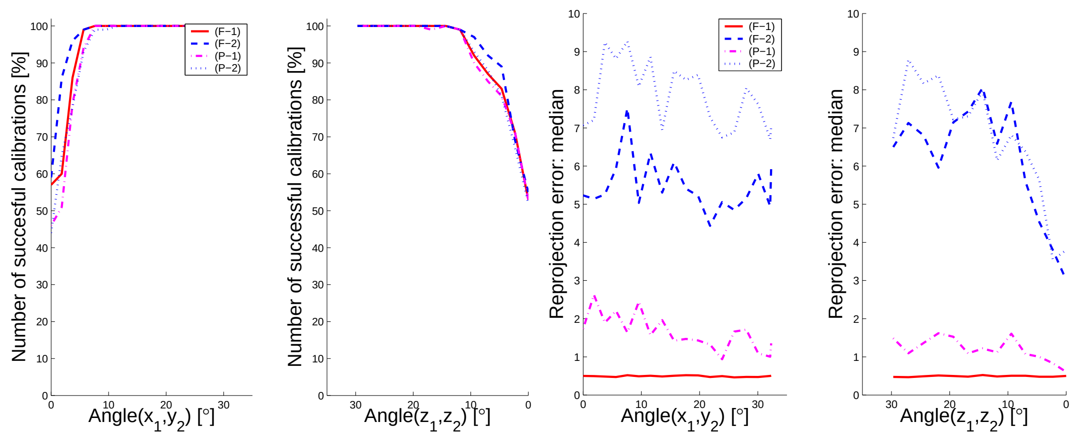
All charts illustrating the results are separated into two parts, depending on the smallest relative angle between camera axes, or equivalently, the proximity to a singular position. The first part shows errors as a function of the angle between the first camera's y-axis and the 2nd camera's x-axis (these two axes being parallel in the initial position P_0). The second part shows results as a function of the angle between the z-axes of both cameras (parallel in the final position P_n). Thus left extremity of left parts and right proximity of right parts of graphs correspond to singular configurations, where the results are expected to be unstable. Labels (F-1) and (F-2) correspond to calibration using the factorization method based on 1 and 2 parallelepipeds, and labels (P-1) and (P-2) correspond to calibration using the reference primitive method. Here, white Gaussian noise with a $\sigma = 1$ pixel standard deviation was added to image point positions.

Singular configurations. Using no prior information about the parallelepipeds is equivalent to doing self-calibration of our 2 cameras, with a known plane at infinity[1]. According to section 4.5.2, singularities for the camera type C occur when two of the cameras' axes (optical axes or pixel axes) are parallel. Thus, both positions P_0 and P_n are singular. Note however that the focal lengths $\alpha_{v,1}$ of the first camera, and $\alpha_{u,2}$ of the second camera can be determined in P_0, and that camera aspect ratios τ_1 and τ_2 of both cameras can be determined in P_n (see appendix B). The charts 7.5-7.8, showing the intrinsic and extrinsic calibration results, reflect clearly the proximities of the singular positions (left and right extremity of figures).

The situation is different, when the information about the right parallelepiped angles is added to the calibration process. This corresponds to case C-3-0 in section 4.5. Singularities only occur when each of the parallelepipeds has one axis parallel to one common direction. This is the case with position P_0 for calibration based on the first parallelepiped only, where the estimation of $\alpha_{v,2}$ remains ambiguous.

Let us study in more details the results of the intrinsic and extrinsic calibration.

[1]Note that the presence of a second parallelepiped, as long as its Euclidean shape is completely unknown, does not remove any calibration ambiguity.



(a) (b)

Figure 7.5: (a) Number of successful calibrations; (b) Median values of reprojection errors. Plots described by (F-*) and (P-*) correspond, respectively, to factorization and reference primitive methods; plots described by (*-1) and (*-2) correspond to calibration based on 1 and 2 parallelepipeds.

7.1.2.2 Results

Discussion of the results for intrinsic calibration. Let us first consider the behavior of our calibration method in a proximity of singular configurations (left and right extremities of figures 7.5-7.7). It can be seen that at both initial and final positions P_0 and P_n, calibration is very unstable, as expected. However, when the minimal angle between the camera axes is larger than 15° the method can be considered stable. All calibrations are successful (figure 7.5–(a)), the relative error on the obtained median values is not larger than 7% (figure 7.6).

As expected, calibration results obtained with the reference primitive method are less stable than those obtained with the factorization method. Indeed, as shown in figure 7.6 the relative error on median values obtained with the reference primitive method is up to 9% (vs 7% for the factorization method). Also, reprojection error is more important ($\sim$ 1.5 pixel for 1 primitive and $\sim$ 8 pixels for 2 primitive based calibration vs $\sim$ 0.5 pixel and $\sim$ 6 pixels for the factorization method).

Naturally, calibration results are more stable using two parallelepipeds than with a single one. First, using larger number of primitives decreases the possibility for singulararity. Figure 7.7–(b) shows results for the second camera for one and two parallelepiped based calibration using information on right parallelepiped angles. While a relative position between the first parallelepiped and second camera is singular for the calibration, adding a second parallelepiped stabilizes the configuration. In non-singular configurations the larger number of primitives and related increased number of equations result in more accurate calibration (figures 7.6- 7.7). However, introducing additional primitives increases the reprojection error (see figure 7.5–(b)). Indeed, the calibration tends to conform both to prior information on scene and projection information. The more constraints are given on the scene structure, the smaller is impact of projection constraints, and consequently, they are not as well satisfied as in a less constrained scene.

Comparing figures 7.6 and 7.7 it can be seen that using the information about the right parallelepiped angles decreases importantly the number of singularities. Also in non-singular positions it increases slightly the stability of the method.

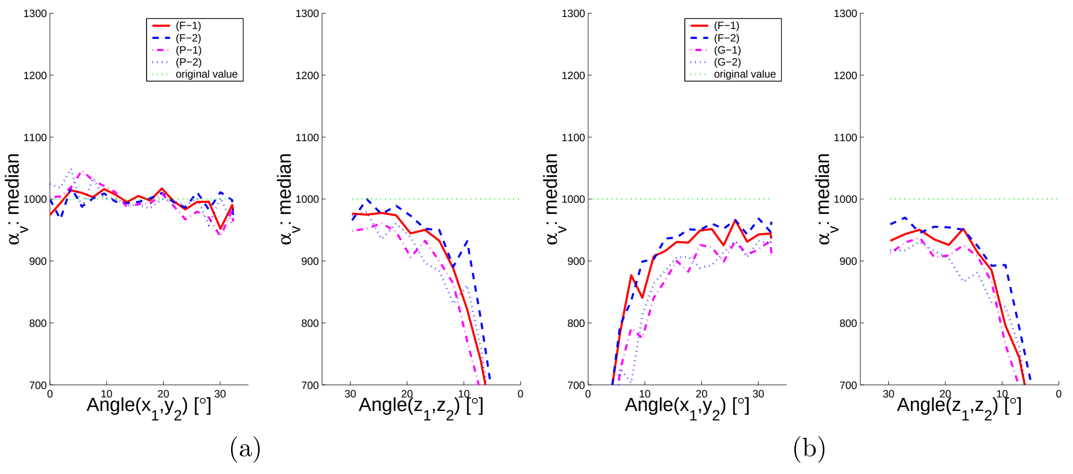


Figure 7.6: Median values obtained for the focal length of (a) first camera; (b) second camera. Notations are similar to figure 7.5.

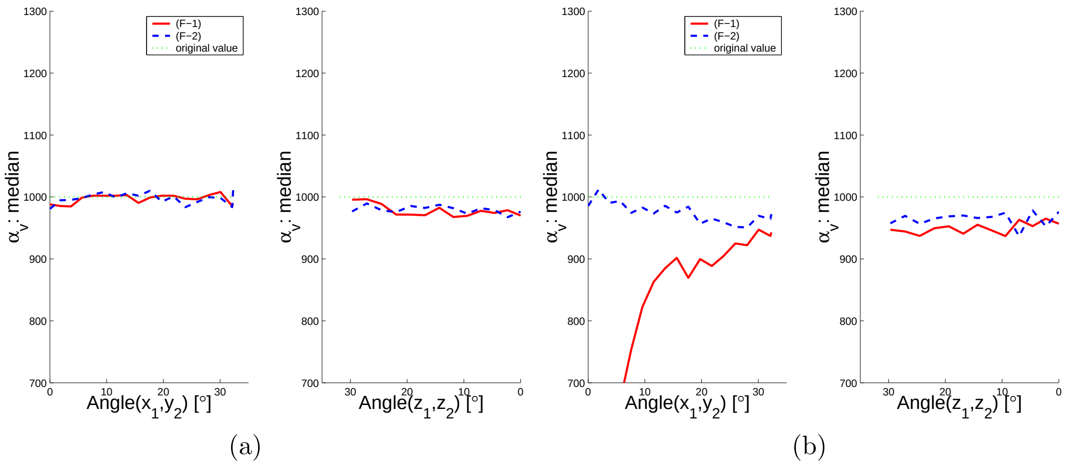
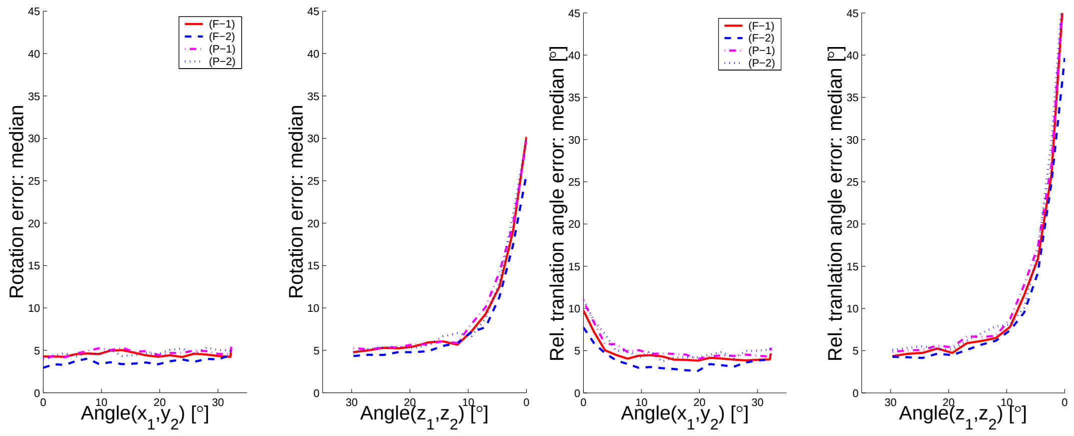


Figure 7.7: Calibration results using the factorization method for the estimation of focal length of (a) the 1st camera and (b) the second camera, using the known right parallelepipeds angles.

Extrinsic calibration. Figure 7.8 presents the results for the estimation of the camera's rotation and translation. Extrinsic calibration was performed only when the previous internal calibration was successful. Displayed error values on recovered rotations (figure 7.8–(a)) are the angle of the relative rotation between the true R and the estimated R' (the presented values are computed as the mean rotation error for both cameras). Translation error (figure 7.8–(b)) is represented by the mean of the two angles between the true and estimated relative camera-parallelepiped position vectors. This allows to evaluate the geometry of the reconstructed cameras independently from errors caused by global scaling and errors on the rotation. For a standard deviation of $\sigma = 1$ and a minimal angle of $10°$ between the camera axes, the rotation error is never larger than $10°$, with a $5°$ median value, and the translation error is similar. Those errors quite significant, but note, that were obtained using projections of at most 12 points per image (6 per parallelepiped projection.

As expected, the rotation and translation estimation is most stable and accurate with the factorization method using two parallelepipeds. Independently from the number of calibration primitives, the factorization method gives slightly better results than the reference primitive method. However, increasing the number of scene constraints (i.e. number of calibration primitives) increases also the reprojection error. The increase of the reprojection error for the methods based on two calibration parallelepipeds can be seen in chart 7.5–(b).



(a) (b)

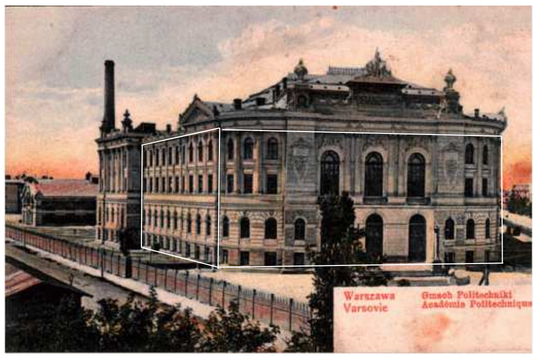
Figure 7.8: Extrinsic calibration results. (a) Median values of the rotation error; (b) Median values of the angle between the true and the estimated direction between the cameras-parallelepiped relative translation vectors. Notations are similar to figure 7.5.

7.2 Real Scenes

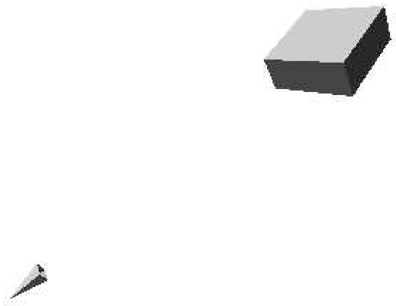
In this section, we present 3D reconstruction results of our method when applied to indoor and outdoor scenes. These examples correspond to situations where automatic methods are bound to fail: small sets of images are used and occlusions occur frequently. Each reconstruction was performed in two steps: first, one or more parallelepipeds were used to calibrate the intrinsic and extrinsic camera parameters; second, scene points and geometrical constraints were used for the reconstruction (see section 6). Results from both, single, and multiple images are shown.

7.2.1 Warsaw Scene

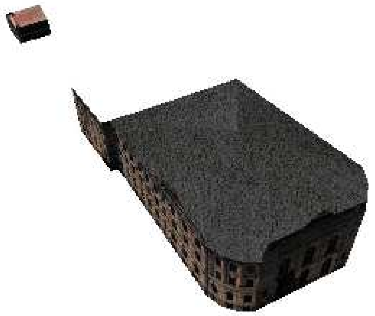
Figure 7.9–(a) shows a postcard used for the reconstruction of the main building of Warsaw University of Technology in its shape at the beginning of the century. The calibration primitive based on the main solid constituting the building is shown on the image. Note that facade of the building is not contained into the calibration primitive. Prior information used for the calibration were: right parallelepiped angles, unit ratio between two parallelepiped edges parallel to the ground plane and camera parameters except for the focal distance. Note that without information about the parallelepiped length ratio and camera aspect ratio the camera is in a singular position (compare with figure 4.1–(a)). For the reconstruction, the walls were defined as parallelograms and the additional elements of the facade were defined using symmetry and coplanarity constraints, resulting in smooth, round shape of model contours. Definition of calibration primitives took about 5 minutes. Definition of the full model took approximately 40 minutes. Figure 7.9–(b) shows the reconstructed



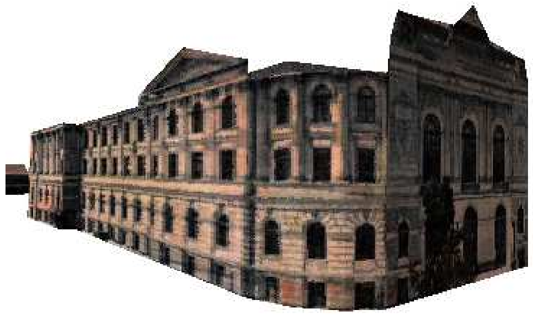
(a)



(b)



(c)



(d)

Figure 7.9: Warsaw scene: (a) The postcard used for the reconstruction; (b) Reconstruction of the calibration parallelepiped and camera poses; (c), (d) The textured model seen from different viewpoints.

parallelepiped as well as the camera pose, and figures 7.9–(c) and 7.9–(d) show the scene rendered from new viewpoints.

7.2.2 Bedroom Scene

Figure 7.10–(a) shows the original image used for the reconstruction of a bedroom scene. It was taken with a short camera focal length, therefore leading to a slight optical distortion which was not corrected here. The calibration was based on the cupboard in the central part of the image. Note that the camera was almost frontoparallel with respect to the cupboard, and thus close to a singular situation for calibration. The prior information used for calibration were: right parallelepiped angles, null camera skew parameter and principal point in the image center. The full model was reconstructed using 29 points, constrained by 2 parallelepipeds (the cupboard and the wooden belt), 3 parallelograms and 6 collinearity and coplanarity constraints. Figure 7.10–(b) shows the reconstructed parallelepipeds as well as the camera pose, and figures 7.10–(c-e) show the scene rendered from new viewpoints.

7.2.3 House Scene

Figure 7.11–(a) shows the original image used for the reconstruction of a house. The calibration was based on the parallelepiped marked in the image. The prior information used for calibration

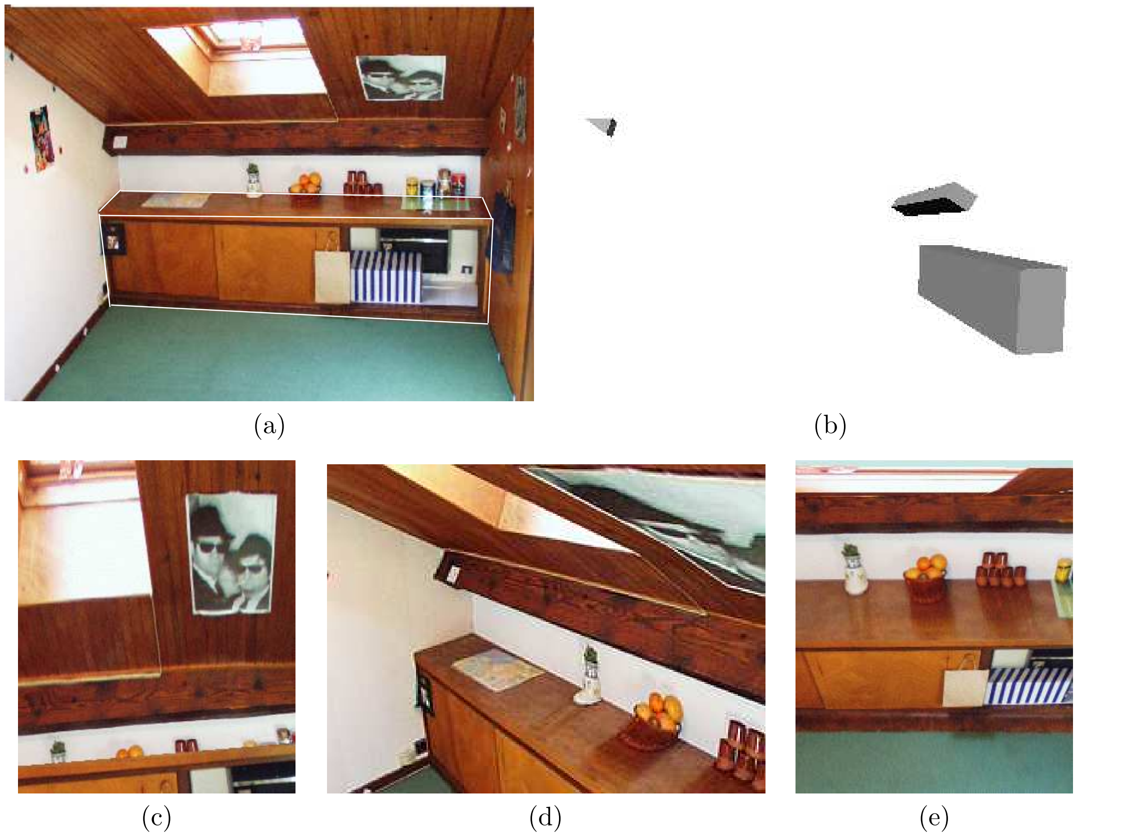


Figure 7.10: Bedroom scene: (a) The original image; (b) The parallelepipeds present in the model; (c)-(e) The textured model seen from different viewpoints.

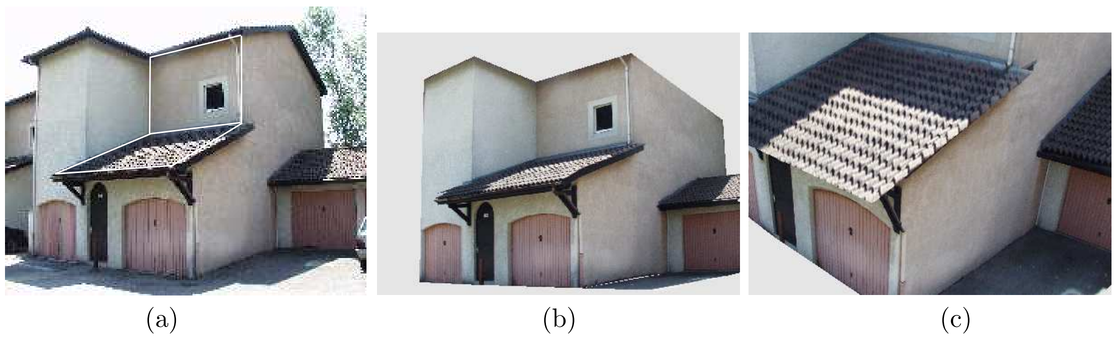


Figure 7.11: House scene: (a) The original image; (b) The textured model seen from different viewpoints.

were: two right parallelepiped angles, null camera skew parameter and principal point in the image center. The full model was reconstructed using 26 points, constrained by 1 parallelepiped, 5 parallelograms and 7 collinearity and coplanarity constraints. Figures 7.11–(b) and 7.11–(c) show the scene rendered from new viewpoints.

7.2.4 Notre Dame Square Scene

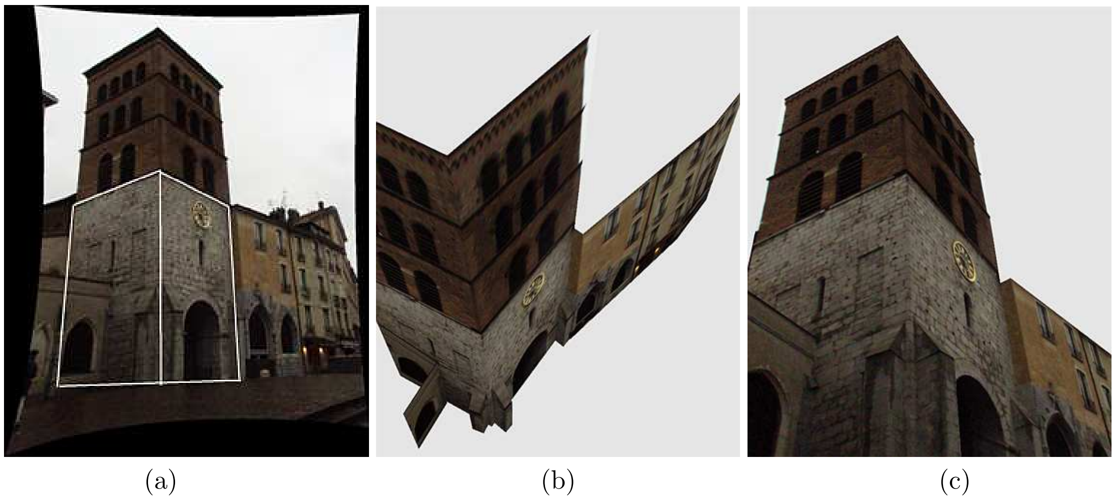


Figure 7.12: Notre-Dame square scene: (a) The original image; (b), (c) Screen-shots of the model obtained using image (a) only.

In this section, we present reconstructions of Notre-Dame square in Grenoble from single and multiple images. In both cases, radial image distortion was corrected in a preliminary step.

Reconstruction from a single image. The image and the calibration primitive for the single image reconstruction are shown in figure 7.12–(a). The prior information used for calibration were: right parallelepiped angles, null camera skew parameter and principal point in the image center. The final model is composed of 42 points, 3 parallelepipeds, 4 parallelograms and 4 lines and planes. New viewpoints of the model are shown in images 7.12–(b-c).

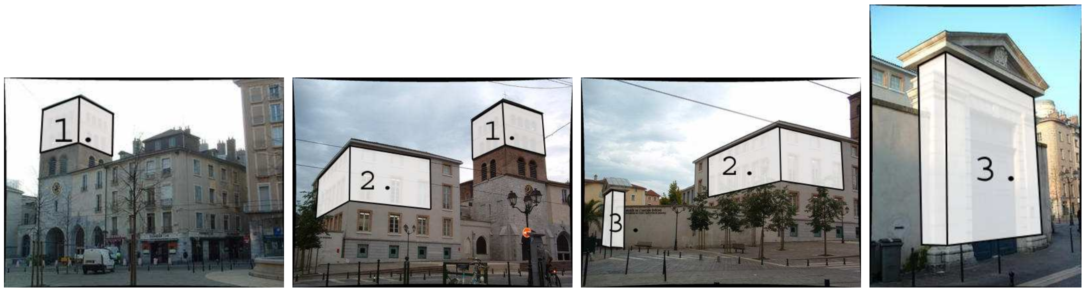


Figure 7.13: Notre-Dame square scene: 4 images from the 15 used for the reconstruction. Parallelepipeds used for the reconstruction are white painted.

Reconstruction from multiple images. The sequence used for the reconstruction is composed of 15 images whose sizes vary from 768×1024 to 960×1280 pixels. The calibration was based on 3 parallelepipeds (shown in figure 7.13). The prior information used for calibration were: right angles for parallelepipeds 1 and 2, null skew parameters, unit aspect ratios and principal points in the image center for all images. Definition of calibration primitives took about 20 minutes. Parallelepiped

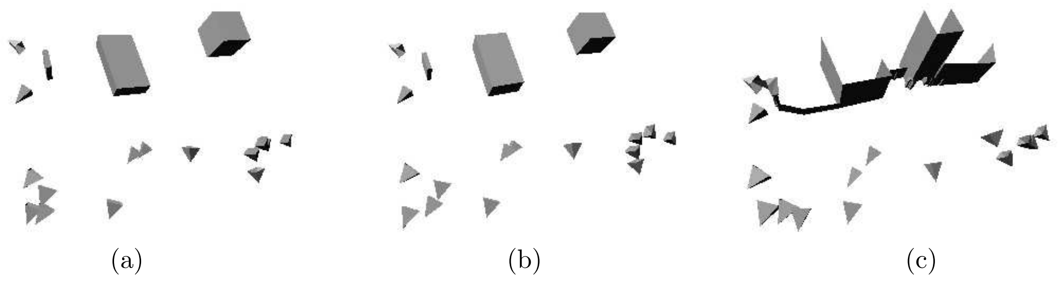


Figure 7.14: Notre-Dame square scene: reconstruction results (a) Cameras and parallelepipeds as estimated by the proposed linear factorization method; (b) Cameras and parallelepipeds parameters optimized by a non-linear method; (c) Cameras and 194 model points optimized by an unconstrained non-linear method.

3 is relatively small in the images where both parallelepipeds 2 and 3 appear. Consequently, the estimation of its vertices is unstable, and thus its parameters were not used for the calibration. The calibration was performed in two steps: first, the proposed linear factorization approach was applied, and second, the parameters obtained from the previous step were optimized. Before the second step, the mean reprojection error of parallelepipeds' vertices used for the calibration was 28 pixels. The errors occurred mainly in the images that were calibrated using parallelepiped 3. The non-linear optimization step reduced this error to 3 pixels.

After the calibration step, primitives were added and reconstructed so that the final model is composed of 194 points, 19 planes and 25 lines. Definition of the full model took approximately 4 hours. This time was dominated by input of the inter-image point correspondences. The mean reprojection error over all the model points increased up to 8 pixels. As might be expected, the most important errors always occurred in images calibrated using parallelepiped 3.

For comparison, a global bundle adjustment was performed over all the model points and the camera focal lengths. This reduced the reprojection error to 2 pixels. It did not reduce, however, the small artifacts occurring in the final model. Both non-linear methods were implemented without code optimization and derivatives were computed numerically. The parallelepiped-based optimization converged after 24 iterations and 15 seconds, and the standard point-based optimization converged after 51 iterations and 20 minutes (on a Pentium III, 733 Mhz).

The calibration primitives and the cameras reconstructed using the factorization method, the parallelepiped-based non-liner optimization and the point-based non-linear optimization are shown in figure 7.14. Synthetic views of the scene reconstructed using the parallelepiped-based calibration are shown in figure 7.15–(h-j).

7.2.5 Opposite Viewpoint Scene

Figure 7.16 shows the reconstruction of a modern building from 2 images taken from opposite viewpoints. The parallelepiped used for calibration and the estimated cameras' positions are shown in the two original images (see figures 7.16–(a-b)). In the first image, intersections of lines were computed to obtain the six points required to define a parallelepiped (see figure 7.16–(a)). The parallelepiped and the cameras reconstructed by the factorization algorithm are shown in the image 7.16–(c). New viewpoints of the whole model, composed of 32 points, 13 parallelograms and 6 planes are shown in the figures 7.16–(d-f).

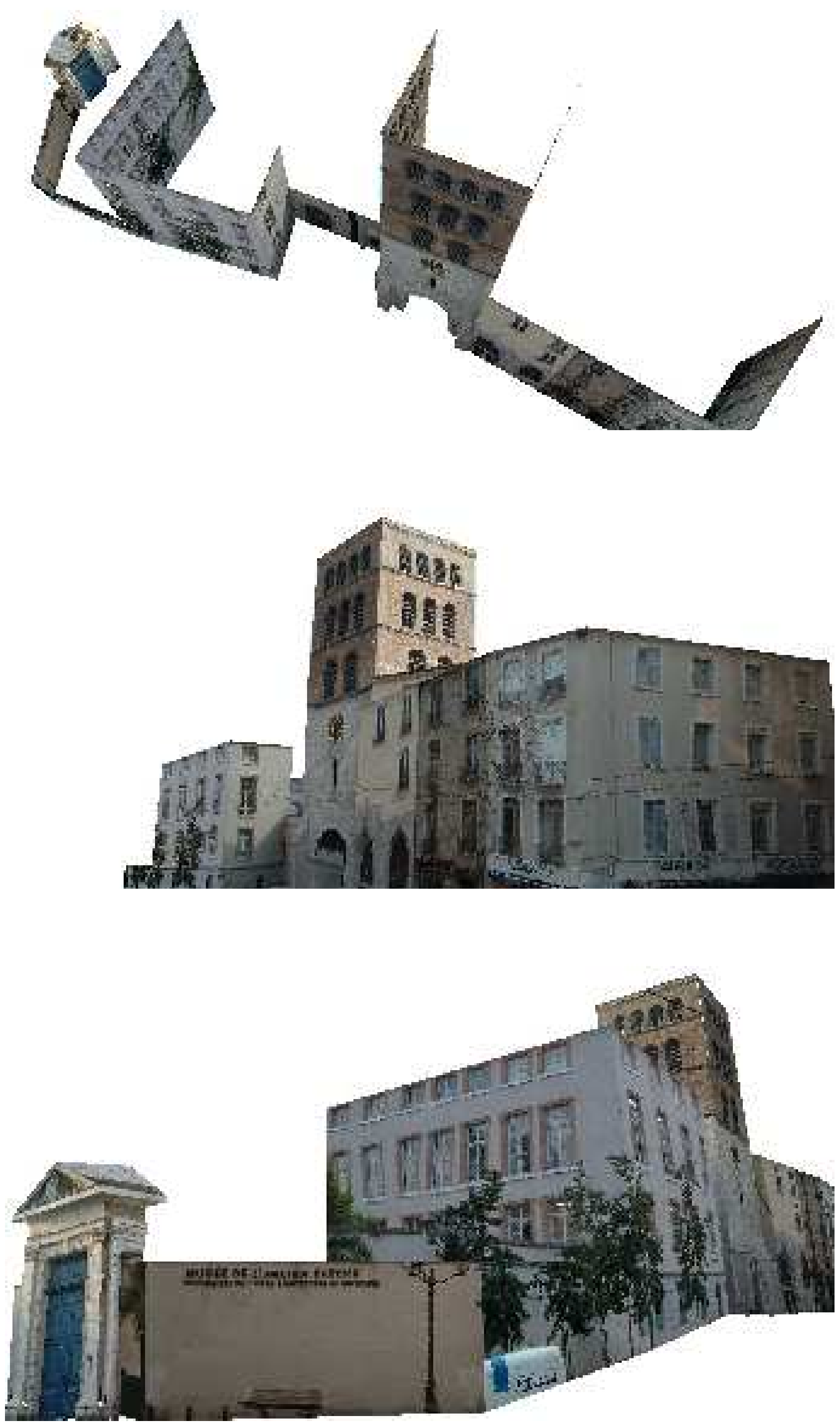


Figure 7.15: Notre-Dame square scene: New viewpoints of the textured model.

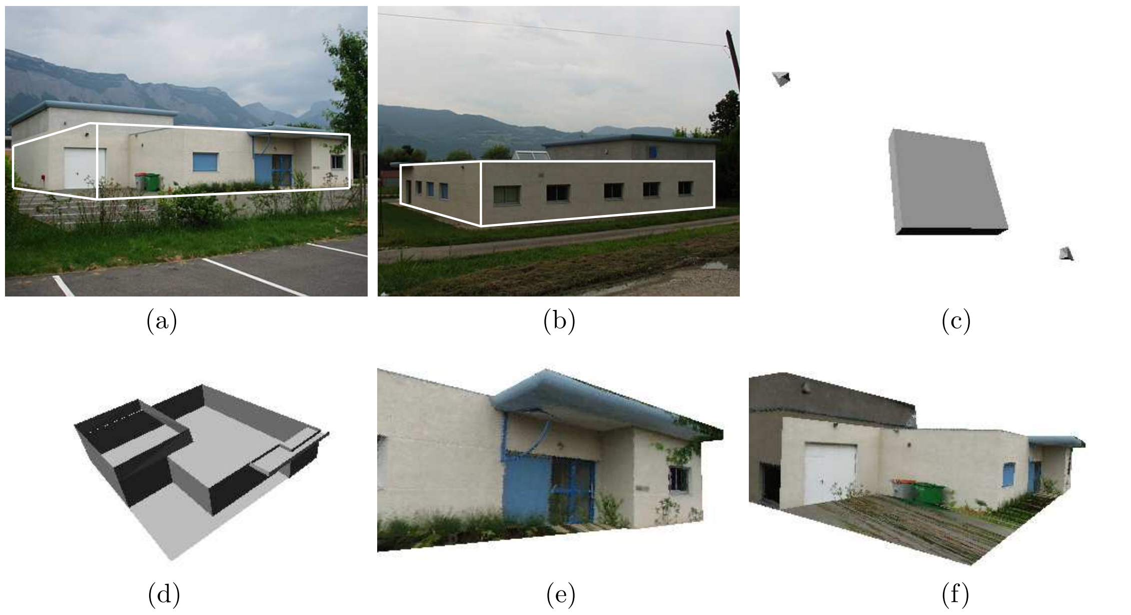


Figure 7.16: (a),(b) The original images used for the reconstruction taken from opposite viewpoints; (c) The reconstruction scenario with the computed model and the cameras' positions; (d),(e) New viewpoints of the model.

7.2.6 Chetwode Church

The Chetwode church reconstruction was based on 2 images found in the Internet and 1 calibration primitive. Both images used for the reconstruction and the calibration primitive are shown in figures 7.17–(a-b). The information used for calibration were: 3 right angles of the church tower, null camera skew parameters, unit aspect ratios and principal points in the image center. The final model consists of 34 points, 28 lines and 13 planes, constrained by 35 parallelism and 3 orthogonality constraints. The cameras and primitives reconstructed are shown in figure 7.18–(c) and a new viewpoints of the scene are shown in figures 7.18–(d) and 7.18–(e).

7.2.7 Kio Towers

The Kio towers reconstruction was based on 3 images and 2 calibration primitives. One of the images used for the reconstruction is shown in figure 7.18–(a). The information used for calibration were: 2 right angles in each tower, null camera skew parameters, unit aspect ratios and principal points in the image center. The cameras and primitives reconstructed are shown in figure 7.18–(b) and a new viewpoint of the scene is shown in figure 7.18–(c).

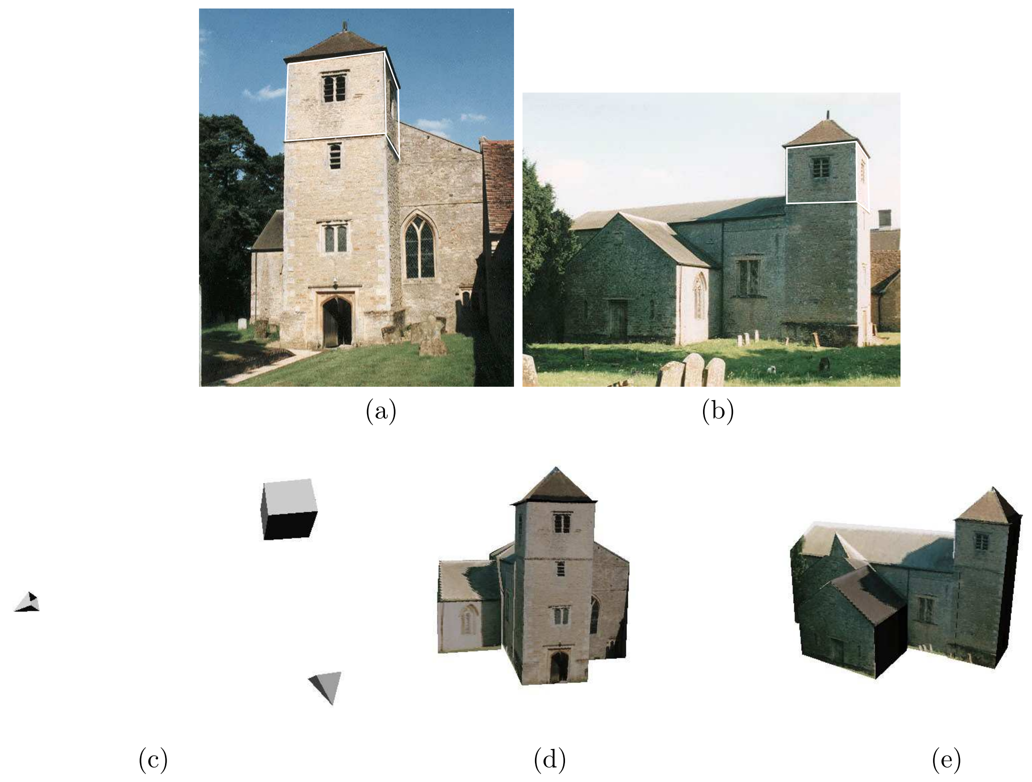


Figure 7.17: Chetwode church: (a),(b) The original images (courtesy of Mr. Kevin Quick); (c) Reconstructed calibration primitive and cameras; (d),(e) Screen-shots of the reconstructed model.

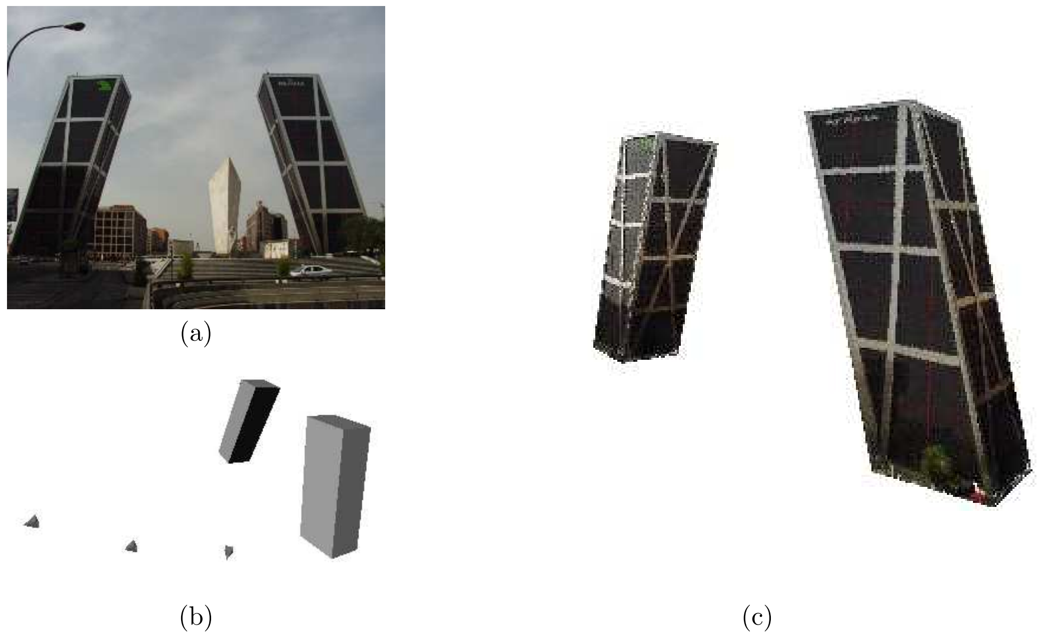


Figure 7.18: Kio towers in Madrid: (a) The original image; (b), (c) Screen-shots of the reconstructed model.

Part III

Towards a Minimal Model Parameterization

Introduction

In this part of the thesis we propose an approach for incorporation of a large set of various geometrical constraints into an image-based 3D model acquisition system. User-defined primitives, such as points, lines or planes and geometrical dependencies between them, such as parallelism, orthogonality, or distance constraints are used to build a model satisfying all the constraints, conforming to the image information (by minimization of the reprojection error).

The related work in Computer Vision is overviewed in section 2.2.1.3. An important feature differing our approach from most of the existing systems is the fact that it is able to deal with various types of objects and constraints. In classical computer vision systems, this complexity is usually reduced by considering only a limited set of objects and constraints, such as exclusively collinearity or coplanarity constraints. Indeed, in sufficiently restricted contexts, the constraint satisfaction problem can be solved using numerical, symbolic or ad-hoc techniques. In a general case however, the geometrical problem in 3D is translated into a large system of linear, bilinear, or quadratic equations that are difficult to solve, which makes classical approaches unfeasible.

The Computer-Aided Design community has designed numerous algorithms for solving geometric constraints. These methods are usually based on an analysis of the inter-object relations, a structural analysis of the corresponding equation system, the application of basic rules of euclidean geometry, local propagation mechanisms or the assembly of rigid subparts in the system [Kwaiter and Gaildrat, 1998]. However, in Computer Vision, an important part of the constraints are projections of model features in images. The projections could constitute an additional set of geometrical constraints that are imposed in the same way as constraints between the model elements. However, due to the image noise, the projection information is usually less reliable that prior information on the model. Thus, giving the same priority to all these constraints may lead to important reconstruction errors. Moreover, typically to compensate the low reliability of projection contraints, most model features are projected onto several images. This means that the model is highly overconstrained. On the contrary, when the projections are not considered, the introduced geometrical constraints do not fully fix all the degrees of freedom of the model and the corresponding system of equations is under-constrained. Unfortunately, there are no standard methods to treat under- or over-constrained systems.

We propose an approach to adapt algorithms developed by the Constraint Programming community to Computer Vision. Our reconstruction system transforms the constrained model (defined by a set of variables and constraints) into a reduced parametric model defined by a set of free variables, called *input parameters* in this thesis. The input parameters, combined with the geometric constraints, fully fix all the degrees of freedom and completely describe the model.

Our reconstruction system makes use of a dictionary of so-called *r-methods*, based on theorems of geometry. R-methods are hard-coded procedures that can solve a subset of geometric constraints exactly and in a very efficient way[2]. Graph-based algorithms are used to find a set of input

[2]In addition, the clear geometrical interpretation of r-methods allows us to control singular configurations (such as a plane defined by three collinear points) and to favor solving routines that are well-conditioned numerically.

parameters in the scene, and to decompose the constraint system into a sequence of r-methods, called *plan*. When a value is given to the input parameters, a plan execution gives a finite set of solutions for the rest of the system satisfying the imposed constraints. In order to (bundle) adjust the model to the images, the model is refined using a standard model optimization applied only over the values of the input parameters. At each iteration of an optimization step, the r-methods in the computed sequence are executed to produce a model that satisfies all the constraints.

The constraint satisfaction is based on an algorithm called *"General Propagation of Degrees of Freedom"* (GPDOF), derived originally from local propagation methods. GPDOF has several advantages. First, GPDOF can produce a set of input parameters and a sequence of r-methods in polynomial-time. Second, we should highlight that, provided that the system contains no redundant equation, GPDOF can always compute a sequence of r-methods if such a sequence exists. Finally, using r-methods corresponding to "ruler and compass" rules or theorems of geometry allows GPDOF to handle a system of geometric constraints while working at the equational level with a simple and general scheme.

Our system has been validated on two architectural models including several hundreds of linear, bilinear and quadratic constraints. In both models, the computation of the plan requires a few seconds while the optimization takes a few minutes. The obtained models are geometrically and visually correct, and fit well the images.

Outline

Chapter 8 starts by an overview of our method, illustrated by an example. Then details on objects and constraints modelling, as well as the background necessary to understand the constraint satisfaction process are given. Chapter 9 details various methods used during the constraint satisfaction process, such as the automatic r-method addition phase and the GPDOF algorithm. The chapter is completed by a discussion on the difficulties linked to constraint satisfaction, that is, the cases of singularity and the consequences of redundant constraints. The optimization phase is described in chapter 10. Chapter 11 shows the experiments we have performed on two models. Chapter 12 compares GPDOF to other equation system decomposition algorithms, some of them being developed by the Computer Aided Design Community. Chapter 13 concludes this thesis.

Chapter 8

Method Overview and Background

Many of the concepts used in this part of the thesis were developed within the Constraint Programming community and might be unknown to most of our readers. In this chapter, we give a brief outline of our approach as well as the necessary background to understand the algorithms presented in the following chapters. We start by a short overview of the different steps in our approach for the model acquisition process in section 8.1. The proposed method is illustrated by an example in section 8.1.1. Section 8.2 details the modelling of different structures used in the system. A scene is described by a complex system of relations between objects at the same description level (e.g. coordinates-variables, objects-constraints) as well as objects at different description levels (e.g. coordinate-object-scene). Thus a good insight of the used representation is necessary to understand the algorithms described in the following chapters.

8.1 Overview of the Approach

Our model acquisition approach makes use of geometric constraints. It is divided into three main phases: initialization, constraint planning and optimization.

Initialization

In the current implementation the model is defined by *points*, *lines* and *planes*. They are subject to linear, bilinear and quadratic constraints such as *distance*, *incidence*, *parallelism* and *orthogonality*.

Any other constraint which can be expressed via equations in terms of object coordinates, like angles, distance and angle ratios can be incorporated.

The model is initialized using methods described in part II. However, any other approach for calibration and reconstruction could be used here. *After this phase, we should highlight that all the variables (camera and model parameters) have an initial value.*

Constraint Planning

The goal of the constraint planning phase is to transform a model defined by constraints among objects into a parametric model. Thus, the output of this step is composed of:

- *Input parameters*: a set of variables whose values, together with the inter-object constraints, define the shape of the whole model. Input parameters are a subset of the scene object coordinates. Ideally, the number of input parameters equals to the number of degrees of freedom of the model $n_m = n_o - n_c$, where n_o is the sum of degrees of freedom of the model objects and n_c is sum of degrees of freedom of the model constraints (i.e., the number of independent equations). Such a model parameterization is called minimal.

- *Plan*: A sequence of routines (called *r-methods*) which, given values assigned to the input parameters, compute the coordinates of all the scene objects such that all the inter-object relations are satisfied.

As shown in section 9.2.3, our approach can be used to compute a minimal parameterization. However, in order to draw the greatest benefit from r-methods and reach a better performance, we are satisfied with a quasi-minimal parameterization. Note, inspite of the fact that the parameterization is not minimal, all the imposed constraints are satisfied.

The model reconstruction system we propose requires an input set of *r-methods* which allows us to decompose the whole equation system into small subsystems. An r-method [Trombettoni, 1998, 1997] is a hard-coded procedure used to solve a subset of geometric constraints. An r-method computes the coordinates of *output objects* based on the current value of *input object* coordinates with respect to the underlying constraints between input and output objects.

An example is an r-method that computes the parameters of a line based on the current position of two points incident to this line. Another example is an r-method that computes the position of some 3D point A located at known distances from three other points B, C, D. This r-method computes the (at most) two possible positions for A by intersecting the three spheres centered respectively in B, C, and D. 60 r-method patterns have been incorporated in a dictionary used by our system. They correspond to ruler-and-compass routines used in geometry or, more generally, to standard theorems of geometry.

The geometric constraint system and the corresponding algebraic equations are represented by graphs, called respectively *constraint graph* and *equation graph*. These graphs yield the dependencies between constraints and objects (constraint graph), and between equations and variables (equation graph). Based on these graphs, the constraint planning uses two algorithms:

1. *R-method addition phase:* Add *automatically* in the equation graph all the r-methods corresponding to r-method patterns present in the dictionary. For instance, the r-method pattern "line incident to two known points" may occur a lot of times in the system. Every time two points incident to a line are recognized in the constraint graph, a corresponding r-method is added to the equation graph (see example below).

   This phase thus produces an equation graph "enriched" with r-methods.

2. *Planning phase:* Perform GPDOF [Trombettoni, 1998] on the enriched equation graph. GPDOF produces:

   - a set of *input parameters*, that is, a subset of the variables describing the scene such that, when a value is given to them, there exists a finite set of solutions for the rest of the system satisfying the constraints;
   - a sequence of r-methods (called *plan*) to be executed one by one.

Model Optimization

The model is refined by an unconstrained optimization process run over the input parameters only. The optimization produces values for all the model variables such that all the constraints are satisfied (exactly) and the sum of reprojection errors is minimal. At each iteration of the optimization algorithm, the input parameter values are modified and the plan is executed, resulting in new coordinate values for all the other scene objects. The cost function is then computed as the reprojection error of all the points (and possibly lines).

8.1.1 Example of Constraint Planning

To illustrate the algorithms presented in this article, we will take a small example describing a parallelogram in 2D in terms of lines, points, incidence constraints and parallelism constraints (see figure 8.1). Of course, the scenes we handle with our tool are in 3D, and this example is just presented for didactic reasons. figure 8.1 also shows the bipartite constraint graph containing four points $\mathbf{P}_a$,...,$\mathbf{P}_d$ with coordinates $(x_{pa}, y_{pa}), ..., (x_{pd}, y_{pd})$, four lines $\mathbf{L}_a$,...,$\mathbf{L}_d$ with coordinates $(a_{la}, b_{la}, c_{la}), ..., (a_{ld}, b_{ld}, c_{ld})$, eight incidence constraints C_1,...,C_8 and two parallelism constraints C_9, C_{10}.

The equation graph corresponding to the 2D scene is shown in figure 8.2-left. The right side of figure 8.2 shows the equation graph enriched with a set of r-methods that can be used to solve subparts of the system. These r-methods belong to one of the three following categories (which could appear in a dictionary of 2D r-methods):

- line incident to two points (e.g., r-methods m_1 and m_7);
- point at the intersection of two known lines (e.g., m_2, m_4, m_6, m_8);
- line passing through a known point and parallel to another line (e.g., m_3, m_5).

The GPDOF algorithm, presented in section 9.2, works on the enriched equation graph. It is able to select for example the coordinates of $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_d$ as a set of input parameters. It also produces a plan, e.g., the sequence of r-methods (m_1, m_7, m_3, m_5, m_4), whose execution results respectively in coordinates of objects $\mathbf{L}_a$, $\mathbf{L}_d$, $\mathbf{L}_b$, $\mathbf{L}_c$, $\mathbf{P}_c$. Figure 8.3 illustrates an execution of this plan.

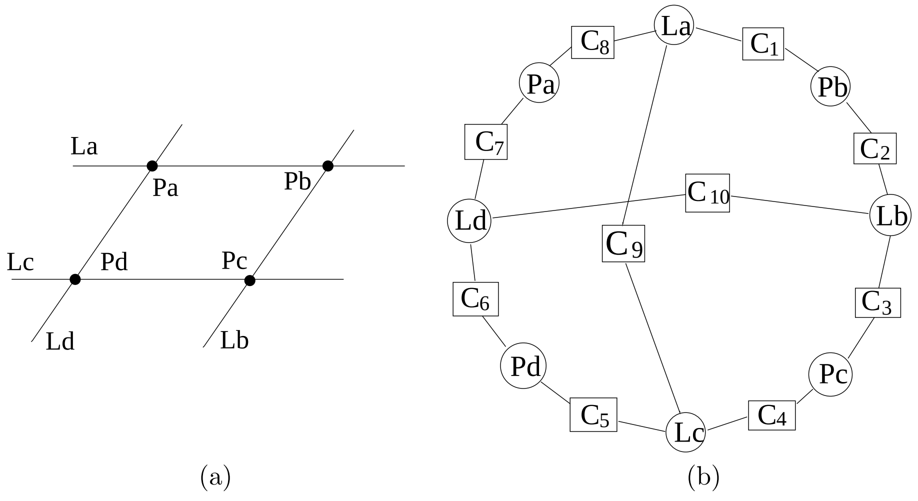


Figure 8.1: A didactic example of a 2D scene (a) and the corresponding constraint graph (b).

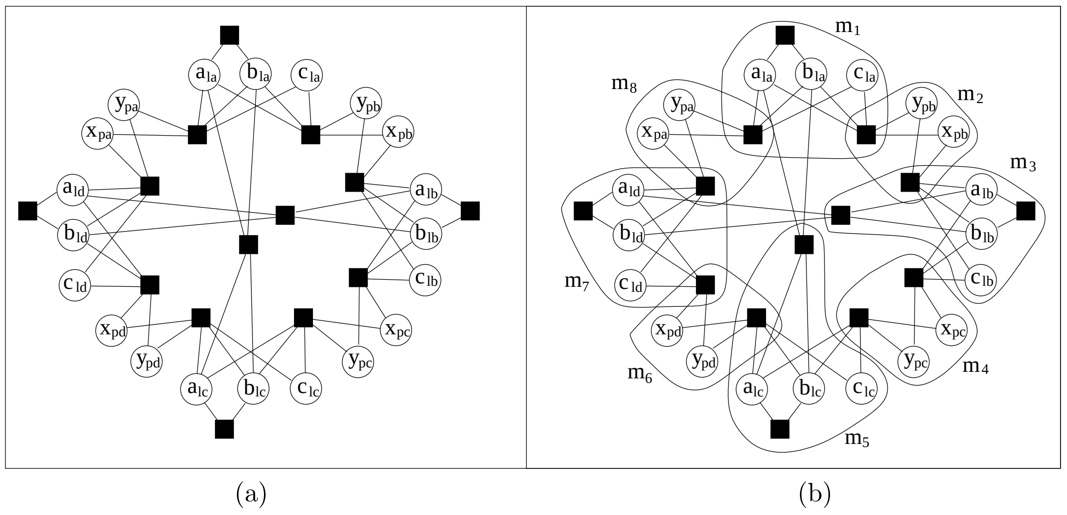

a_{la} b_{la} c_{la}
y_{pa} x_{pa} y_{pb} x_{pb}
a_{ld} b_{ld} c_{ld}
a_{lb} b_{lb} c_{lb}
x_{pd} y_{pd} x_{pc} y_{pc}
a_{lc} b_{lc} c_{lc}
(a)
m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8
(b)

Figure 8.2: (a) The equation graph of the 2D scene. Variables are represented by circles and equations are represented by black rectangles. An equation implying only the a and b coordinates of a line has the form $a^2 + b^2 = 1$ and allows a unique representation of a line. (b) The enriched equation graph. An r-method is represented by a hyper-arc including its equations and its output variables. Only eight of the sixteen possible r-methods are depicted for the sake of clarity.

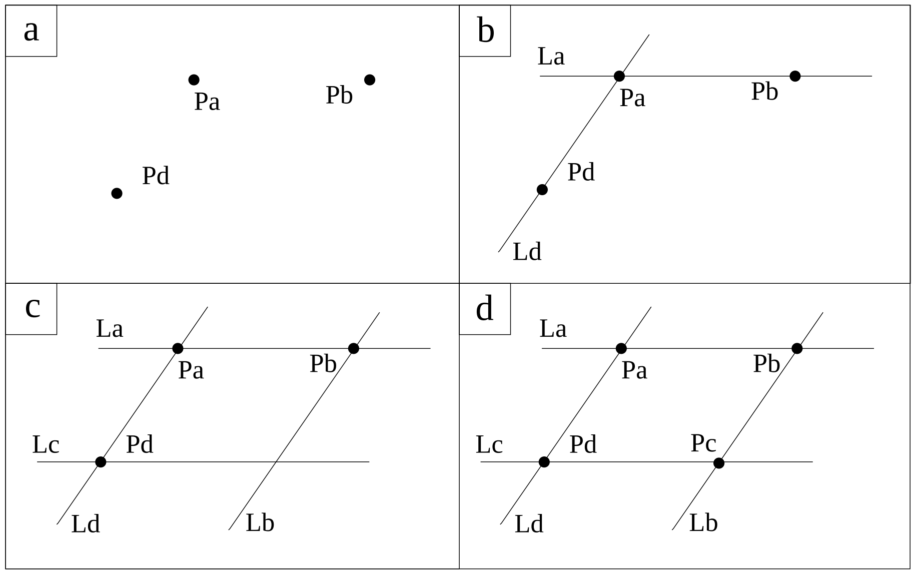


Figure 8.3: Execution of r-methods in the plan (m_1, m_7, m_3, m_5, m_4). (a) The input parameters (i.e., coordinates of $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_d$) are replaced by their current value. (b) m_1 and m_7 place lines $\mathbf{L}_a$ and $\mathbf{L}_d$ resp. (c) m_3 and m_5 place $\mathbf{L}_b$ and $\mathbf{L}_c$ resp. (d) Finally, m_4 places point $\mathbf{P}_c$.

8.2 Scene Modeling and Background

8.2.1 Geometric Objects

In the current implementation of the system, available objects are points, lines and planes. Their representation has a significant impact on the performance of the system. It is necessary to use a representation corresponding to the number of degrees of freedom characterizing the objects and allowing for an efficient computation. Section 1.4.1.1 gives algebraic representation of objects. We briefly detail how the objects are represented in the system.

Points (see figure 8.4–(a)). the 3 degrees of freedom (dof) of a point are represented by 3 variables x, y, z.

Lines (see figure 8.4–(b)). the 4 degrees of freedom of a line are represented in Plücker coordinates by 6 variables and 2 internal relations. The line coordinates are related by two conditions: the directional vector is constrained to be of unit length (equation r_d) and all the line coordinates are constrained by Plücker condition (equation r_l).

Planes (see figure 8.4–(c)). the 3 degrees of freedom of a plane are represented by 4 variables and 1 internal relation. The plane normal vector $\mathbf{n}$ is constrained to be of unit length (equation r_d).

Note that the solving process described in this article can also support other parameterizations and other types of primitives. Also note that the final model is represented by a set of points, i.e., lines and planes are only used to define constraints between points.

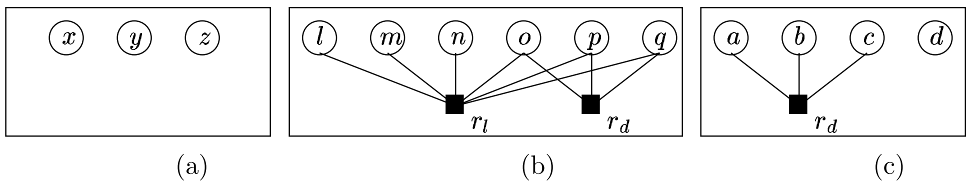


Figure 8.4: Representation of points, lines and planes. Circles correspond to variables (coordinates) and rectangles correspond to internal equations constraining the variables.

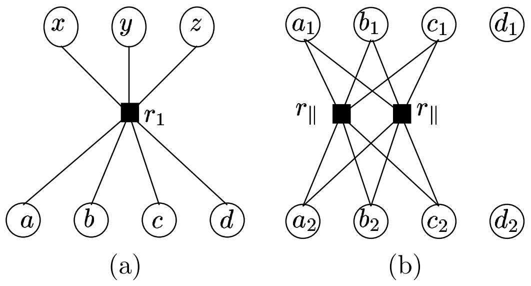


Figure 8.5: Representation of dependencies between object coordinates induced by constraints. (a) a point-plane incidence constraint (1 dof) is represented by the equation: $ax+by+cz+d = 0$ relating all the variables of both objects. (b) a plane-plane parallelism constraint (2 dofs) is represented by equation $(a_1, b_1, c_1) \sim (a_2, b_2, c_2)$ relating only the variables representing both plane normals.

8.2.2 Geometric Constraints

In the current implementation, we consider the following constraints:

- *distance:* point-point, point-line, point-plane;
- *incidence:* point-line, point-plane, line-plane;
- *parallelism:* line-line, line-plane, plane-plane;
- *orthogonality:* line-line, line-plane, plane-plane.

The equations engendered by these constraints are given in section 1.4.1.2. Any other constraint which can be expressed as equations in terms of objects coordinates, like angles, distance and angle ratios can be incorporated.

8.2.3 Variables and Equations

The geometric constraints in the scene induce a set of equations between the parameters of the objects. The scene can then be modelled by:

- a set V of *variables* over the reals with a current value each; the variables are the coordinates (or parameters) of geometric objects;
- a set E of *equations* generated by geometric constraints; the equations are linear or non-linear.

Examples of dependencies between the variables induced by inter-object constraints are given in figure 8.5.

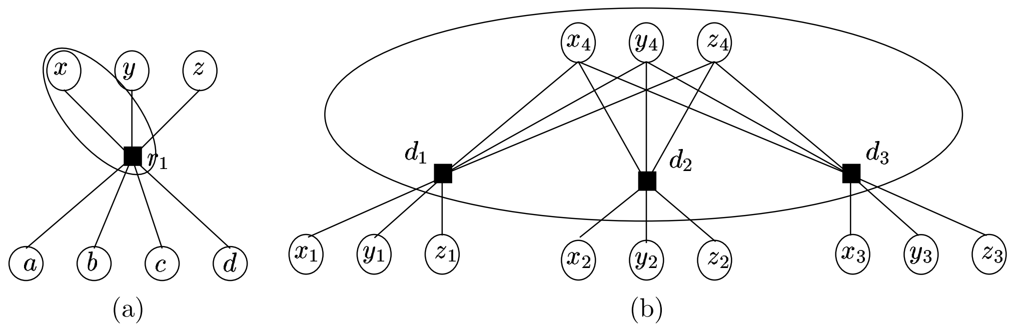


Figure 8.6: Representation of r-methods by hyper-arcs that include the satisfied equations and the output variables. (a) An r-method fixing the remaining degrees of freedom of a point incident to a plane. (b) An r-method computing the position of a point using 3 point-point distance constraints.

8.2.4 R-Methods

Our model reconstruction system is based on a dictionary containing an input set M of *r-methods*.

An r-method is a routine executed to satisfy a subset E_m of equations in E by calculating values for its *output variables* as a function of the other variables implied in the equations. Two examples of r-methods are shown in figure 8.6.

Definition 1 *An* **r-method** *m in M is a function over a set of* **input variables** *I. The variables involved in the equations $E_m \subset E$ are divided into a non-empty set of* **output variables** *$O \subset V$, and a set of input variables $I \subset V$.*

Given a set of values $\overline{I}$, the r-method m yields the set of solutions for O satisfying E_m. This operation is called **execution** *of the r-method m.*

The r-method m is **free** if no variable v in O is involved in a constraint in $E \setminus E_m$. Thus, executing a free method cannot violate other equations in $E \setminus E_m$.

Note that a given equation can generally be solved by several r-methods (e.g., 3 r-methods could be used to solve the point-plan incidence shown in figure 8.6(a)), making the problem of computing a sequence of r-methods to solve all the equations combinatorial.

The current dictionary of our system contains 60 r-methods. These r-methods use only 40 different generic execution procedures. For instance, parallelism of planes and parallelism of lines are solved by the same procedure.

The dictionary includes all the r-methods that solve constraints by computing (output) parameters of *one* object. Details on the design and implementation of r-methods are given in Appendix C. More complicated r-methods computing more than one object at a time can be envisaged. The algorithms used in our system can deal with any type of r-method, although the time complexity will grow with the number of constraints involved in r-methods (see section 11.3).

When we detail the automatic r-method addition phase, we will often talk about *r-method patterns* present in the dictionary. An **r-method pattern** of an r-method m is a generic constraint graph corresponding to the equations solved by m. We design this constraint graph by a pattern because a similar constraint graph (pattern) may occur several times in the actual constraint graph corresponding to the model, thus leading to the creation of several similar r-methods. For instance, the r-method pattern "line incident to two known points" is a graph made of 4 vertices (3 objects and 1 constraint). Every time two points incident to a line are recognized in the constraint graph (as a subgraph), a corresponding r-method is added to the equation graph.

Finally, it is important for our system to distinguish so-called *linear r-methods* giving one solution and *non-linear r-methods* giving several solution.

Definition 2 *Let m be an r-method, E_m be the set of equations solved by m, and O (resp. I) be the set of output (resp. input) variables of m.*

The r-method m is a **linear r-method** *iff all the equations in E_m are linear in terms of variables in O (i.e., E_m in which the variables in I are replaced by a constant become linear). R-method m is a* **non-linear r-method** *iff at least one equation in E_m (in terms of O) is non-linear: m may produce several solutions.*

For instance, an r-method which computes the position of some 3D point A located at known distances from three other points B, C, D is a non-linear r-method. This r-method generally produces two possible positions for A.

An r-method which computes the parameters of a line based on the current position of two points incident to this line is a linear r-method. Even though an incidence constraint is bilinear, when the coordinates of the involved points are known, this gives linear equations relating line coordinates.

Hypotheses on r-Methods

R-methods must compute a finite set of solutions for the output variables. In other words, the dimension of the variety of the solutions is 0. Therefore r-methods have generally as many equations as output variables. This is the case for the r-methods in our dictionary.

In addition, an r-method, especially a non-linear r-method, must be able to compute *all* the solutions satisfying the involved equations. Indeed, this allows the backtracking phase described in Section 10.1 to combine the solutions computed by the different r-methods in the plan, without losing any solution (see section 10).

The remark above highlights that local numerical minimization methods cannot be used for executing an r-method because they cannot obtain *all* the solutions for the output variables.

To build fast non-linear r-method execution procedures, we made symbolic manipulations of the equations involved in r-methods as shown in Appendix C.

Interests of r-Methods

Using r-methods for decomposing the constraint system of the model has a lot of advantages:

- The code of an r-method allows a very good performance. Executions of r-methods in our dictionary run in several microseconds.

- A lot of r-methods in our dictionary are linear while the implied constraints are bilinear (e.g., incidence, parallelism). This highlights that using r-methods to decompose a system of equations is a significant way to lower the complexity of the equations and thus to improve the performance.

- The semantics behind a given r-method (i.e., the fact that it is a theorem of geometry) ensures that the implied geometric constraints can be solved and helps to detect singular configurations. On the contrary, the subsystems of equations created by pure graph-based decomposition methods, such as the maximum-matching (see chapter 12), are arbitrary and may even correspond to contradictory equations. An example is shown in [Trombettoni, 1998].

- As said above, r-methods yield all the solutions to the implied equations.

8.2.5 Graph Representation of the Model and Data Structures

The algorithms used by our system require a structural view of the entities in the scene. The geometric constraint system and the equation system are respectively represented by a *constraint graph* and an *equation graph* (see figure 8.1 and 8.2). An equation graph indicates the dependencies between equations and variables in the scene.

Definition 3 *A* **constraint graph** *is a bipartite graph where nodes are constraints and objects, represented by rectangles and circles respectively. Each constraint is connected to its objects.*
An **equation graph** *is a bipartite graph* (V, E, A) *where nodes are equations in* E *and variables in* V*, represented by rectangles and circles respectively. Each equation is connected to its variables by an edge in* A*.*
An **enriched equation graph** (V, E, A, M) *is an equation graph* (V, E, A) *enriched with a set* M *of r-methods.*

Our system is implemented in the `C++` programming language. The different entities (constraints, geometric objects, equations, variables and r-methods) are represented by structured objects. Several fields have been added to allow a direct access to the entities. For example, for a given variable v, we can know in constant time the set of equations involving v, in which r-method v is an output variable, to which geometric object v belongs, and so on. In this implementation, the constraint graph, the equation graph and the enriched equation graph share the same data structures.

The dictionary of r-methods is implemented as a hash table in order to make the automatic r-method addition phase quicker. Details about this hash table are given in section 9.1.

The next chapter details the algorithms necessary for the constraint planning: adding automatically r-methods in the equation graph based on the dictionary, computing a set of input parameters and a sequence of r-methods, based on the enriched equation graph.

Chapter 9

Constraint Solving

This chapter presents the algorithms used to satisfy the geometrical constraints imposed on the model. These algorithms work on constraint and equation graphs introduced in the previous chapter. First, the constraint graph is used to find object-constraint configurations which can be solved by r-methods contained in the r-method dictionary. Then the matched r-methods are superimposed on the equation graph. This process is described in section 9.1. Subsequently, the `GPDOF` algorithm, outlined in section 9.2 is run to find a set of input parameters and a sequence of r-methods describing the model. As the constraints used in the system are user provided, it is possible that the input is incorrect, e.g contains redundant constraints or singular object configurations. Using the geometrical analysis of the scene via r-methods it is possible to overcome these problems. Approaches implemented in our system are described in section 9.3.

9.1 Automatic R-Method Addition Phase

This phase consists in enriching the equation graph with r-methods found in the dictionary. It considers as input:

- the constraint graph corresponding to the scene;
- the dictionary of r-method patterns.

This phase works on the constraint graph. It performs a subgraph matching between subgraphs (made of constraints and objects) in the constraint graph and r-method patterns present in the dictionary. More precisely, we handle a *subgraph isomorphism* problem. All the connected subgraphs of the constraint graph with a "small" size are explored. When a subgraph corresponds to an entry in the dictionary, the corresponding r-methods are added to the equation graph.

For instance on the 2D scene, a certain iteration of the r-method addition phase considers the subgraph made of nodes $\mathbf{P}_a$, C_8, $\mathbf{L}_a$, C_1, $\mathbf{P}_b$ (see figure 8.1). A corresponding subgraph pattern (i.e, 2 points incident to a same line) is found in the dictionary, so that the r-method m_1 (i.e., line L_a passing through two known points P_a and P_b) is created and added to the equation graph.

The algorithm explores all the connected subgraphs of size less than a small value k (see next paragraph). For every found subgraph, the procedure `Subgraph_recognition` compares it with the subgraph patterns in our dictionary. If the subgraph matches, the corresponding r-methods are added to the equation graph[1].

9.1.1 Exploring all Connected Subgraphs of Size at Most k

```
algorithm All_connected (S: set of nodes; d: current depth; k: max size; G: constraint graph):
    Subgraph_recognition ($S$)
    if $d < k$ then
        $N' \leftarrow$ Selected_neighbors ($S$, $d$, $k$, $G$)
        for every neighbor $n$ in $N'$ do
            All_connected ($S \cup \{n\}$, $d + 1$, $k$, $G$)
        end
    end
end.
```

The value k is the maximum number of nodes (objects+constraints) implied in any r-method of the dictionary (e.g., 7 in our system). Starting from a single node (S is a singleton), the subgraphs are built by incrementally adding a neighbor node to the current connected subgraph S until the size k is reached. This depth-first search algorithm detailed in Algorithm `All_connected` is a simplification of the algorithmic scheme presented in [Avis and Fukuda, 1996].

Note that a given subgraph of size l can be built from l different subgraphs of size $l - 1$. Thus, the key idea allowing the algorithm to explore a *tree* of subgraphs (and hence to explore a given subgraph only once instead of l times) is to consider at each step only a specific subset N' of selected neighbors: The property taken into account by the function `Selected_neighbors` is based on a unique numbering of the nodes [Avis and Fukuda, 1996]. The property states: a subset $S \cup \{n\}$ is a "son" of S iff n has the smallest number among the objects in $S \cup \{n\}$ such that S is connected. Thus, if the property holds for $S \cup \{n\}$ then $S \cup \{n\}$ will be considered; otherwise this subgraph is discarded because handled at another iteration.

[1] Remember that several r-methods may exist for the same set of constraints.

The time complexity of this algorithm[2] is $O(N \times a \times k^4)$, where N is the actual number of connected subgraphs of size k or less (N is $O(n^k)$) and a is the maximum degree of nodes in the graph.

9.1.2 Subgraph Recognition

The function `Subgraph_recognition` compares every subgraph S found in the constraint graph with the subgraph patterns in our dictionary. However, the problem of deciding whether two graphs are *isomorphic* is still an open problem for which no polynomial algorithm is known [Papadimitriou, 1994]. In order to quickly know whether a subgraph S is isomorphic with a subgraph S' in the dictionary, we proceed as follows:

1. We first compute a string e corresponding to the number of nodes in S in every category: the number of points in S, its number of lines, number of planes, number of point-line incidence constraints, and so on. The dictionary is implemented as a hash table, and theses types of strings represent hash functions. If the hash function e corresponds to no entry in the dictionary, the second step below must not be performed and no r-method will be created based on the subgraph S.

   Otherwise, this means that an entry in the hash table contains one set ES' of subgraphs (having the same number of nodes in every category). The step 2 below checks whether one subgraph pattern $S' \in ES'$ is isomorphic with S.

2. To know whether two graphs S and S' (with the same number of nodes in every category) are isomorphic, we use a combinatorial process inspired by the solving process of Constraint Satisfaction Problems (chronological backtracking). In short, objects in the subgraph S are reordered to be matched with objects in the pattern S'. Two objects at the same rank in the order must have the same type and also the same types of constraints with objects placed before.

3. If S and S' match, then the r-methods associated to S' are added to the equation graph.

In our dictionary, the 60 r-method patterns are generated by 45 different subgraph patterns. The hash table contains 45 entries, which means that all the subgraph patterns are discriminated by their number of nodes in every category (i.e., the size of ES' is always 1 in our current version). The example in figure 9.1 highlights why the combinatorial process (step 2) remains necessary.

9.1.3 Practical Time Complexity

In practice, as detailed in chapter 11, the time complexity of the r-method addition phase is negligible (one or two seconds for our models). Two reasons explain this good behavior. First, in our current dictionary, the subgraph patterns are small, so that the size k is small. Second, constraint graphs corresponding to scenes are rather sparse (the number of constraints is close to the number of objects).

The situation would worsen if the designer wanted to add in the dictionary more complicated r-methods, especially r-methods with more than one object as output. In this case, we think that more sophisticated subgraph isomorphism algorithms should be envisaged [Ullman, 1976; Régin, 1995; Sorlin and Solnon, 2004].

[2]The call to `Subgraph_recognition` is not taken into account.

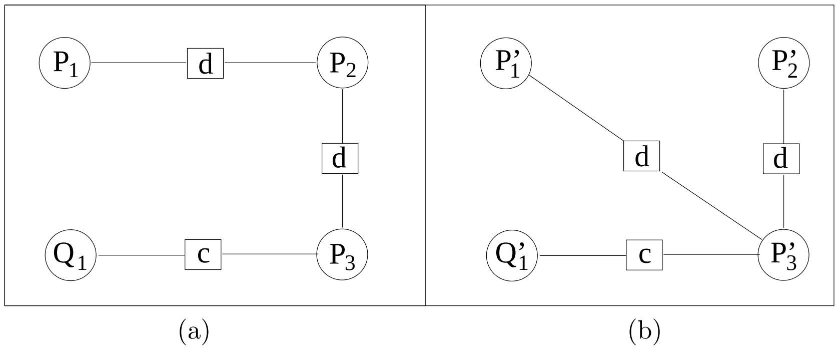


Figure 9.1: Two subgraphs with the same entry in the hash table, i.e. characterized by the same number of points, planes, distance constraints and incidences. (a) a “bad” subgraph found in the constraint graph with no corresponding r-method; (b) a subgraph pattern in our dictionary made of 3 points P_1', P_2', P_3', a plane Q_1', two distance constraints and one incidence.

9.2 Computing a Plan and a Set of Input Parameters

These computations are obtained by the `GPDOF` algorithm [Trombettoni, 1998]. `GPDOF`[3] computes a sequence of r-methods to be executed for satisfying all the equations (the plan). `GPDOF` solves this combinatorial problem in polynomial-time. The main advantages of `GPDOF` are the following:

- `GPDOF` is very fast (quasi-linear in practice).
- `GPDOF` can find a sequence of r-methods if such a plan exists.
- A set of input parameters can be immediately deduced from the plan. Thus, `GPDOF` is also a procedure to determine a set of input parameters in polynomial time.

These attractive properties come under the assumption that the constraint system contains no redundant constraints, that is, the system must include only independent equations. Section 9.3 details this point and explains the first procedures used by our tool for removing redundant constraints before the use of `GPDOF`.

Chapter 12 highlights that `GPDOF` compares favorably with local propagation solvers and geometric solvers for decomposing equation systems or geometric constraints.

9.2.1 Description of GPDOF

`GPDOF` [Trombettoni, 1998] is a generalization of a local propagation algorithm used to solve multiway dataflow constraints [Sutherland, 1963]. It works on an enriched equation graph (see section 9.1). `GPDOF` runs the three following steps until no more equation remains in the equation graph G (success) or no more free r-method is available (failure):

1. select a **free** r-method [4] m,
2. remove from G the equations and the output variables of m,
3. create all the *submethods* of an r-method m_i that share equations or output variables with m (see section 9.2.2).

[3]`GPDOF` stands for General Propagation of Degrees of Freedom.
[4]Recall that output variables of a *free r-method* appear in no “external” equations (see Definition 1).

A plan can be obtained by reversing the selection order: the first selected r-method will be executed last. The work of GPDOF is illustrated in figure 9.2.

The first two steps above define the standard PDOF algorithm [Sutherland, 1963] on which GPDOF is based (PDOF accepts only r-methods solving one equation). Iteratively selecting free r-methods ensures that no loop is created in the plan.

It appears that, when r-methods can solve several equations, there is no guarantee that the standard PDOF finds a plan, even if one exists. This highlights the importance of *submethod* appearing in the third step above which guarantees that GPDOF can find a plan if one exists. The following section details this important notion.

GPDOF may fail when it is not able to remove some equations because no more free r-method is available. In this case, it is ensured that no complete plan can be computed (with the r-method patterns defined in the dictionary). One obtains an incomplete plan which solves only a subset of the equations (i.e., the equations removed by step 2 of GPDOF) and which contains thus more input parameters. This incomplete plan can nevertheless be used during the model optimization. Of course, the constraints corresponding to the equations that have not been removed from the equation graph by GPDOF will not be satisfied.

9.2.2 Overlap of R-Methods and Submethods

Theoretically, the case may occur that any possible plan contains at least two r-methods which *overlap*, that is, two r-methods which share the same constraints or output variables[5]. In other words, if the selection of overlapping r-methods was forbidden, then we could not ensure that an existing plan would be found. A plan that contains overlapping r-methods means that some constraints will be solved several times during one plan execution.

As shown in the example (see figure 9.3), the notion of *submethod* and step 3 of GPDOF have been introduced in GPDOF precisely in order to make it able to select r-methods which overlap. Informally, a submethod m'_i of r-method m_i is created when a free r-method m is removed in step 2, which removes some equations and/or output variables from m_i. For example, r-methods m'_5, m'_7 are the submethods of resp. m_5, m_7 due to the selection of m_6 and the removal of equations solved by m_6 (black rectangles). [Trombettoni, 1998, 1997] detail how submethods are precisely constructed and why submethods always remain available for a future selection.

It is important to understand that the notion of submethod is only used during the execution of GPDOF whose work is purely structural. This notion is then forgotten, so that the obtained plan contains no submethod. Indeed, every time GPDOF selects a submethod, the corresponding r-method is added to the plan.

The example also shows that, due to the selection of overlapping r-methods, some constraints are solved several times by different r-methods in the plan. For instance the two r-methods m_2 and m_3 belong to the same plan so that the incidence constraint between point P_b and line L_b will be solved twice.

Solving several times a same equation has no significant impact on the plan execution. Especially, all the constraints are solved at the end. The only drawback is that a plan with overlapping r-methods contains generally more r-methods (e.g., the plan shown in figure 9.2 contains 5 r-methods, while the plan in figure 9.3 contains 7 r-methods). As a result, we could expect a loss in performance.

However, r-methods are executed in microseconds (see chapter 11). Moreover, the phenomenon of selecting overlapping r-methods can be easily limited by heuristics: when GPDOF can choose several free r-methods at a given iteration (step 1), GPDOF selects one r-method which is not a submethod (if any).

[5]An example is shown in [Trombettoni, 1998]. The case does occur in real size applications.

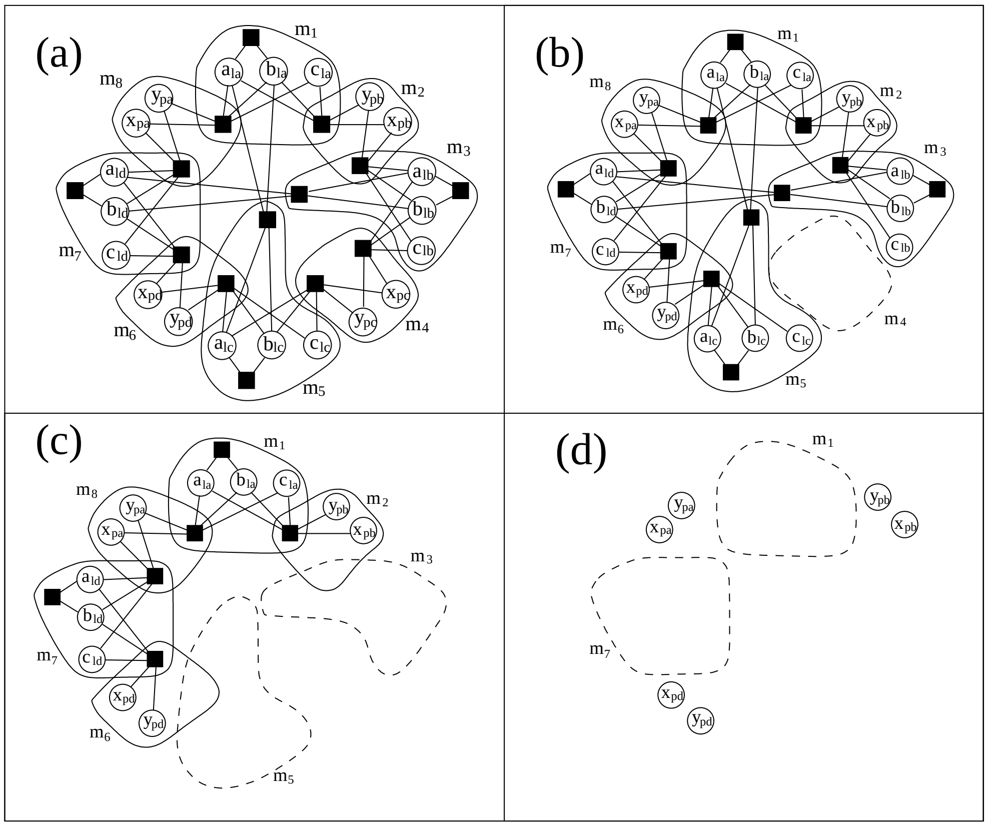


Figure 9.2: A constraint planning phase performed by GPDOF on the 2D scene. (a) At the beginning, r-methods m_2, m_4, m_6, m_8 are free, so that one of them is selected, e.g., m_4. (b) This selection implies the removal of the equations and the output variables of m_4 from the equation graph. (c) This frees r-methods m_3 and m_5 which are selected and removed next in any order. (d) The r-methods m_1 and m_7 become then free and can be selected. The process ends since no more constraint remains in the equation graph. The obtained plan is the sequence (m_1, m_7, m_3, m_5, m_4) and the input parameters are the remaining variables.

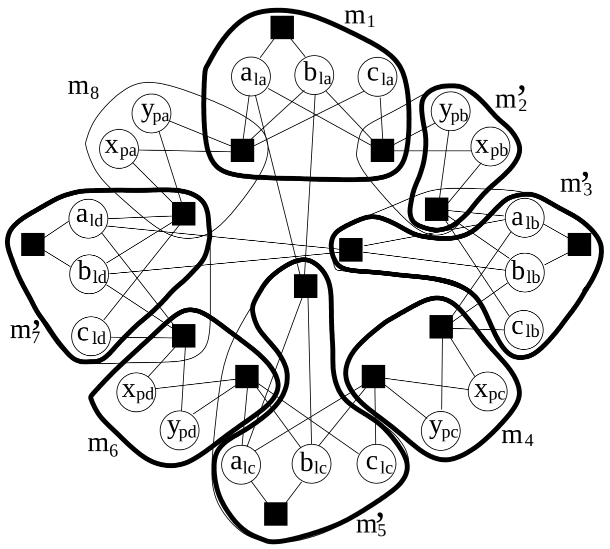

m1
m8
m'2
m'3
m'7
m6
m4
m'5
a_{la} b_{la} c_{la}
y_{pa} x_{pa}
y_{pb} x_{pb}
a_{ld} b_{ld} c_{ld}
a_{lb} b_{lb} c_{lb}
x_{pd} y_{pd}
x_{pc} y_{pc}
a_{lc} b_{lc} c_{lc}

Figure 9.3: GPDOF may first select m_6 which is free. Step 3 of GPDOF then creates the submethod m'_5 of m_5 and the submethod m'_7 of m_7. The process continues and selects m_4, m'_5, m_1 (creation of submethod m'_2), m'_2 (creation of submethod m'_3), m'_3, and finally m'_7. The obtained plan is the sequence (m_7, m_3, m_2, m_1, m_5, m_4, m_6). Selected r-methods (m_1, m_4, m_6) and submethods (m'_2, m'_3, m'_5, m'_7) are represented by thick hyper-arcs.

Properties of GPDOF

In [Trombettoni, 1998], and due to the notion of submethod, it is proven that GPDOF guarantees to compute a sequence of r-methods, if one such sequence exists.

In addition, GPDOF solves this combinatorial problem in polynomial time. Its worst-case time complexity is $O(n \times dc \times dv \times m \times (g \times dc + g^2))$ (see [Trombettoni, 1998]), where n is the number of equations, m is the maximum number of r-methods per equation, dc and dv are the maximum degrees of respectively equations and variables in the equation graph, and g is the maximum number of equations and output variables involved in an r-method. This complexity is a polynomial function of the input parameters: $dv \leq n$; $g \leq n$; $dc \leq |V|$; $m \leq |M|$.

GPDOF runs in a few seconds on our two models with several hundreds equations, showing that it should be acceptable for scene modelling applications.

9.2.3 Computing the Input Parameters

The values of the input parameters must be known before the plan is executed. This is the case in our scene modeling system since all the variables have an initial value after the initialization phase (see Section 8.1). Every time new values are computed for the input parameters by the numerical algorithm in the optimization process, the plan is executed.

Since GPDOF computes the plan in a reverse order, obtaining the input parameters is a side-effect of GPDOF.

When no r-method in the plan corresponds to a submethod selection, the input parameters simply consist of the variables which are output of none of the r-methods in the plan. This yields the 6 coordinates of points P_a, P_b, P_d for the plan illustrated in figure 9.2.

The general case is a little bit more complicated and produces two disjoint subsets of input parameters:

- The set $\mathcal{P}_1$ contains the variables which are output by no r-method in the plan (as above).
- The set $\mathcal{P}_2$ comes from the overlap phenomenon (submethods). $\mathcal{P}_2$ contains variables v such that:
  - $v \notin \mathcal{P}_1$,
  - v is an input variable of an r-method m_j in the plan,
  - v is not an output variable of any r-method m_i placed before m_j in the sequence,
  - v is an output variable of an r-method m_k placed after m_j in the sequence.

In the plan illustrated in figure 9.3, the subset $\mathcal{P}_1$ contains the 2 coordinates of point P_a. The subset $\mathcal{P}_2$ contains the parameters of points P_b, P_c, P_d and lines L_a, L_b.

The values of variables in $\mathcal{P}_1$ must be known before the plan is executed and will not be modified by this execution. On the opposite, the initial value of a variable v in $\mathcal{P}_2$ is used when an r-method m_j is executed, but this value will be modified later by another r-method m_k in the plan.

Towards a Minimal Parameterization of the Model

The example has been chosen to illustrate the two subsets of input parameters and contains a large set $\mathcal{P}_2$. In practice however, due to the heuristics avoiding the selection of submethods by `GPDOF`, $\mathcal{P}_2$ is small. This will be underlined in the experiments made on the two realistic models presented below.

Anyway, a natural question arises: what is the number of input parameters?

Assume that all the r-methods are *square*, that is, they have as many output variables as equations[6]. Let n be the number of variables in the equation system, let e be the number of equations.

If $\mathcal{P}_2$ is empty (because no submethod has been selected by `GPDOF`), then it is straightforward to prove that $|\mathcal{P}_1| = n - e$. This means that, in this case, `GPDOF` builds a minimal parameterization of the model.

In the general case however, it can be shown that:

- $|\mathcal{P}_1| \leq n - e$ and
- $|\mathcal{P}_1| + |\mathcal{P}_2| \geq n - e$

Roughly, this means that the more `GPDOF` must select submethods to calculate a plan, the larger will be the set of input parameters.

To our knowledge, no algorithm is known in the Computer Vision community to compute a *minimum* set of input parameters (i.e., a set of minimum size equal to $n - e$) for a scene defined by general set of constraints and objects in a polynomial time. However, although the practical interest is low, two such algorithms exist, assuming that the constraint system contains only independent equations.

First, applying the well-known standard graph-based algorithm called Maximum-matching on the equation graph yields a minimal parameterization [Dulmage and Mendelsohn, 1958; Pothen and Chin-Fan, 1990]. The input parameters are the variables which are not matched by the algorithm, as shown in Chapter 12[7].

[6]This the case for the 60 r-methods in our dictionary.

[7]Unfortunately, there exists no fast solving method to tackle the different subsystems built by Maximum-matching.

Second, and more interesting in practice, the plan produced by GPDOF can be used to yield a minimal parameterization:

- The first subset $\mathcal{P}_1$ of input parameters is computed as above.
- A second subset $\mathcal{P}'_2$ is built such that $|\mathcal{P}_1| + |\mathcal{P}'_2| = n - e$

The set $\mathcal{P}'_2$ is built by applying a Maximum-matching on every equation (sub)graph corresponding to a submethod: the non-matched variables are added to $\mathcal{P}'_2$. For instance, applying Maximum-matching on the submethods of the plan illustrated in figure 9.3 leads to a set $\mathcal{P}'_2$ made of c_{ld} (non matched in the equation graph of m'_7), c_{lc} (m'_5), c_{lb} (m'_3) and x_{pb} (m'_2; y_{pb} could be indifferently be chosen). In addition to the 2 input parameters x_{pa} and y_{pa}, we thus obtain a minimal parameterization with 6 variables.

Unfortunately, no hard-coded procedure exists to solve the "remaining" equation systems in the submethods, so that the plan can roughly be executed as follows. Consider every r-method m in the plan. If m is not a submethod, execute m. If m is a submethod, solve the corresponding equation system with a standard algorithm (able to compute *all* the solutions), such as a symbolic method, a continuation method [Lahaye, 1934] or an interval based algorithm [Van Hentenryck et al., 1997].

The performance of the second approach above would be even better than the use of Maximum-matching (alone). However, we suspect that it is far from being as efficient as the proposed approach (executing overlapping r-methods). Indeed, several orders of magnitude are gained in performance when a hard-coded procedure is executed.

In conclusion, the heuristics used by GPDOF to select in priority free r-methods that are not submethods gives a quasi-minimal parameterization. Considerations about performance let us think that the variant presented above to compute a minimal parameterization is not promising.

9.3 Dealing with Singularities and Redundant Constraints

One problem of our approach is that our graph-based algorithms may be misleaded by redundant constraints. Also, singular configurations may be left undetected. However, these problems can often be fixed in practice by making use of geometric information.

9.3.1 Redundant Constraints

Redundant constraints involve non-independent equations. Because they correspond to theorems of geometry, the r-methods selected by GPDOF in the plan correspond necessarily to non-contradictory and independent systems of equations. However, GPDOF may fail in presence of redundant constraints because the selection of free r-methods is purely structural. As an example, consider figure 9.4 where an additional parallelism constraint has been added between lines $\mathbf{L}_b$ and $\mathbf{L}_d$. This constraint is redundant with the existing parallelism constraint and prevents GPDOF from finding a plan. Since the selection step of GPDOF is structural, all occurs as if all equations were independent.

It is of course not acceptable to rely on the user to not introduce redundant constraints. Dealing with constraint redundancy has been a subject of research in the CAD community for a long time and it is still an open problem in the general case.

Two straightforward procedures have been introduced in our tool to remove very common causes of redundancy:

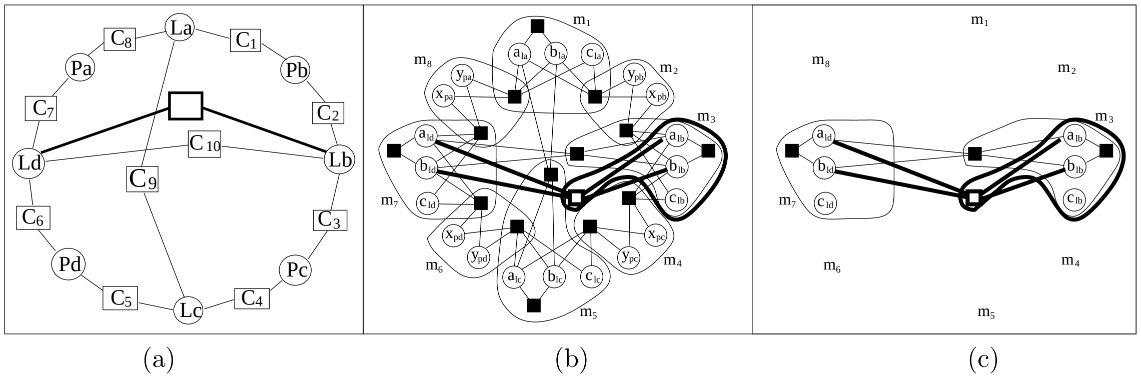


Figure 9.4: Failure of GPDOF in presence of a redundant parallelism constraint. (a) The parallelism constraint is redundant to the parallelism C_{10} (in the constraint graph). (b) The enriched equation graph with the corresponding additional equation (white rectangle) and two additional r-methods (only one is represented). (c) After having removed all possible free r-methods and submethods, GPDOF is stuck because the redundant equation prevents r-method m_3 from being free.

- A redundancy occurs if the user adds an incidence C_1 between a point **P** and a line **L**, an incidence c_2 between the line **L** and a plane **A**, and an incidence c_3 between the point **P** and the plane **A**. In this case, our procedure removes from the whole system all redundant constraints such as c_3 .

- Another straightforward redundancy occurs if the user adds a parallelism p_1 between two lines $\mathbf{L}_1$, $\mathbf{L}_2$, a second parallelism p_2 between the line $\mathbf{L}_2$ and a third line $\mathbf{L}_3$, and a third parallelism p_3 between the line $\mathbf{L}_1$ and the line $\mathbf{L}_3$. In this case, our procedure removes c_3, and is able to apply on a "cycle" of parallelisms of any length.

We found these procedures helpful in practice although we know that many occurring redundancies cannot be handled this way (imagine slightly more tricky configurations combining parallelisms and orthogonalities...).

We hope that special r-methods whose pattern corresponds to a redundant subsystem could be used in a preprocessing step (by searching for the pattern in the whole system) to remove a lot of occurring redundancies[8]. Several works performed by the CAD community follow this idea [Sosnov and Macé, 2002; Sosnov, 2003]. Also, a generalization of the numerical technique mentioned below could be used to detect redundant constraints.

9.3.2 Singularities

We should admit that the singularity issue turned out to be the biggest difficulty we had to overcome in our experiments. While redundant constraints have been rather easily removed automatically or manually, singularities have been the major cause of occasional divergence of the optimization process. We made an effort to make our tool more robust, and we give the main guidelines below.

Let us take an example to illustrate the main difficulty. Assume that 3 defined points are really collinear in the 3D scene, but that the user has not made this information explicit, that is the user has not created a line and has not defined the 3 incidence constraints between every point and this line. The problem is that the automatic r-method addition phase would add the corresponding r-method (i.e., plane based on 3 points) which is singular, thus yielding an infinite number of

[8]"Infinite" patterns must be envisaged for redundant parallelisms for example...

solutions. In practice however, only one arbitrary solution will be returned in the output, leading to distortions in the model. To avoid singularities, we have made several improvements based on basic properties of our system.

First, a singularity occurs only during an r-method execution, and depends only on the values of the input variables and on the constraints satisfied by the r-method. It is thus important to highlight that the system decomposition induced by r-methods (due to their local aspect) is a precious help for handling this issue (manually by the designers at first, and then automatically by the tool).

Second, note that initial values of all the system variables have already been assigned when the r-method addition phase is performed. Thus, it is possible to use a heuristics against a singularity directly after an r-method m has been created and before it is added to the equation graph. Depending on the type of constraints satisfied by the r-method m, different tests are possible. For linear r-methods, we have implemented a test based on the Singular Value Decomposition (SVD) of a matrix formed by the equations of the r-method (where the input variables are replaced by their value). When the matrix is singular, at least one of the singular values vanishes. The r-method m is added to the equation graph only if m is not claimed singular by this procedure.

Since the input data is not perfect, deciding which values should be zeroed is based on a threshold ϵ fixed arbitrarily [Press et al., 1988]. Obviously the performance of this heuristics depends on the choice of ϵ. We have observed however that, for relatively correct input data, the procedure is not very sensitive to modifications of ϵ.

The SVD of the matrix is also used at the r-method creation step in order to choose the input variables ensuring the best conditioning of the r-methods. Details are discussed in Appendix C.

Chapter 10

Optimization Phase

The plan computed by the constraint planning phase is executed numerous times in the optimization phase. As said in the overview, the optimization process produces values for the variables such that all the constraints are satisfied exactly and the reprojection error is minimum[1].

The optimization process is piloted by a standard numerical algorithm. Our tool currently uses the Levenberg-Marquardt algorithm [Press et al., 1988] (we use `vxl` interface [VXL, 2003] for MINPACK routines). However, other optimization methods and libraries could be used instead in our modular architecture. At present, the gradient of the cost function is computed numerically.

The numerical algorithm only modifies the values of the input parameters ($\mathcal{P}_1$ and $\mathcal{P}_2$) . Every time it does so, the plan is executed, resulting in new coordinate values for all the other scene objects. The cost function is then computed as reprojection error of all the points (and possibly lines).

Notice that, due to non-linear r-methods, the execution of a plan can yield several solutions for the variables. Indeed, if the execution of a non-linear r-method in the plan produces k solutions, the number of total solutions given by the plan execution can potentially be multiplied by k. Thus, the total number of solutions is majored by k^r, where r is the number of non-linear r-methods in the plan[2].

Since the optimization process can call thousands of plan executions, we want to avoid a combinatorial explosion during one plan execution. Therefore, a preprocessing phase is performed to select one solution to be followed among the k^r ones. This phase, called *backtracking phase* below, selects, for every r-method in the plan, the solution (index) such that the reprojection error is minimized.

To sum up, the optimization phase is divided into two steps:

1. Based on the plan computed by `GPDOF` and the variable values calculated by the initialization phase, the backtracking phase selects solution indices to be followed during a plan execution. This step is described in section 10.1.

2. Then, the numerical optimization algorithm interleaves input variable modifications and plan executions following the solution indices. This step is described in section 10.2.

[1]Subtleties are detailed below.
[2]Our dictionary contains 7 non-linear r-methods (among 60), each yielding (at most) $k = 2$ solutions.

10.1 Backtracking Phase

This phase performs a combinatorial process which computes a solution with a lowest cost. The precise cost C to be minimized is detailed below. The backtracking executes all the r-methods $(m_1, ..., m_i, ...m_p)$ in the plan in order:

- When all the r-methods have been executed (i.e., the execution of m_p has succeeded), then the new lowest cost C_{best} is updated: $C_{best} \leftarrow C$ [3].
- When an r-method m_i is executed giving possibly k solutions (with $k \geq 1$), the backtracking performs the k choice points, that is, it iteratively replaces the output variables by the values given by the k different solutions and executes the rest of the plan. For every choice point j $(1 \leq j \leq k)$, it updates the output variables of m_i with the values corresponding to the j^{th} solution. The total cost (reprojection error) C is also updated with this modification. Two cases may occur:
  - If $C < C_{best}$, then the backtracking continues with r-method m_{i+1}.
  - Otherwise, if $C \geq C_{best}$, then it is not necessary to continue the process because the cost (error) of the current solution will never be less than the best cost previously found. This branch of the search is cut and one backtracks to the level $i-1$, considering the next solution of m_{i-1} (if any).
- The case may also occur that the execution of an r-method m_i gives no solution.
  - If r-method m_i is linear, then this means that a singularity has occurred during this computation. For example, if m_i wants to place a plane incident to three points which are aligned. In this case, the algorithm does not follow the plan. It cuts this branch and backtracks to the level $i-1$, considering the next solution of m_{i-1} (if any). By doing this, we hope that another branch of the backtracking will change the values of the input variables (e.g., the position of the three points) so that m_i would not be singular anymore.
  - If r-method m_i is non-linear, we preferred to continue with r-method m_{i+1} (no backtrack), and we update the cost C with a measure of the cost violations, as detailed below. This choice increases the computation time but makes the process more robust in case all the solutions obtained during the backtracking have a non-linear r-method giving no solution.

As a result, for every non-linear r-method, the backtracking phase stores the solution number that minimizes the global cost (reprojection error and constraint violations).

Note that the current backtracking algorithm might terminate with no solution in case an r-method selected in the plan is singular in all the branches. We then consider that the whole process fails and that our prototype must be improved to avoid singularities. Most of the features described in section 9.3 come from this strict attitude. However, it is easy to modify the current backtracking scheme to ignore singular r-methods, leaving the corresponding constraints unsatisfied.

The backtracking results depends on the initial values of model variables. It may happen that the solution is not visually correct, especially when the initial reconstruction does not satisfy well the geometrical constraints. However, our experiences have shown that the reprojection error criterion is quite reliable.

In the current implementation, the backtracking step is performed only once before the optimization. Following the remark above, some other backtracking steps could also be performed during the optimization in case the initial reconstruction was very bad.

[3]Before the backtracking, the cost C is initialized to $+\infty$ and is updated every time a new solution is computed.

10.2 Optimization

As explained in the overview, the optimization interleaves the value modification of input parameters (by a Levenberg-Marquardt algorithm) and plan executions.

A plan execution executes the r-methods one by one and updates the cost function incrementally. Several cases must be considered when a non-linear r-method m_i is executed:

- If m_i fails, the constraint violations are added to the cost function.
- If m_i succeeds, the reprojection error corresponding to the output variables of m_i is added to the current cost.

Note that, after several optimization iterations, the model quality increases and the case occurs that a given r-method m_i succeeds for the first time. That is, m_i had always given 0 solution during the backtracking phase and in the previous plan execution steps as well. Consider for example an r-method computing the position of a point using distance constraints from 3 other points. It may happen that the initial positions of these points are inconsistent with the distance contraints, so that the r-method yields no solution. However, with the increasing quality of the reconstruction, the point positions can be moved to positions allowing the distance constraints to be satisfied, and the r-method to give the two possible positions for the output point. When m_i succeeds for the first time, among the k possible solutions yielded by m_i, our optimization process selects the one leading to the lowest reprojection error.

This means that the number of unsatisfied constraints decreases as long as the model quality increases. This great behavior highlights the interest of our fast plan execution step included inside the numerical algorithm.

10.3 Reprojection Error and Constraint Violation Cost

The same cost function is taken into account in the backtracking phase and in the optimization. This cost function has two components: the well-known reprojection error R, but also a constraint violation cost. The latter is the sum of the constraint violation costs induced by all non-linear r-methods in the plan. The constraint violation cost associated to an r-method m_i is 0 if the r-method succeeds and gives one or more solutions. Otherwise, according to a parameter given by the user, the *constraint violation cost* of m_i is equal to:

- either the number of equations C_N handled by m_i,
- or a "smoother" measure C_S of the error related to the semantics of constraints. For instance, the error of a point-point distance is the square difference between the distance value in the equation and the actual distance between the points. The error of an incidence constraint is the square of the actual distance between the two related objects.

The two components yield a multi-criteria minimization problem and the overall cost is a weighted sum between both components: $R + \alpha C_N$ or $R + \alpha C_S$ where α is a weight allowing us to tune the ratio between the components. Experiments detail the results obtained with the two different constraint violation costs and different weights.

Chapter 11

Experimental Results

The approach presented in this part of the thesis has been validated on two models. The first model has been based on 120 objects and 5 images, and the second model has been based on 238 objects and 15 images. Section 11 contains a detailed description of the main characteristics of both models. Qualitative results of reconstruction are presented in section 11.1. Section 11.2 contains performance tests of algorithms and section 11.3 details some practical issues concerning the reconstruction process.

Model Characteristics

We have used our approach to build a model of the *Place Notre-Dame* in Grenoble. A set of images have been used, together with architectural plans from which several distance measurements have been extracted. We have first built a medium-size model constructed from 5 images, called `ND` hereafter. A larger model, called `E_ND`, including peripheric walls and additional details, has then been built from 15 images. The characteristics of these two models are reported in Table 11.1. Three of the images used for reconstructing these models are shown on figure 11.1.

	`ND`	`E_ND`		`ND`	`E_ND`
#images	5	15	#point projections	286	546
#variables	436	819	#equations	273	452
#objects	120	238	#constraints	151	279
#points	90	189	#incidences	124	234
#lines	23	28	#angles	17	34
#planes	7	21	#distances	10	11

Table 11.1: The two scenes: Notre Dame (`ND`) and Extended Notre Dame (`E_ND`)

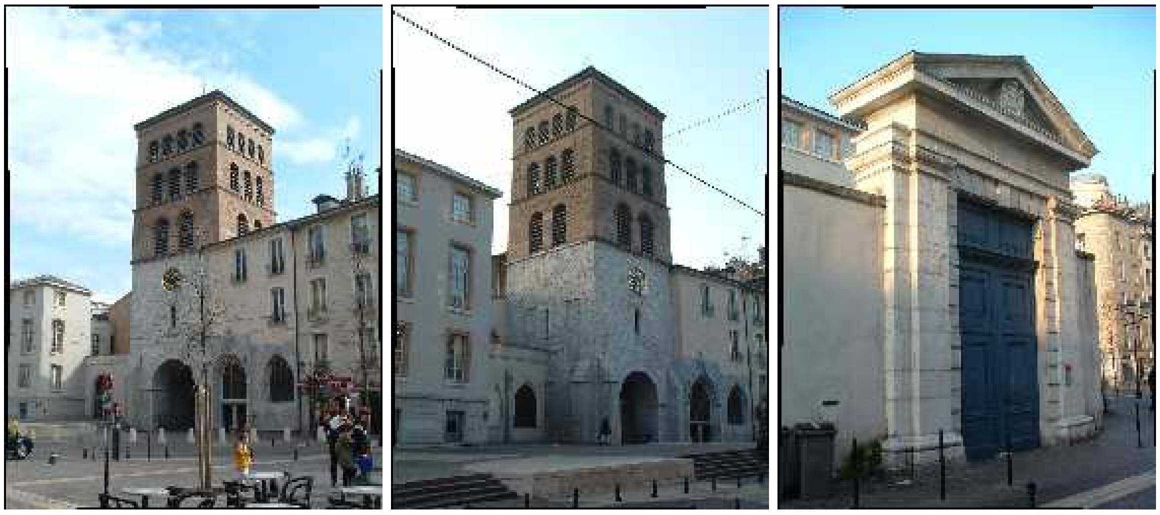

(a) (b) (c)

Figure 11.1: Three images from the sequence *Notre Dame*. Image (c) has been included only into the `E_ND` sequence.

11.1 Reconstruction Results

The multi-linear approach used during the initialization phase takes into account only some bilinear constraints. Thus, for example, distance constraints are not taken into account in the reconstruction of the initial model and may be unsatisfied. Indeed, the maximum relative error of distance constraints is 7% for the `ND` scene and 6% for `E_ND`. The maximum distance between two objects constrained to be incident are 80 cm for `ND` and 60 cm for `E_ND` (to give an idea, the height of the tower is $\sim$ 14m). The maximum angle error is 0.25° for ND and 2.54° for the E_ND scene. The relative error of the violated distance constraints comes down to 0.26% after the optimization process performed by our system (due to 2 distance constraints that are not satisfied). The reconstructed models are presented in figures 11.2 and 11.3.

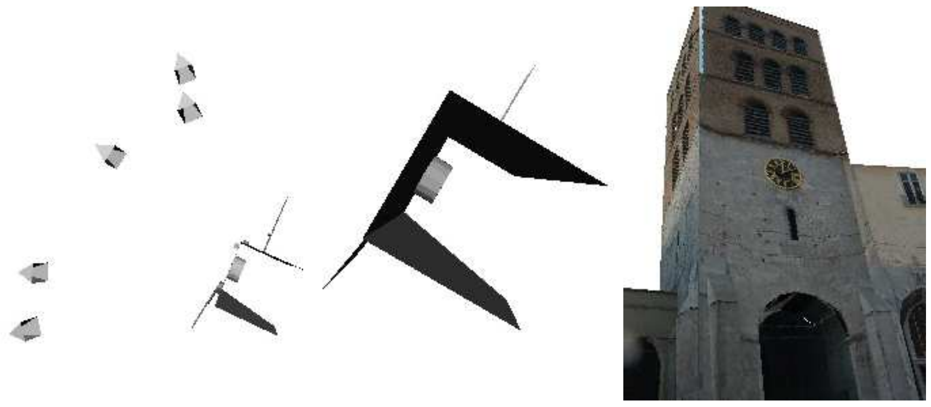


Figure 11.2: Reconstruction of ND scene. (a) Initial model with reconstructed cameras; (b), (c) Views of the final model.

The interest of our method is especially well illustrated in figure 11.3. The reconstruction results highlight that models are visually and geometrically correct. The first column contains the top and side views of the initial model, where contraints are respected approximately. The second column contains the top and side views of the model obtained using a standard unconstrained optimization method. One of the points is visible only in two images with a very small baseline and its position is false due to divergence of the optimization process. Other parts of the model also suffer from several artifacts such as an unsatisfied coplanarity. By imposing appropriate constraints, we have overcome these problems. The third column of figure 11.3 contains the top and side views of the model produced by our method. We show how the parts of the model mentioned above have been corrected, leading to a visually correct model.

Several artifacts have been corrected after several optimization steps, which highlights the interest of our optimization phase and of our fast plan execution (due to r-methods). Also, when in the initial reconstruction the constraints are not sufficiently satisfied, it may happen that a plan execution causes artifacts in the scene, such as in the center of figure 11.5–(a). The optimization corrects the errors created this way (see figure 11.5–(b)).

Different stages of the model reconstruction, from the set of input parameters, is shown in figure 11.6. For the sake of clarity, we show results of only the smaller scene ND. In the top left image are represented objects whose coordinates are included into input parameters. Onto the bottom right image, used for the reconstruction, are superimposed all the model objects. At each r-method execution, the corresponding output object is added to the model.

Table 11.2 reports statistics about the number of r-methods added automatically to the equation graph (#r-methods), the number of r-methods selected in the plan (plan size), the number of bundle adjustment iterations (#iterations) and plan executions (#executions) performed by the optimization. The reprojection error (reproj. error) and the number of violated constraints (#violated) is given before the optimization (i.e., after a single plan execution), and also after the optimization.

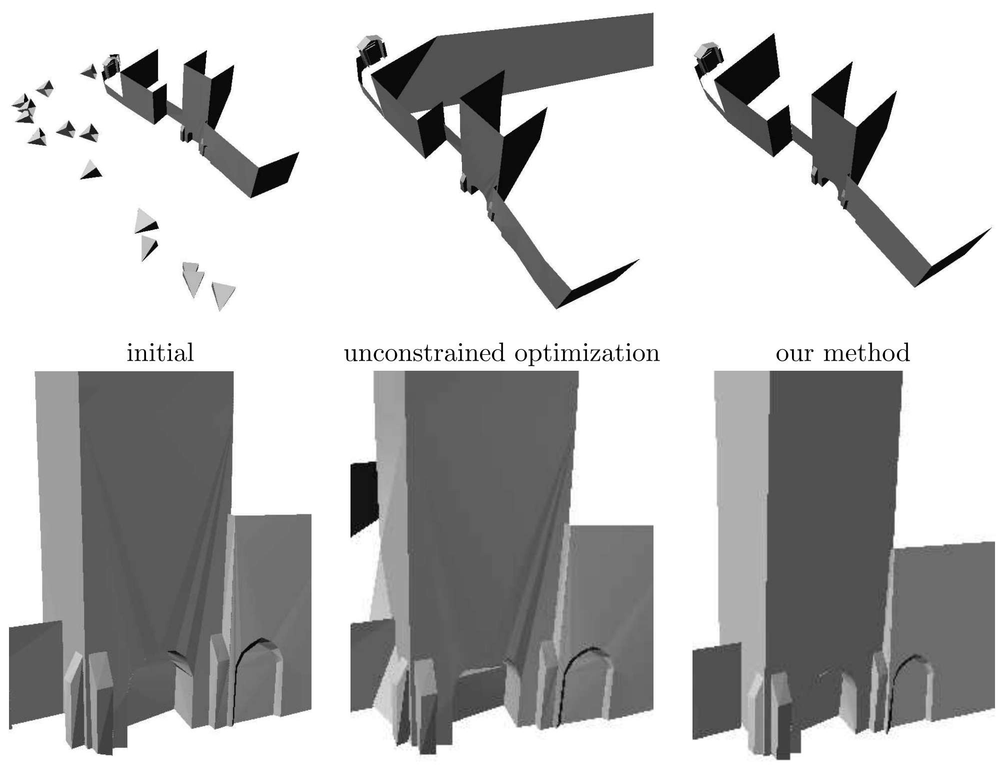


Figure 11.3: Reconstruction of the `E_ND` scene. Top (1^{st} row) and side (2^{nd} row) views of the initial model (1^{st} column), model obtained using unconstrained optimization method (2^{nd} column) and using our method ($3^{rd} column$).

11.2 Performance Tests

All the times reported below have been obtained with a `Linux` operating system on a `Pentium IV 2 Ghz` processor.

Recall that our r-method dictionary contains 60 r-methods. The most complex r-methods solve 3 geometric constraints (6 equations) and imply 4 geometric objects (1 as output and 3 as input).

We have first evaluated the time required to execute one r-method. This time varies from $1 \cdot 10^{-15}$ s to 90μs and depends on the type of r-method. For example, an r-method satisfying a parallelism constraint between two directions needs only 3 assignment operations and is very fast. The execution time of linear r-methods depend on the number of (output) variables and varies from 4μs to 90μs ($\sim$ 30μs on average). The execution time of non-linear r-methods varies from 2μs to 30μs ($\sim$ 28μs on average). The execution time of non-linear r-methods is shorter because the corresponding procedures are hard-coded, while linear r-method routines use a generic SVD (Singular Value Decomposition).

Table 11.3 details the times spent in the different phases of our model reconstruction system.

The time for the initialization phase is dominated by the non-linear unconstrained optimisation process ($\sim$ 80% of time) which is executed to refine the initial, parallelepiped-based calibration. This step could be skipped. A very interesting characteristic of our system is that the constraint planning phase (automatic r-method addition, `GPDOF` and backtracking) requires only a few seconds.

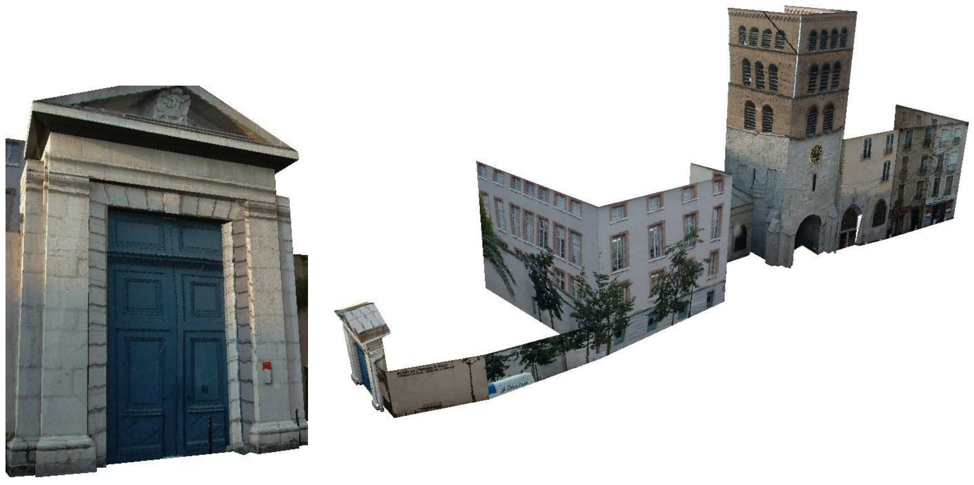

(a) (b) (c)

Figure 11.4: Screenshots of the textured model obtained by our method.

11.3 Details

Exploration of All the Connected Subgraphs of Size k at Most

Table 11.2 (right) clearly shows that the exploration of all the connected subgraphs of size at most k is fast (0.72 s on ND). As mentioned in Section 9.1.1, the time complexity of this phase is highly dominated by the number of connected subgraphs. This number strongly depends on the size k of the largest subgraph. In the first version of our tool Wilczkowiak et al. [2003d]; Trombettoni and Wilczkowiak [2003], this phase was even more time-consuming (253 s on ND). The exploration of the constraint graph considered connected subgraphs in terms of objects *and* constraints. The value of k was 7 because the largest r-methods in the dictionary include 3 geometric constraints

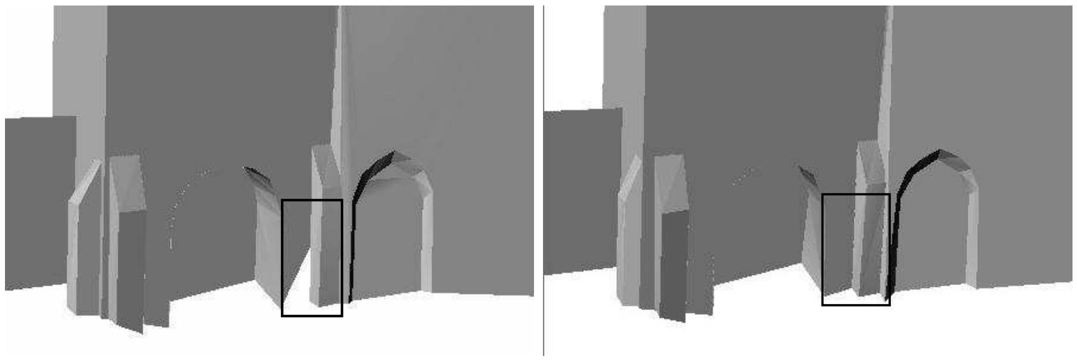

(a) (b)

Figure 11.5: Part of the E_ND scene after (a) one step of GPDOF; (b) optimization: initial artifacts are corrected.

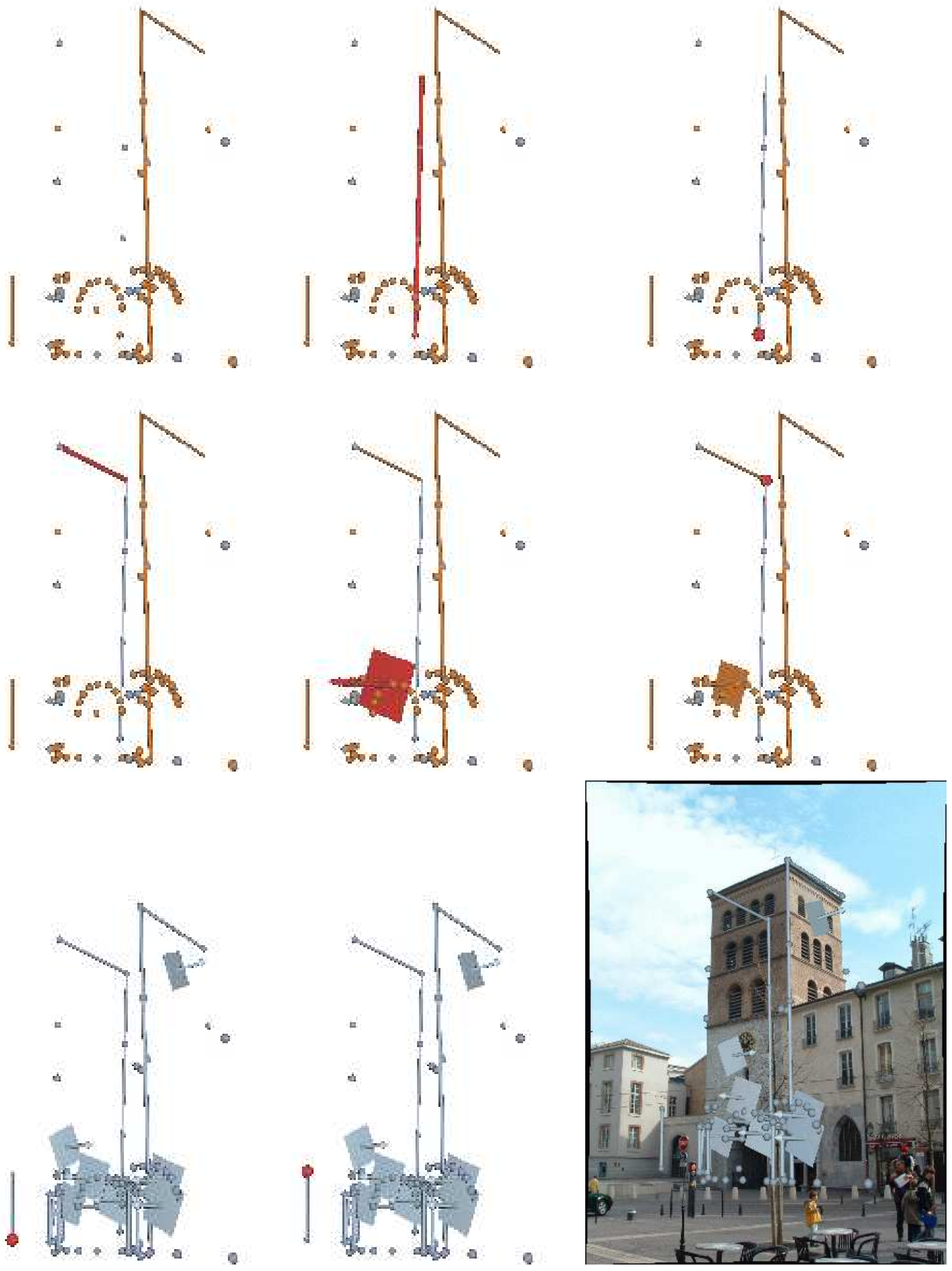


Figure 11.6: Plan execution for the ND scene. In grey are objects whose coordinates are fully fixed (input parameters), in orange objects whose coordinates are partially fixed and in red objects reconstructed in the last executed method. The 1st and 2nd lines present objects after the execution of the five first r-methods in the plan; the 3rd line presents the scene after having executed the two last r-methods in the plan, and one of the images with superimposed primitives representing the complete model.

	ND	E_ND		ND	E_ND
# r-methods	3017	6695	Plan size	118	228
# iterations	11	17	$\mathcal{P}_1$	158	352
# executions	2040	7866	$\mathcal{P}_2$	10	25
Reproj. error before	20.42	23.01	Reproj. error after	4.038	4.16
# violated before	2	2	# violated after	2	1

Table 11.2: Statistics on the model reconstruction.

Phase	ND	E_ND
Initialization	21	331
R-method addition	0.7	1.7
GPDOF	1.4	3.9
Backtracking	0.2	0.2
Optimization	53	467

Table 11.3: Performance of the different phases of our model reconstruction system (in seconds)

and 4 objects. In the new version, $k = 3$ because the connected subgraphs are built in terms of constraints only. Two constraints are neighbors in the constraint graph iff they share an object.

k	# subgraphs	# con. subgraphs	time [s]	# con. subgraphs	time [s]
		(objects+constraints)		(constraints)	
2	3.6×10^4	1.5×10^3	0.03	944	0.06
3	3.3×10^6	4.5×10^3	0.11	6578	**0.72**
4	2.2×10^8	1.5×10^4	0.5	48304	5.2
5	1.2×10^{10}	7×10^4	3.6	348310	75
6	5.3×10^{11}	3.8×10^5	31	2.4×10^6	767
7	2×10^{13}	2.55×10^6	**253**	1.6×10^7	6774

Table 11.4: Exploration of all the connected subgraphs of size at most k in ND. The results highlight three interesting points. First, limiting the search to *connected* subgraphs is crucial and leads to gain 7 orders of magnitude for $k = 7$. Second, searching for connected subgraphs in terms of constraints is more time-consuming, but only a constant factor of value around 25 is lost. Anyway, the number of connected subgraphs increases exponentially with k and our approach cannot be used in practice for patterns of size more than 10.

In conclusion, we believe that our simple automatic r-method addition algorithm can handle in practice any type of r-method that outputs a single object (point, line or plan), even if other types of constraints are added, such as angles, distance ratios. Indeed, the degree of freedom of a rigid object is at most 6 in 3D and so the implied geometric constraints cannot retrieve more than 6 degrees of freedom, bounding k by 6, but rather 4 (since basic objects have at most 4 degrees of freedom and constraints retrieve at least 1 degree of freedom).

Playing with the Multi-Criteria Cost Function

In the backtracking phase, we have experimented different cost functions on our two models by using C_N (number of violated constraints) or C_S (degree of violation of the constraints) and making the weight α vary in the set $\{1, 10, 100\}$. This has led to choose C_S for the constraint cost and $\alpha = 1$. Especially, a large value for α leads to occasional divergences of the optimization.

We suspected that C_S gives better results because the constraint part and the reprojection part of the cost have a similar order of magnitude. Choosing a large value for α gives a great importance to the violation of a non-linear constraint c solved by an r-method m. However, if the values of the input variables of m are bad (recall that the optimization has yet been performed), respecting the constraint c implies that the subsequent r-method executions will imply a large reprojection error. The same principle would suggest to attach an increasing importance to the satisfaction of non-linear equations during the optimization.

11.4 Comparison with the penalty function method

In order to give an intuition about the performance of our method comparing to the existing constrained optimization methods we have implemented the penalty function method. The most important advantage of this method is its universality. Indeed, the constraints are imposed by simply adding corresponding penalty terms to the cost function.

More precisely, the solution to the reconstruction problem is found as:

$$\mathbf{x} = \operatorname{argmin}(\hat{C}), \quad \hat{C} = f(\mathbf{x}) + \frac{1}{2}K\|g(\mathbf{x})\|^2, \tag{11.1}$$

where:

- vector $\mathbf{x}$ contains terms corresponding to camera intrinsic and extrinsic parameters, as well as to coordinates of all the objects (points, lines and planes);
- $f(\mathbf{x})$ contains terms corresponding to point reprojection errors;
- $g(\mathbf{x})$ contains terms corresponding to the model constraints;
- K is a weight corresponding to the model constraints.

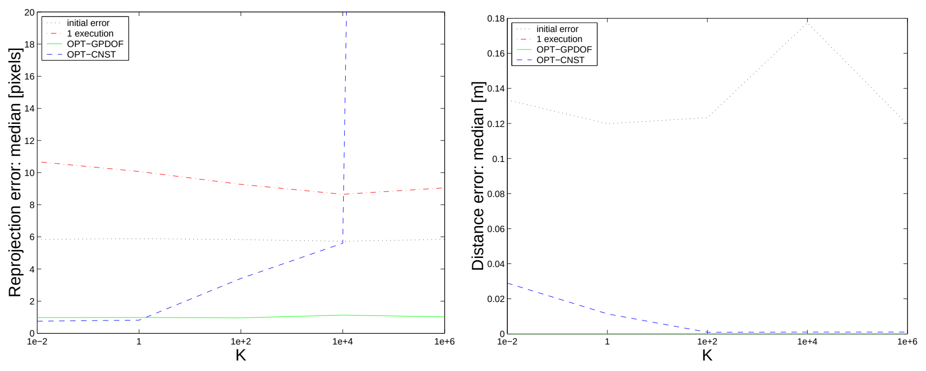
The solution is found using the same Levenberg-Marquardt routine which is used to optimize the model parameterized using GPDOF. The function gradient is computed numerically. In each cost function calculation step the camera and object parameters are updated using current values contained in vector $\mathbf{x}$ and used to compute the point reprojection errors as well as terms corresponding to the model constrains.

11.4.1 The experimental results

In order to perform the experiments we have created first a data set respecting all the geometrical and projection constraints *perfectly*. The model `ND` was first parameterized using GPDOF and optimized, providing a model respecting the geometrical constraints. Then the model points were reprojected using the computed camera projection matrices, providing perfect 2D data.

We have tested performance of both GPDOF-based and penalty function methods with varying values for parameter K in expression (11.1) and by application of increasing Gaussian noise on 2D data and initial values of 3D object coordinates. Applying noise to the values of coordinates of 3D objects, we have distinguished coordinates corresponding to direction vectors for lines and planes, which are scaled to unit length and the remaining coordinates, defining the positions of objects in space. The noise applied to the directions was constant and corresponding to angle $\alpha \sim 0.33°$ on average. The errors on 3D positions are given in meters to give an idea of the real errors on the model of dimensions $\sim$ 60x30 meters. The results shown below are the median results from 15 randomly generated data sets. We illustrate the values for the obtained reprojection errors as well as errors on distance constraints. The values are given for the initial model, the model after satisfaction of constraints using one execution of the GPDOF plan, the model optimized using the GPDOF-based method and the model optimized using the penalty function method.

Varying parameter K. Figure 11.7 shows results of tests with a varying parameter K. The initial solutions for the optimization methods were obtained by application of a Gaussian noise to the image data and the initial 3D object with respectively, σ=1.0 pixels and $\sigma = 0.13$m. The parameter K was varying from 10^{-2} to 10^{6}. The time needed for convergence was oscillating around 50 seconds for the GPDOF-based method and 200 seconds for the penalty function method. Figure 11.7–(a) illustrates the impact of the parameter K on the reprojection error. For K up to 1 both

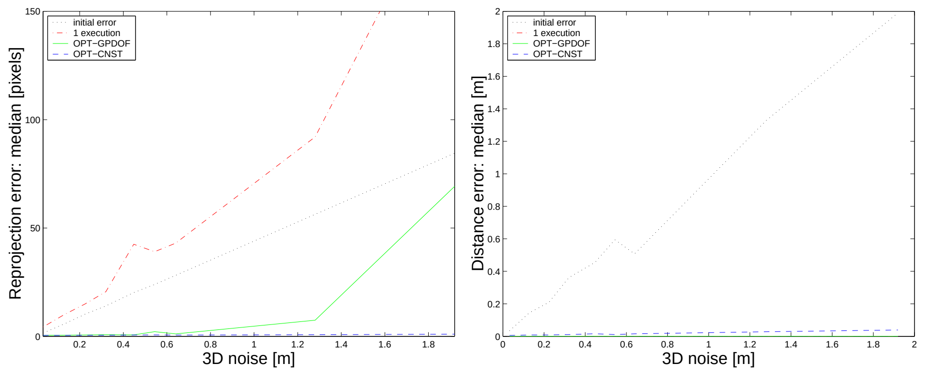


(a) (b)

Figure 11.7: Optimization results as a function of increasing K factor; (a) Median reprojection error; (b) Median distance error. Note that after at least one GPDOF execution the distance constraints are respected exactly, so the curves corresponding to the model after 1 GPDOF execution and after the GPDOF-based optimization are identical to the x-axis.

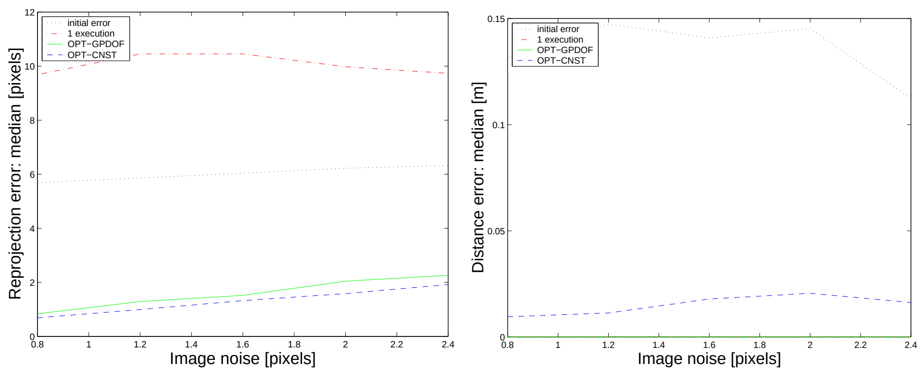
methods give similar reprojection error results with a mean reprojection error smaller than 1 pixel. As expected, an increasing value K leads to an increasing reprojection error. For $K = 10^6$ the penalty function method diverges. Figure 11.7–(b) illustrates the satisfaction of distance errors as a function of the increasing factor K. The initial median error of the distance constraints is around 0.13m. While the GPDOF-based method results in models respecting exactly the constraints, the Penalty Function method for factor K smaller than 100 results in models with a median distance error of about 2-3 cm.

Varying 3D noise. Figure 11.8 illustrates results of tests as a function of the Gaussian noise applied to the 3D object coordinates with σ varying from 0.032m to 1.92m. The penalty function method was executed with K=1. The execution time varied from 100 to 150 seconds for the GPDOF-based method and from 200 to 400 seconds for the penalty function method.



(a) (b)

Figure 11.8: Optimization results as a function of increasing 3D noise; (a) Median reprojection error; (b) Median distance error. Note that after at least one GPDOF execution the distance constraints are respected exactly, so the curves corresponding to the model after 1 GPDOF execution and after the GPDOF-based optimization are identical to the x-axis.



(a) (b)

Figure 11.9: Optimization results as a function of increasing 2D noise; (a) Median reprojection error; (b) Median distance error. Note that after at least one GPDOF execution the distance constraints are respected exactly, so the curves corresponding to the model after 1 GPDOF execution and after the GPDOF-based optimization are identical to the x-axis.

The median reprojection error of the penalty function method remains smaller than 1 pixel (figure 11.8–(a)). The error for the GPDOF-based optimization method is not larger then 7 pixels for a 3D noise smaller than 1.3m. For larger values the reprojection error increases importantly. This is mainly due to the fact, that with very inaccurate initial object coordinates, one execution of the GPDOF-plan, satisfying the model constraints, can violate the reprojection constraints (see the red curve). The optimization of the model decreases the reprojection error, however is easily stuck in a local minimum. The satisfaction of the distance errors is illustrated on figure 11.8–(b). The penalty function method leads to errors not larger than 0.04m.

Varying 2D noise. Figure 11.9 illustrates results of tests as a function of noise applied to the 2D point projections with σ varying from 0.8 to 2.4 pixels. In all experiments also a Gaussian noise with σ =0.13m was applied to the 3D object positions. The penalty function method was executed with K=1. The execution time was approximately 60 seconds for the GPDOF-based method and 160 seconds for the penalty function method.

Due to the application of the 3D noise, the initial values of the reprojection errors are about 6 pixels for all the values of the applied 2D noise. After one execution of the GPDOF plan the reprojection error increases to about 10 pixels. After the optimization however, both GPDOF-based and the penalty function methods converge to models with similar average reprojection error not larger than 2 pixels (figure 11.9–(a)). The median error on distance constraints is slightly increasing for the Penalty Function method, but remains smaller than 0.02m.

The above results show, that the convergence time of the GPDOF-based method is on average between 2 and 4 times smaller than for the penalty function method. The GPDOF method is more sensitive to the accuracy of the initial data, however up to a relative error 3% on the initial object positions reveals very good convergence properties. When the satisfaction of the geometrical constraints is really important, for example for visualization goals, it seems also to be more convenient than the penalty function, which tends to diverge for important values of the penalty factor K. When the initial data is too noised to obtain a satisfying solution using GPDOF-based optimization, a possible way to proceed is to use the penalty function optimization to refine the 3D structure and then run the GPDOF-based optimization method to impose the constraints exactly.

Chapter 12

Comparing GPDOF *with Equation Decomposition Systems and Geometric Solvers*

This chapter shows that GPDOF compares favorably with other algorithms for decomposing a system of geometric constraints. Section 12.1 focuses on algorithms developed by the Computer Aided Design community, while section 12.2 describes general-purpose algorithms that have been applied to geometric constraint systems.

12.1 Geometric Solvers

Some systems use algorithms which are specific to restricted subclasses of constraints. Owen's method [Owen, 1991] is limited to 2D points and lines with distance and angular constraints. Sunde's method [Sunde, 1986] (extended by Verroust in [Verroust et al., 1992]), considers points and segments in 2D with distance and angular constraints.

Several algorithms ([Kramer, 1992; Fudos and Hoffmann, 1997; Hoffmann et al., 1997; Jermann et al., 2003]) try to recursively rigidify subparts of the constraint system (i.e., rigid subparts are detected and assembled together). However, these algorithms are not convenient for under-constrained systems. Indeed, they cannot decompose non rigid subparts and cannot compute input parameters.

The algorithm designed by Bondyfalat et al. [D.Bondyfalat et al., 1999] for scene modeling uses a propagation mechanism to place one object after the other. The constraints must be binary (relating at most 2 objects), and only linear or bilinear constraints are taken into account. Although efficient heuristics have been made to choose the next object to place (step 1 below), this algorithm cannot guarantee finding a correct order without a backtracking step, i.e. in polynomial time.

Figure 12.1 shows an example. Let us first recall its principle. The algorithm is based on a constraint graph labeled with the degrees of freedom of objects and constraints. The propagation is performed as follows:

1. Select one unbuilt object O with a minimal degree of freedom. (Other heuristics are also used to discriminate the objects.) All different choices are performed when no solution is found, making the whole propagation process combinatorial.

2. Compute coordinates of O by using the equations engendered by constraints related to the object.

3. Consider every object O' linked to O by a constraint C. Decrease the degree of freedom of O' by the degree of freedom of C.

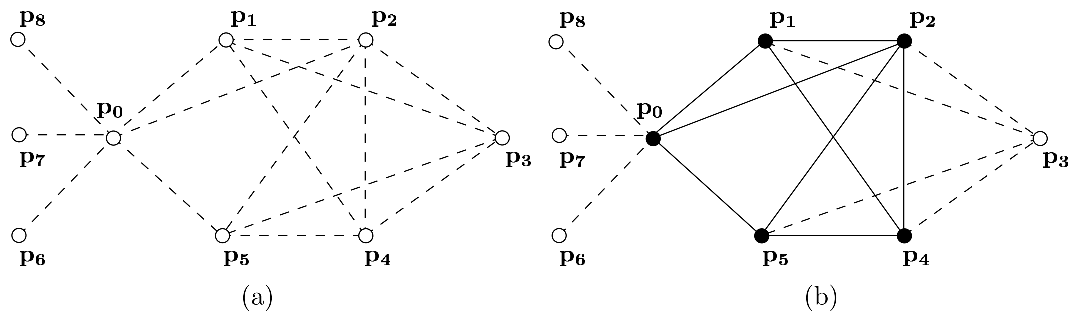


Figure 12.1: An example for which the algorithm proposed in [D.Bondyfalat et al., 1999] fails without a backtracking step. Circles represent points $\mathbf{p}_0$-$\mathbf{p}_8$ in 3D. Lines represent distance constraints. Lines are solid when the constraints are already satisfied. The subfigure (a) illustrates the initial graph. At first, all the objects have three degrees of freedom. Point $\mathbf{p}_0$ is reconstructed first because it is involved in six constraints. Then, points $\mathbf{p}_2, \mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_4$ are reconstructed in order, giving the graph shown in subfigure (b). Point $\mathbf{p}_3$ cannot be reconstructed based on the four distance constraints implying it (overconstrained configuration), which causes a backtracking.

Note that GPDOF can easily find a plan reconstructing in order $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_0, \mathbf{p}_6, \mathbf{p}_7, \mathbf{p}_8$.

It appears that the backtracking mechanism cannot be avoided in reactive propagation mechanisms[1] because "bad" choices of input parameters can be made by them (e.g., the point $\mathbf{p}_0$ in figure 12.1).

12.2 Equation Decomposition Systems

Numerous interactive applications, such as [Ducasse et al., 1995], use a variant of a reactive algorithm (like the one described above but at the variable-equation level), to decompose a constraint system. As explained in [Trombettoni, 1997], this approach is blind and cannot guarantee finding an order without backtracking.

Other approaches are more sophisticated and proceed in two phases: a planning phase computes the order and the execution (or evaluation) phase solves the found sub-systems. Two among these approaches come from a field called *local propagation* [Sutherland, 1963; Borning, 1979; Freeman--Benson et al., 1990] that intended to solve interactive systems, such as drawing applications and advanced user interfaces. They can compute a plan including r-methods that solve only *one* constraint and have only *one* output variable. Roughly, the *propagation of conflicts* computes the sequence of r-methods from the first to the last one, while the `PDOF` scheme does it in reverse order. As shown in [Trombettoni, 1997], the propagation of conflicts scheme becomes exponential if general r-methods are allowed while `PDOF` remains polynomial (giving `GPDOF`).

Another algorithm has been used for the planning phase, first, by the local propagation community [Gangnet and Rosenberg, 1992], second, for solving very large systems of linear equations [Dulmage and Mendelsohn, 1958; Pothen and Chin-Fan, 1990], and third for solving geometric constraints [Ait-Aoudia et al., 1993; Lamure and Michelucci, 1997]: the *Maximum-matching* of the equation graph performed by the well-known standard graph-based algorithm. We show an example to highlight why `GPDOF` compares favorably to `Maximum-matching` for decomposing a system of geometric constraints.

A *maximum matching* of a bipartite graph yields a subset of edges in the graph with a maximum size and such that every pair of edges in the matching does not share a vertex. Briefly, applying a `Maximum-matching` on the equation graph yields:

- a sequence of subsystems of equations (an edge in the matching yields an equation and an output variable of one subsytem),
- a *minimal* set of input parameters (the variables that are not incident to an edge in the matching),

Applied to our 2D scene, `Maximum-matching` could yield exactly the same plan as the one given by `GPDOF`. Unfortunately, the algorithm may also produce subsystems of arbitrary size and for which no specific solving procedure exists. Figure 12.2 shows that `Maximum-matching` could yield a bad plan with one large subsystem including the whole system.

To sum up, `Maximum-matching` is a pure graph-based algorithm that produces any type of sub-system[2] and of arbitrary size, while `GPDOF` yields geometrically correct subsystems associated to a fast solving routine (i.e., an r-method).

[1]A *reactive* algorithm has no planning phase. It selects a subsystem and solves it immediately (on the fly).

[2]A subsystem may also contain contradictory equations due to a bad choice of input variables (see [Trombettoni, 1998]).

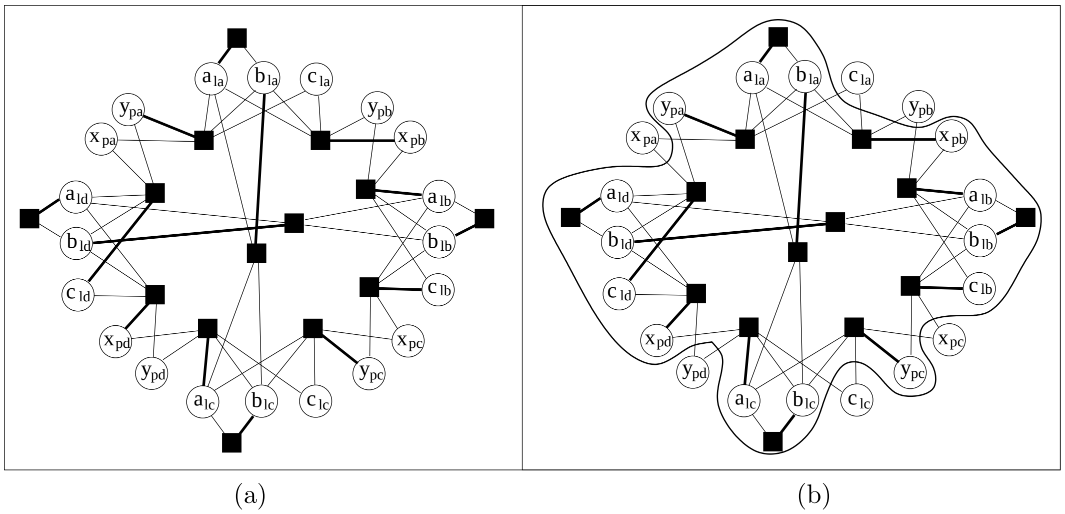


Figure 12.2: Applying `Maximum-matching` to the 2D scene. (a) The edges of the matching are bold-faced; (b) Only one subsystem is computed. No general and efficient method is known to solve such systems of bilinear equations.

Chapter 13

Conclusion and Perspectives

Image-based reconstruction of photorealistic three-dimensional models of objects is often an under-constrained problem. This is mainly due to the fact that, while the human vision system is able to simultaneously process various types of incoming information, only a subpart of this information is usually treated at once by computer vision algorithms. Among the various features revealing three-dimensional characteristics of a scene we were mainly interested in the use of geometrical constraints, such as collinearity, parallelism, orthogonality, etc. These are common in man-made environments, and relatively easy to introduce interactively by a user, and to model in the system.

Using prior information on a scene together with camera information enables the reconstruction of models from a small set of images. Indeed, approaches presented in this thesis were validated on models reconstructed from 1 to 15 images. When only a small number of images is used it is possible to rely on a human operator and his intuitive knowledge of the main properties of the objects commonly present in scenes. The experimental results presented in this thesis illustrate that interactive input enables the reconstruction of models from sets of sparse images, with small overlap and numerous occlusions.

The three concepts and accompanying methods proposed in this thesis cover initial calibration, initial reconstruction and non-linear refinement steps of 3D model acquisition from uncalibrated sets of images. Although all the approaches exploit prior information about scene and camera parameters, they differ in the way the available data is represented and incorporated into the system. Let us separately discuss the proposed algorithms and their possible extensions.

Parallelepiped-based calibration. An approach for calibration and pose estimation using projections of parallelepipeds was presented in chapters 3-5. We have shown that useful constraints such as parallelism, coplanarity and right angles, can often be nicely modeled via parallelepipeds. Especially, this allows to couple constraints between neighboring scene primitives (points, lines, planes), which potentially induces a higher stability than only using constraints between pairs of primitives. Moreover, a satisfying solution can already be obtained with a small number of images and correspondences (starting from 4 correspondences per image pair or 6 per image and parallelepiped).

We have introduced the notion of duality between the calibration primitives: parallelepipeds and cameras. One of the consequences of this duality is the fact that the calibration problem can be formulated in terms of a canonic cube instead of an absolute conic. The notion of duality was used to propose a factorization framework for the estimation of camera intrinsic parameters and euclidean shapes of parallelepipeds, as well as their poses. Using well constrained, three-dimensional structures allows to easily deal with common problems of factorization approaches: missing data and unknown scale factors.

The calibration method is completed by a detailed study on singular cases. Singularities are derived theoretically, and the impact on the method's performance due to the proximity to singular configurations is shown by simulated experiments. Experiments with real images show that our calibration approach gives excellent initial results for general 3D model reconstruction methods.

We believe that the presented approach is a useful tool for easily calibrating cameras using images of unknown though constrained scenes. Also, it allows to efficiently obtain models of the global structure of scenes (including camera pose), which are good starting points for methods aiming at a more automatic and/or flexible model definition, such as the two methods described below.

A possible extension of this system consists of formulating the calibration constraints based on other types of primitives, like prisms, rectangles etc, which would allow to linearly exploit all the prior scene information. Such an approach might be coupled with the multi-linear reconstruction method presented below.

Multi-linear reconstruction approach. In chapter 6 a multi-linear method for the reconstruction of models defined by points, lines and planes and bilinear constraints between them was presented. The main algorithm behind this method is a practical approach for the detection of well-constrained variables in linear equation systems. This approach is based on the SVD of the equation system's matrix and can be applied straightforwardly. Its application domain covers in particular all computer vision algorithms based on linear algebra. In the implemented reconstruction system it enables the detection of features which are sufficiently constrained and, consequently, to propagate the available geometrical information.

Although the reconstruction results are very promising, the performance of the method would be improved by the incorporation of an uncertainty analysis. It would allow to vary the influence of the objects on the reconstruction process, depending if their definition is close to singular or well defined. In order to introduce a more systematic error analysis, solutions based on Total Least Squares [Van Huffel and Vanderwalle, 1991] are for example possible. On the other hand, interchanging reconstruction with camera recalibration steps, exploiting all the currently available object and constraints, would increase applications and robustness of the method. Indeed, this would allow the refinement of the camera calibration during the process, leading to better reconstruction results. This would enable also reconstruction from sets containing uncalibrated images.

Scene modeling based on constraint system decomposition techniques. In part III we have presented a solution to the problem of 3D scene modeling under geometric constraints, based on techniques for constraint system decomposition. The proposed method is original and efficient. A quasi-minimal set of input parameters and a sequence of routines whose execution results in a model respecting the constraints (plan) are extracted in polynomial-time. Executing a sequence of fast r-methods (which take into account geometrical properties) can build a model that satisfies the constraints exactly. Compared to related approaches, our reconstruction system is very fast, and flexible enough to accept different types of constraints.

Our system has been validated on two models with several hundreds of primitives and geometric constraints, the constraints being linear, bilinear or quadratic. The obtained results are geometrically correct (all bilinear constraints and most of the distances are satisfied exactly) and fit well to the images.

Several improvements can be thought of, as described in the following.

Several implementation issues related to singularities and redundant constraints have been implemented. Further developments could be performed to detect other cases of redundant constraints (see section 9.3).

In the current version, the backtracking is performed only once. It seems interesting to incorporate several backtracking phases inside the optimization to change the backtrack solution followed (among all the possible solutions) as long as the reprojection error decreases.

Numerous different plans can be built by `GPDOF` depending on the r-method selection order. Without going into details, we think that the shape of the model depends on the computed plan. Several heuristics for the r-method selection should be compared, e.g., so as to uniformly distribute the input parameters throughout the model.

For the plan computation, we have proposed a variant that can obtain a minimal parameterization (see section 9.2.3). Although not promising because of performance considerations, this variant should be compared to the current version. In this case, the submethods could be solved by interval techniques [Van Hentenryck et al., 1997].

We hope that our work will be considered an original and significant advance in model reconstruction under constraints and will be followed by other developments.

As mentioned above, the approaches proposed in this thesis cover all the stages needed to extract a three-dimensional model from an uncalibrated set of images. However, several extensions can be thought of to make the model acquisition process less laborious and easier for an inexperienced user.

First, incorporation of techniques enabling at least partially automatic input would decrease the amount of necessary user interaction.

At the preliminary stage, matching methods for sparse image sequences might be used to detect and match simple primitives such as points, lines or planes. Knowing these pertinent image features, some assumption on the scene, such as the dominant orthogonal directions, combined with information on camera parameters might be used to create hypotheses concerning geometric dependencies between objects.

More difficult is the detection and matching of more complicated objects, such as cubes and prisms. Indeed, a human operator can perform these tasks mainly thanks to experience in analysis of 3D scenes from images. Introducing equivalent expert reasoning into a reconstruction system would require complicated learning and recognition procedures.

Interesting possibilities appear when the model is partially reconstructed. Indeed, when approximate 3D structure and camera calibration are known, it is possible to compare reprojection of model features with the original image data. This can be used to refine the reconstruction: feature positions and camera parameters might be adjusted and new features could be detected and matched between images. Moreover, approximate scene structure can be used again to create hypotheses on inter-object dependencies or presence of complex primitives. Finally, the surface of the model can be automatically computed, for example by generating hypotheses on triangles belonging to the model and comparing them to the image data. Surface reflectance can then be computed by fusion of color information from the images.

The main problems concerning automatic matching and surface generation methods are occlusions and variation of color between images. Indeed, depending on the point of view, different parts of objects are visible, and the angle between camera and light source varies, changing the appearance of the observed surface. Moreover, depending on the exposure settings and current illumination, whole ranges of color may change between two images taken from the same point of view. Progress in the understanding of color variations between images has been significant in the last decade, however it remains an open problem for general scenes viewed by uncalibrated cameras.

Another possible extension is related to the fact that when working with small image sets it is very common to encounter singular configurations, both for calibration and reconstruction. We have proposed algorithms to detect which cameras and objects are underconstrained. However, a more precise analysis of the underconstrained features might give directions to the user how to add supplementary images or information in order to improve the reconstruction.

In this thesis we were interested in using geometric constraints in a most general way possible. We studied large variety of primitives and constraints and proposed approaches to combine all the available data. However more general systems, able to deal with other sources of information can be thought of. Firstly, integration of dense data acquired using scanners, shape-from shading, or visual hull approaches would significantly increase the possibility of modeling irregular scene elements. Secondly, integration of an image-based reconstruction system into a CAD system would allow to take advantage of the scene modeling methods developed in this field.

Appendix A

Singular Value Decomposition

The Singular Value Decomposition (SVD) methods are based on the following theorem of linear algebra [Press et al., 1988; Golub and van Loan, 1989; Bjorck, 1990]:

Theorem 1 *Any $m \times n$ matrix A can be represented as product of an $m \times n$ column-orthogonal matrix U, an $n \times n$ diagonal matrix W and the transpose of an $n \times n$ orthogonal matrix V:*

$$\begin{pmatrix} \\ \mathsf{A} \\ \\ \end{pmatrix}_{m\times n} = \begin{pmatrix} \\ \mathsf{U} \\ \\ \end{pmatrix}_{m\times n} \begin{pmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{pmatrix}_{n\times n} \begin{pmatrix} \mathsf{V}^\mathsf{T} \end{pmatrix}_{n\times n}. \tag{A.1}$$

This decomposition is unique up to:

- *making the same permutations of the columns of U, elements of W and colums of V;*
- *forming linear combinations of any columns i, j of U and V corresponding to equal values w_i, w_j.*

In the following we suppose that the values w_i are sorted in a decreasing order: $w_1 \geq w_2 \geq \cdots \geq w_n$. Let us summarise some useful properties of the decomposition (A.1):

- The elements $w_i,\ i \in [1 \ldots n]$ are the singular values of matrix A.
- One definition of *condition number* of a matrix is defined as the ratio $\frac{w_{max}}{w_{min}}$, where w_{max} is the largest of the absolute values of the w_i and w_{min} is the smallest one [Press et al., 1988]. The matrix is *singular* if its condition number is infinite and *ill-conditioned* if its reciprocal approaches the machine's floating point precision (e.g. $1e^{-6}$ for single precision and $1e^{-12}$ for double).
- The (Moore-Penrose) pseudo-inverse of a matrix is defined as

$$\mathsf{A}^+ = \mathsf{V}\left[\operatorname{diag}\left(\frac{1}{w_i}\right)\right]\mathsf{U}^\mathsf{T}, \tag{A.2}$$

where the expression $\frac{1}{w_i}$ is replaced by 0 for w_i=0. Indeed, due to the column-orthogonality of U and V, $\mathsf{A}^+\mathsf{A} = \mathbf{I_n}$. For square non-singular matrices, the expression (A.2) gives the inverse of the matrix: $\mathsf{A}^{-1} = \mathsf{A}^+$.

- Let us consider a linear mapping $\mathbb{R}^n \to \mathbb{R}^m$ represented by matrix A:

$$\mathsf{A}\mathbf{X} = \mathbf{B}. \tag{A.3}$$

When A is singular, then there exist a subspace of $\mathbb{R}^n$ mapped to zero: $\mathsf{A}\mathbf{X} = 0$. This subspace is called the *nullspace* $\Phi(\mathsf{A})$ of matrix A and its dimension is called the *nullity* of A. The nullspace of A is spanned by columns of matrix $\mathbf{V}$ corresponding to the zero values w_i. The subspace of $\mathbb{R}^m$ containing vectors which are mapped to by expression (A.3) is called the *range* of A. The range of a matrix A is spanned by columns of matrix U corresponding to the non-zero values w_i. The dimension of the range is called the *rank* r of the matrix and is equal to the number of non-zero values of w_i. Rank plus nullity of a matrix equals n.

- All the vectors $\mathbf{X}$ satisfying (A.3) are given by:

$$\mathbf{X} = \mathsf{A}^+\mathbf{B} + \sum_{i=r+1}^{n} \lambda_i \mathbf{v}^i, \ \lambda_i \in \mathbb{R}, \tag{A.4}$$

where vectors $\mathbf{v}^i$ are the columns of V and λ_i are arbitrary scalar factors. For vectors $\mathbf{B}$ not lying in the range of A the expression (A.4) returns a vector from the range of A minimising the *residual* $\|\mathsf{A}\mathbf{X} - \mathbf{B}\|$. For singular matrices A, the expression $\mathsf{A}^+\mathbf{B}$ returns a vector of the minimal length satisfying (A.3). In homogeneous systems we have $\mathsf{A}^+\mathbf{B} = \mathbf{0}$, thus any linear combination of vectors $\mathbf{v}_i$ spanning the nullspace $\Phi(\mathsf{A})$ satisfies (A.3).

Appendix B

Proof for Singular Configurations of Cameras of Type C

In this section we derive the relative rotation matrices that are singular for cameras of type C (cf. table 4.4), i.e. cameras with known skew factor and principal point. Two sub-cases will be considered: one camera viewing a parallelepiped with only right angles and multiple cameras viewing one parallelepiped with unknown shape.

Let $\mathbf{e}_1 = (1,0,0), \mathbf{e}_2 = (0,1,0)$ and $\mathbf{e}_3 = (0,0,1)$ be the usual three *elementary vectors.* We now prove the following proposition, that will be used below to derive singularities.

Proposition 1 *Let matrices* $\mathsf{A} = diag(a_1, a_2, a_3)$ *and* $\mathsf{B} = diag(b_1, b_2, b_3)$ *be diagonal and matrix* R *be a rotation matrix. There exist matrices* A *and* B *different from* $\mathbf{I_3}$ *satisfying*

$$\mathsf{A} \sim \mathsf{R}^\top \mathsf{B} \mathsf{R} \tag{B.1}$$

in two following cases:

(i) *all columns* $\mathbf{r}_j$ *of* R *are elementary vectors (up to sign), i.e. there exist permutations* (j_1, j_2, j_3) *and* (i_1, i_2, i_3) *of* $(1,2,3)$ *with:*

$$\forall k \in \{1,2,3\} : \mathbf{r}_{j_k} = \pm \mathbf{e}_{i_k}$$

(ii) *a single column* j_1 *of* R *is an elementary vector (up to sign):* $\mathbf{r}_{j_1} = \pm \mathbf{e}_{i_1}$. *In such a case matrices* A *and* B *must satisfy* $a_{j_2} = a_{j_3}$ *and* $b_{i_2} = b_{i_3}$, *where* $\{j_2, j_3\} = \{1,2,3\} \setminus \{j_1\}$ *and* $\{i_2, i_3\} = \{1,2,3\} \setminus \{i_1\}$.

Note that in case (ii) there is no ambiguity in the estimation of values $\frac{a_{j_2}}{a_{j_3}}$ and $\frac{b_{i_2}}{b_{i_3}}$.

Proof: Consider matrices A and B that satisfy equation (B.1), and let k be the scale factor such that: $\mathsf{A} = k\mathsf{R}^\top \mathsf{B}\mathsf{R}$. It follows that A and B are similar and that their spectra are identical up to the scale factor k, i.e. $(a_1, a_2, a_3) = kP(b_1, b_2, b_3)$, where P may denote any permutation [Golub and van Loan, 1989].

Equation (B.1) can be rewritten as:

$$\mathsf{R}\begin{pmatrix} a_1 & & \\ & a_2 & \\ & & a_3 \end{pmatrix} = k\begin{pmatrix} b_1 & & \\ & b_2 & \\ & & b_3 \end{pmatrix}\mathsf{R} \tag{B.2}$$

In the following the elements of matrix R will be denoted as r_{ij}.

"⇒" Let us begin by multiplying equation (B.2), successively, by $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. The following equations are obtained:

$$\forall i, j \in \{1..3\} : (a_j - kb_i)r_{ij} = 0. \tag{B.3}$$

The condition $(a_1, a_2, a_3) = kP(b_1, b_2, b_3)$ can be satisfied in six possible ways, corresponding to the six permutations of elements (b_1, b_2, b_3). In the following let us consider, without loss of generality, that $(a_1, a_2, a_3) = k(b_1, b_2, b_3)$. The reasoning for the remaining permutations can be done analogously.

Let us consider separately the three following cases:

(i) the a_i are mutually different.

According to the assumptions:

$$\begin{aligned} &\left\{\begin{array}{l} (a_1 \neq kb_2) \\ (a_1 \neq kb_3) \\ (a_2 \neq kb_1) \\ (a_2 \neq kb_3) \\ (a_3 \neq kb_1) \\ (a_3 \neq kb_2) \end{array}\right. \Rightarrow \left\{\begin{array}{l} (r_{21} = 0) \\ (r_{31} = 0) \\ (r_{12} = 0) \\ (r_{32} = 0) \\ (r_{13} = 0) \\ (r_{23} = 0) \end{array}\right. \\ &\mathbf{r}_j^\top \mathbf{r}_j = 1 \\ &\det \mathsf{R} = 1. \end{aligned} \tag{B.4}$$

All the conditions (B.4) are satisfied if and only if the matrix R is of the form

$$\mathsf{R} = \begin{pmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & \pm 1 \end{pmatrix}$$

such that $\det \mathsf{R} = 1$. Thus, R is of form (i) given in proposition 1.

Performing the analogous reasoning for the other permutations, it is easy to show that all singular cases for the assumptions made here, are described by case (i) of the proposition.

(ii) $a_1 = a_2 \neq a_3$.

According to the assumptions:

$$\begin{aligned} &\left\{\begin{array}{l} (a_1 \neq kb_3) \\ (a_2 \neq kb_3) \\ (a_3 \neq kb_1) \\ (a_3 \neq kb_2) \end{array}\right. \Rightarrow \left\{\begin{array}{l} (r_{31} = 0) \\ (r_{32} = 0) \\ (r_{13} = 0) \\ (r_{23} = 0) \end{array}\right. \\ &\mathbf{r}_j^\top \mathbf{r}_j = 1 \\ &\det \mathsf{R} = 1. \end{aligned} \tag{B.5}$$

The conditions (B.5) imply that $\mathbf{r}_3 = \pm\mathbf{e}_3$. Together with the assumption $a_1 = a_2$ (which implies $b_1 = b_2$), we identify here the conditions corresponding to case (ii) of the proposition. Again, applying the same reasoning to all possible cases of $a_{j_1} = a_{j_2}$ and all permutation P in $(a_1, a_2, a_3) = kP(b_1, b_2, b_3)$, leads always to case (ii) of the proposition. Note that all matrices R satisfying condition (i) satisfy also condition (ii).

(iii) $a_1 = a_2 = a_3$.

This is equivalent to $\mathsf{A} \sim \mathsf{B} \sim \mathbf{I_3}$, which is excluded in the proposition.

"$\Leftarrow$" Let (i_1, i_2, i_3) and (j_1, j_2, j_3) be two permutations of $(1, 2, 3)$ and the rotation matrix R such that one of its columns is an elementary vector: $\mathbf{r}_{j_1} = \mathbf{e}_{i_1}$. Applying the orthogonality condition on R, its elements r_{ij} must satisfy, for some angle θ:

$$\left\{\begin{array}{rcl} a_{j_1} & = & \pm k b_{i_1} \\ a_{j_2}\cos\theta & = & k b_{i_2}\cos\theta \\ a_{j_2}\sin\theta & = & k b_{i_3}\sin\theta \\ a_{j_3}\sin\theta & = & k b_{i_2}\sin\theta \\ a_{j_3}\cos\theta & = & k b_{i_3}\cos\theta \end{array}\right. \tag{B.6}$$

Equation (B.2) implies that all elements r_{ij} must satisfy $r_{ij}a_j = k r_{ij} b_i$. Developing this expression for $i = i_1, i_2, i_3$ and $j = j_1, j_2, j_3$ and applying conditions (B.7) leads to:

$$\left\{\begin{array}{l} r_{i_1 j_1} = \pm 1 \\ r_{i_1 j_2} = r_{i_1 j_3} = r_{i_2 j_1} = r_{i_3 j_1} = 0 \\ r_{i_2 j_2} = r_{i_3 j_3} = \cos\theta \\ r_{i_3 j_2} = -r_{i_2 j_3} = -\sin\theta \end{array}\right. . \tag{B.7}$$

Conditions (B.6) are satisfied if and only if:

$$\begin{array}{rcl} (\sin\theta = 0 & \wedge & a_j, b_i \in \mathbb{R}) \\ \vee(\cos\theta = 0 & \wedge & a_j, b_i \in \mathbb{R}) \\ \vee(\theta \in \{0..2\pi\} & \wedge & a_{j_2} = a_{j_3} = k b_{i_2} = k b_{i_3}) \end{array}$$

Thus, equation (B.1) is satisfied for the following forms of matrices A, B, R:

1. All columns $\mathbf{r}_j = \mathbf{e}_i$ and the values a_j, b_i are arbitrary. This corresponds to case (i) of the proposition.
2. One column $\mathbf{r}_{j_1} = \mathbf{e}_{i_1}$ and the values $a_{j_2} = a_{j_3} = k b_{i_2} = k b_{i_3}$. Only matrices with two equal diagonal elements satisfy equation (B.1). This means that there is no ambiguity in the computation of ratios of elements $\frac{a_{j_2}}{a_{j_3}}$ and $\frac{b_{i_2}}{b_{i_3}}$. This corresponds to case (ii) of the proposition.

■

One Parallelepiped Seen by One Camera

In the following, the rotation matrices that are singular for case C-3-0 (cf. section 4.5) will be derived. Case C-3-0 corresponds to a known principal point and skew and parallelepiped with three right angles, i.e. ω'' and μ'' are diagonal matrices: $\omega'' = \mathrm{diag}(b_1, b_2, b_3)$ and $\mu'' = \mathrm{diag}(a_1, a_2, a_3)$.

According to the definition of singularity given in section 4.5, proposition 1 gives us directly the singular relative rotations R:

(i) $\mathsf{R} = (\pm\mathbf{e}_{i_1}, \pm\mathbf{e}_{i_2}, \pm\mathbf{e}_{i_3})$, where (i_1, i_2, i_3) is a permutation of $(1, 2, 3)$ and such that $\det \mathsf{R} = +1$. In that case, none of the elements of ω'' and μ'' can be estimated, i.e. calibration fails completely. This case corresponds to configurations where each edge of the parallelepiped is parallel to one of the three camera axes.

(ii) One of the columns of R is an elementary vector. This corresponds to the case when one camera axis is parallel to one axis of the parallelepiped. In that case, although calibration fails overall, some individual parameters may indeed be estimated successfully. According to the results of proposition (1) and to the form of matrices ω and μ, we can summarize this in the following table. For example, if $\mathbf{r}_2 = \pm\mathbf{e}_3$, the aspect ratio τ as well as the length ratio $\frac{l_1}{l_3}$ of parallelepiped edges, can be estimated.

$=\pm$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$
$\mathbf{r}_1$	$\frac{l_2}{l_3}, \alpha_v$	$\frac{l_2}{l_3}, \alpha_u$	$\frac{l_2}{l_3}, \tau$
$\mathbf{r}_2$	$\frac{l_1}{l_3}, \alpha_v$	$\frac{l_1}{l_3}, \alpha_u$	$\frac{l_1}{l_3}, \tau$
$\mathbf{r}_3$	$\frac{l_1}{l_2}, \alpha_v$	$\frac{l_1}{l_2}, \alpha_u$	$\frac{l_1}{l_2}, \tau$

One Parallelepiped Seen in Two Cameras

As mentioned previously, due to the observation of a parallelepiped, we know affine scene structure, i.e. the plane at infinity in a projective reconstruction. We can thus apply the appropriate results of [Kahl, 1999] on singularities for general self-calibration. We will show that self-calibration of cameras with known principal point and zero skew is singular for the same type of rotations as the calibration in case C-3-0 described above. Let the true intrinsic parameters of the cameras be represented by diagonal matrices ω_1 and ω_2 and the estimated matrices be represented by diagonal matrices ω'_1, ω'_2. The coordinate system is defined such that the rotation of the first camera is $\mathsf{R}_1 = \mathbf{I_3}$ and the rotation of the second camera is R_2. Let C_f denote a 3×3 matrix corresponding to a false absolute conic on the plane at infinity. The image of a conic lying on the plane at infinity is [Kahl, 1999] represented by

$$\omega'_i \sim \mathsf{K}_i{}^{-\mathsf{T}} \mathsf{R}_i \mathsf{C}_f \mathsf{R}_i^\top \mathsf{K}_i{}^{\mathsf{T}}, \tag{B.8}$$

Let us consider matrices $\omega''_i = \mathsf{K}_i{}^{\mathsf{T}} \omega'_i \mathsf{K}_i{}^{-\mathsf{T}}$. Then:

$$\omega''_i \sim \mathsf{R}_i \mathsf{C}_f \mathsf{R}_i^\top. \tag{B.9}$$

Matrices ω''_i must be diagonal. Let us represent them as $\omega'_1 \sim \operatorname{diag}(a_1, a_2, a_3)$ and $\omega''_2 \sim \operatorname{diag}(b_1, b_2, b_3)$. Using the equation (B.9) and due to $\mathsf{R}_1 = \mathbf{I_3}$ we have $\mathsf{C}_f \sim \operatorname{diag}(a_1, a_2, a_3)$. Equation (B.9) aplied to the second camera is $\omega''_2 \sim \mathsf{R}_2 \mathsf{C}_f \mathsf{R}_2^\top$. The rotation is singular when there exist matrices C_f and ω''_2 different from the identity that satisfy equation (B.9). According to proposition 1 the following cases are singular:

(i) $\mathsf{R}_2 = (\pm\mathbf{e}_{i_1}, \pm\mathbf{e}_{i_2}, \pm\mathbf{e}_{i_3})$, where (i_1, i_2, i_3) is a permutation of $(1, 2, 3)$ and $\det \mathsf{R}_2 = +1$. No intrinsic parameter can be estimated. This corresponds to cases where each axis of the first camera is parallel to some axis of the second camera (not necessarily to the corresponding axis).

(ii) One column of R_2 is an elementary vector. This corresponds to cases where one axis of the first camera is parallel to one axis of the second camera. Using the results of proposition 1 and the form of matrices ω_1 and ω_2, in the following special cases some individual intrinsic parameters can be estimated:

$= \pm$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$
$\mathbf{r}_1$	$\alpha_{v,1}, \alpha_{v,2}$	$\alpha_{v,1}, \alpha_{u,2}$	$\alpha_{v,1}, \tau_2$
$\mathbf{r}_2$	$\alpha_{u,1}, \alpha_{v,2}$	$\alpha_{u,1}, \alpha_{u,2}$	$\alpha_{u,1}, \tau_2$
$\mathbf{r}_3$	$\tau_1, \alpha_{v,2}$	$\tau_1, \alpha_{u,2}$	τ_1, τ_2

Appendix C

Implementation of r-Methods

The current implementation of the system contains a dictionary including about sixty methods. It includes r-methods that solve constraints by computing (output) coordinates of *one* output object. Note that certain r-methods compute only a subset of the coordinates in the output object (for example only a normal vector of a plane).

How to Not Forget Any Possible r-Method

Our dictionary contains all the r-methods which compute coordinates of a single object. To build all the r-methods in an exhaustive way, we have proceeded as follows. In order to compute a finite number of solutions, the equation system satisfied by an r-method must be well-constrained. A necessary condition is that the r-method is "square", that is the number of equations (internal or not) must be equal to the number of output variables. This rule has been used to design methodically the r-methods included in our dictionary. Based on the allowed set of geometric constraints, it is easy to find all the combinations of constraints whose number of degrees of freedom sums up to the one of the output object. However, this simple count is not a sufficient condition to ensure that the corresponding system is well-constrained. As an example, let us consider a plane as output object. Both point-plane incidence constraint and plane-plane orthogonality constraint fix 1 degree of freedom of a plane. However, the orthogonality constraint is related only to plane coordinates corresponding to the plane normal. It is thus not possible to compute a plane position using 3 orthogonality constraints. On the opposite, a plane-point incidence constraint relates all the coordinates belonging to both objects so that 3 plane-point incidence constraints can determine a plane.

A more robust methodology is based on graph properties. Without going into details, a well-constrained equation system corresponding to a "correct" r-method has a perfect matching, that is a maximum matching including all the output variables and all the equations (see [Pothen and Chin-Fan, 1990]).

Design of r-Methods

Linear r-method routines use linear algebra. To build fast non-linear r-method execution procedures able to yield all the solutions, we made symbolic manipulations of the equations involved in r-methods.

An example of an r-method solving 3 quadratic distance equations is sketched in figure 8.6–(b). A point is computed using distance constraints from 3 other points. Using the known coordinates of 3 points (as input variables), the equations on coordinates (x, y, z) of the 4^{th} point are:

$$\begin{aligned}
F_1 &:= x^2 + y^2 + z^2 + c_{11}x + c_{12}y + c_{13}z + c_{14};\\
F_2 &:= x^2 + y^2 + z^2 + c_{21}x + c_{22}y + c_{23}z + c_{24};\\
F_3 &:= x^2 + y^2 + z^2 + c_{31}x + c_{32}y + c_{33}z + c_{34};
\end{aligned}$$

Solving 3 quadratic equations is a very difficult problem for automatic symbolic methods. However using the assignment $G_1 = F_1 - F_2$ and $G_2 = F_1 - F_3$ allows expressing two of the variables (x, y, z) in terms of the third one using linear equations. Thus, using one of equations F_i, two solutions satisfying the whole system are obtained. Finally, the r-method execution procedure will result in a sequence of fast atomic steps: evaluations of polynomial terms and solving of equations of the form: $a\,x^2 + b\,x + c = 0$ (giving the at most two possible solutions over the real numbers).

Only Well-Conditioned r-Methods Are Created in the Equation Graph

Certain r-methods are symetric in a sense that output and input variables are interchangeable. For example, when a point $P(x, y, z)$ is constrained only by a point-line incidence constraint (fixing 2 dofs), 3 symetric r-methods could be applied: any of the coordinates x, y, or z are used as input variable (together with the coordinates of the line), and the values of the two other variables are computed by the r-method. Among the 3 r-methods that could be built, we create the best-conditioned one (according to the values of input variables given by the initialization phase). Indeed, depending on the initial line position, the choice of the input coordinate influences the numerical properties of the linear system used to compute the two (output) point coordinates, as illustrated in figure C.1. Using initial values of input variables, these properties can be analysed using SVD.

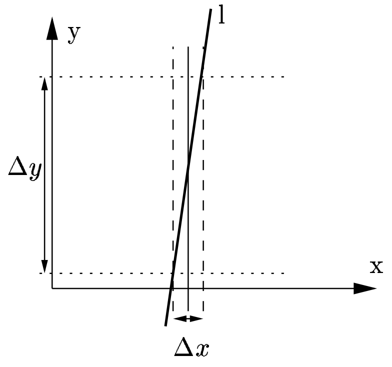


Figure C.1: Well-conditioned and bad conditioned computations. Based on the represented line equation, the computation of the coordinate y using values of x is much more sensitive to noise than the computation of coordinate x using values of y.

Bibliography

S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphic*, 1993.

M. Armstrong, A. Zisserman, and P. Beardsley. Euclidean structure from uncalibrated images. In E. Hancock, editor, *Proceedings of the fifth British Machine Vision Conference, York, England*, volume 2, pages 509–518, September 1994.

M. Armstrong, A. Zisserman, and R. Hartley. Self-calibration from image triplets. In B. Buxton and R. Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1064 of *Lecture Notes in Computer Science*, pages 3–16. Springer-Verlag, April 1996.

S. Avidan and A. Shashua. Threading fundamental matrices. *Lecture Notes in Computer Science*, 1406:124–140, 1998.

D. Avis and K. Fukuda. Reverse Search Enumeration. *Discrete Applied Mathematics*, 6:21–46, 1996.

A. Bartoli and P. Sturm. Constrained structure and motion from N views of a piecewise planar scene. In *Proceedings of the First International Symposium on Virtual and Augmented Architecture, VAA'01, Dublin, Ireland*, pages 195–206, June 2001.

P.-L. Bazin. A parametric scene reduction algorithm from geometric relations. In *Proceedings of Vision Geometry IX, SPIE's 45th annual meeting, San Diego*, June 2000.

P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In B. Buxton and R. Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1065 of *Lecture Notes in Computer Science*, pages 683–695. Springer-Verlag, April 1996.

A. Bjorck. *Numerical Methods For Least Squares Problems*. SIAM, 1990.

D. Bondyfalat and S. Bougnoux. Imposing euclidean constraints during self-calibration processes. In R. Koch and L. Van Gool, editors, *Proceedings of the SMILE Workshop on 3D Structure from Multiple Images of Large-Scale Environments, Freiburg, Germany*, volume 1506 of *Lecture Notes in Computer Science*, pages 224–235. Springer-Verlag, June 1998.

D. Bondyfalat, T. Papadopoulo, and B. Mourrain. Using scene constraints during the calibration procedure. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 124–130, July 1999.

A. Borning. THINGLAB*: A Constraint–Oriented Simulation Laboratory*. PhD thesis, Stanford University, 1979.

A. Brunn, F. Lang, E. Gülch, and W. Förstner. A hybrid concept for 3d building acquisition. ISPRS *Journal of Photogrammetry and Remote Sensing*, 53(2):119–129, April 1998.

T. Buchanan. The twisted cubic and camera calibration. *Computer Vision, Graphics and Image Processing*, 42(1):130–132, April 1988.

B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4:127–140, 1990.

S. Carlsson. Multiple image invariants using the double algebra. In J.L. Mundy and A. Zissermann, editors, *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Azores, Portugal*, pages 335–350, October 1993.

C.S. Chen, C.G. Yu, and Y.P. Hung. New calibration-free approach for augmented reality based on parameterized cuboid structure. *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, 4:30–37, 1999.

R. Cipolla and E. Boyer. 3d model acquisition from uncalibrated images. In *Proceedings of IAPR Workshop on Computer Vision, Chiba, Japan*, pages 559–568, November 1998.

S.C. Cornou, M. Dhome, and P. Sayd. Architectural reconstruction with multiple views and geometric constraints. In *Proceedings of the 14th British Machine Vision Conference, Norwich, England*, pages 739–749, September 2003.

J.M. Coughlan and A.L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 941–947, June 1999.

A. Criminisi, I. D. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

D.Bondyfalat, B. Mourrain, and T. Papadopoulo. An application of automatic theorem proving in computer vision. In *Automated Deduction in Geometry*, pages 207–231, 1999.

L. de Agapito, R.I. Hartley, and E. Hayman. Linear self-calibration of a rotating and zooming camera. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA*, 1999.

P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry-and image-based approach. In SIGGRAPH '96*, New Orleans*, August 1996.

S. Dedieu, P. Guitton, C. Schlick, and P. Reuter. Reality: An interactive reconstructiuon tool of 3d objects from photographs. In *Proceedings of the 6th International Fall Workshop VISION, MODELING, AND VISUALIZATION, Stuttgart, Germany*, pages 195–202, November 2001.

A.R. Dick, P.H.S. Torr, S.F. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, 2001.

S. Ducasse, M. Blay-Fornarino, and A.M. Pinna-Déry. A reflective model for first class dependencies. In *Tenth Annual Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, pages 265–280, October 1995.

A.L. Dulmage and N.S. Mendelsohn. Covering of bipartite graphs. *Canad. J. Math.*, 10:517–534, 1958.

O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 563–578. Springer-Verlag, May 1992.

O. Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint.* Artificial intelligence. The MIT Press, Cambridge, MA, USA, Cambridge, MA, 1993.

O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. Technical Report 2665, INRIA, October 1995.

O. Faugeras, L. Quan, and P. Sturm. Self-calibration of a 1d projective camera and its application to the self-calibration of a 2d projective camera. In *Proceedings of the 5th European Conference on Computer Vision, Freiburg, Germany*, pages 36–52, June 1998.

O.D. Faugeras, Q.T. Luong, and S.J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 321–334. Springer-Verlag, May 1992.

O.D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida, USA*, pages 15–20, June 1986.

M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24 (6):381 – 395, June 1981.

A.W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, pages 311–326, june 1998.

W. Förstner, A. Brunn, and S. Heuel. Statistically testing uncertain geometric relations. In G. Sommer, N. Krüger, and Ch. Perwass, editors, *Mustererkennung 2000*, pages 17–26. Springer, September 2000.

B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, January 1990.

I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, 1997.

M. Gangnet and B. Rosenberg. Constraint programming and graph algorithms. In *Second International Symposium on Artificial Intelligence and Mathematics*, 1992.

P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

G.H. Golub and C.F. van Loan. *Matrix Computation.* The Johns Hopkins University Press, Baltimore, 1989.

G. Gracie. Analytical photogrammetry applied to single terrestrial photograph mensuration. In *Proceedings of the XIth International Congress of Photogrammetry, Lausanne, Switzerland*, July 1968.

E. Grossmann. *Maximum Likelihood 3D Reconstruction From One or More Uncalibrated Views Under Geometric Constraints.* Ph.d. thesis, Universidade Técnica de Lisboa, Portugal, 2002.

E. Grossmann and J. Santos-Victor. Dual representations for vision-based 3d reconstruction. In *The Eleventh British Machine Vision Conference, University of Bristol, UK*, pages 516–526, 2000.

E. Grossmann and J. Santos-Victor. Single and multi-view reconstruction of structured scenes. In *Proceedings of the Fifth Asian Conference on Computer Vision, Melbourne, Australia*, pages 93–104, January 2002.

P. Gurdjos and R. Payrissat. Plane-based calibration of a camera with varying focal length: the centre line constraint. In *Proceedings of the 12th British Machine Vision Conference, Manchester, UK*, pages 623–632, September 2001.

P. Gurdjos and P. Sturm. Methods and geometry for plane-based self-calibration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, pages 491–496, 2003.

R.I. Hartley. Euclidean reconstruction from uncalibrated views. In *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Azores, Portugal*, pages 187–202, October 1993.

R.I. Hartley. Self-calibration from multiple views with a rotating camera. In *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, pages 471–478. Springer-Verlag, May 1994.

R.I. Hartley. A linear method for reconstruction from lines and points. In E. Grimson, editor, *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 882–887. IEEE Computer Society Press, June 1995.

R.I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997a.

R.I. Hartley. Self-calibration of stationary cameras. *International Journal of Computer Vision*, 22 (1):5–23, 1997b.

R.I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Urbana-Champaign, Illinois, USA*, pages 761–764, 1992.

R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000.

O. Henricsson, A. Streilein, and A. Gruen. Automated 3-D reconstruction of buildings and visualization of city models. In *Proceedings of the Workshop on 3D City Models, Bonn, Germany*, October 1996.

S. Heuel. Points, lines and planes and their optimal estimation. In *Pattern Recognition, 23rd DAGM Symposium*, number 2191 in LNCS, pages 92–99. Springer, September 2001.

A. Heyden and K. Åström. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 438–443. IEEE Computer Society Press, June 1997.

C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming CP'97*, pages 463–477, 1997.

Y. Horry, K.I. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 225–232. ACM Press/Addison-Wesley Publishing Co., 1997. ISBN 0-89791-896-7.

D. Jacobs. Linear fitting with missing data: Applications to structure-from-motion and to characterizing intensity images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 206–212. IEEE Computer Society Press, June 1997.

D. Jelinek and C.J. Taylor. Reconstruction of linearly parameterized models from single images with a camera of unknown focal length. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 23(7):767–774, July 2000.

C. Jermann, B. Neveu, and G. Trombettoni. Algorithms for Identifying Rigid Subsystems in Geometric Constraint Systems. In *Proc.of IJCAI'03, International Joint Conference on Artificial Intelligence*, pages 233–238, 2003.

B. Johansson. View synthesis and 3D reconstruction of piecewise planar scenes using intersection lines between the planes. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 54–59, Kerkyra, Greece, September 1999. IEEE press.

F. Kahl. Critical motions and ambiguous euclidean reconstructions in auto-calibration. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, volume 2, pages 469–475, 1999.

F. Kahl, B. Triggs, and K. Åström. Critical motions for auto-calibration when some intrinsic parameters can vary. *Journal of Mathematical Imaging and Vision*, 13(2):131–146, October 2000.

K. Kanatani. *Statistical Optimisation for Geometric Computation: Theory and Pra*. Elsevier Science, 1996.

R. Kaucic, R.I. Hartley, and N.Y. Dano. Plane-based projective reconstruction. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 420–427, 2001.

J. Kosecka and W. Zhang. Video compass. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 476–491, May 2002.

G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.

R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: a parallax based approach. In *Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel*, pages 685–688, 1994.

G. Kwaiter and V. Gaildrat. Modelling with constraints : A bibliographical survey. In *International Conference on Information Visualisation, IV'98*, pages 211–219, 1998.

E. Lahaye. Une méthode de résolution d'une catégorie d'équations transcendantes. *Compte-rendu des Séances de L'Académie des Sciences*, 198:1840–1842, 1934.

H. Lamure and D. Michelucci. Qualitative study of geometric constraints. In Beat Brüderlin and Dieter Roller, editors, *Workshop on Geometric Constraint Solving and Applications*, pages 134–145, Technical University of Ilmenau, Germany, 1997.

O. Lhomme, P. Kuzo, and P. Mace. Desargues : a constraint-based system for 3d projective geometry. In *Workshop on Geometric Constraint Solving and Applications. Ilmenau, Allemagne*, September 1997.

D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, USA*, pages 482–488, June 1998.

D. Liebowitz and A. Zisserman. Combining scene and auto-calibration constraints. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, September 1999.

Q.T. Luong and O.D. Faugeras. Determining the fundamental matrix with planes: Instability and new algorithms. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, New York, USA*, pages 489–494, June 1993.

E. Malis and R. Cipolla. Camera self-calibration from unknown planar structures enforcing the multi-view constraints between collineations. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 4(9), September 2002.

D. Martinec and T. Padjla. Structure from many perspective images with occlusions. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 355–369, 2002.

D. Martinec and T. Pajdla. Structure from many perspective images with occlusions. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 355–369, May 2002.

S.J. Maybank and O.D. Faugeras. A theory of self calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

C. McGlone. Bundle adjustment with object space geometric constraints for site modeling. In D.M. McKeown and Dowman, editors, *Proceedings of the* SPIE *Conference on Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, Orlando, Florida, USA*, volume 2486, pages 25–36, April 1995.

P. McLauchlan, X. Shen, A. Manessis, P. Palmer, and A. Hilton. Surface-based structure-from-motion using feature groupings. In *Proceedings of the Fourth Asian Conference on Computer Vision*, pages 699–705, 2000.

T. Moons, L. Van Gool, M. van Diest, and E. Pauwels. Affine reconstruction from perspective image pairs. In *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Azores, Portugal*, pages 249–266, October 1993.

D.D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, India*, pages 696–702, January 1998.

N. Navab, Y. Genc, and M. Appel. Lines in one orthographic and two perspective views. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 25(7):912–917, July 2003.

J. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the* 1^st *ACM Symposium on Solid Modeling and CAD/CAM Applications*, pages 397–407. ACM Press, 1991.

C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

M. Pollefeys. Tutorial on 3d modelling from images. In *http://www.esat.kuleuven.ac.be/ pollefey/tutorial/*, 2000.

M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, India*, pages 90–95, January 1998.

M. Pollefeys and L. Van Gool. A stratified approach to metric self-calibration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 407–412. IEEE Computer Society Press, June 1997.

M. Pollefeys, L. Van Gool, and M. Proesmans. Euclidean 3D reconstruction from image sequences with variable focal lengths. In B. Buxton and R. Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1064 of *Lecture Notes in Computer Science*, pages 31–42. Springer-Verlag, April 1996.

A. Pothen and J. Chin-Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software*, 16(4):303–324, 1990.

P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Proceedings of Eurographics Workshop on Rendering 98*, pages 93–104, June 1998.

W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

L. Quan and T. Kanade. A factorization method for affine structure from line correspondences. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, San Francisco, California, USA*, pages 803–808, 1996.

I.D. Reid and D.W. Murray. Active tracking of foveated feature clusters using affine structure. *International Journal of Computer Vision*, 18(1):41–60, April 1996.

J.C. Régin. *Développement d'outils algorithmiques pour l'Intelligence Artificielle. Application à la chimie organique.* PhD thesis, LIRMM, Montpellier, France, 1995.

D.P. Robertson and R. Cipolla. An interactive system for cosntraint-based modelling. In *The Eleventh British Machine Vision Conference, University of Bristol, UK*, pages 536–545, September 2000.

D.P. Robertson and R. Cipolla. Building architectural models from many views using map constraints. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 155–169, May 2002.

C. Rother. Linear multi-view reconstruction of points, lines, planes and cameras. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 1210–1217, 2003a.

C. Rother. *Multi-view reconstruction and camera recovery using a real or virtual reference plane.* PhD thesis, KTH, Stockholm, Sweden, 2003b.

C. Rother and S. Carlsson. Linear multi view reconstruction and camera recovery. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 42–49, July 2001.

C. Rother and S. Carlsson. Linear multi view reconstruction and camera recovery using a reference plane. *International Journal of Computer Vision*, 9(49(2/3)):117–141, 2002.

C. Rother, S. Carlsson, and D. Tell. Projective factorization of planes and cameras in multiple views. In *Proceedings of the 15th International Conference on Pattern Recognition, Quebec, Canada*, pages 737–740, 2002.

R.H. Schrand and R. Seidl. *Canoma Visual Insight*. Coriolis Value, 2000. URL `http://www.metacreations.com/products/canoma/`.

J.G. Semple and G.T. Kneebone. *Algebraic Projective Geometry*. Oxford Science Publication, 1952.

H.-Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, USA*, pages 427–433, June 1998.

S. Sorlin and C. Solnon. A global constraint for graph isomorphism problems. In *Proc of the first conference CP-AI-OR'2004*, Nice, France, 2004.

A. Sosnov. *Modélisation Géométrique par Séparation de Contraintes*. Thèse de doctorat, Université de Nantes, 2003.

A. Sosnov and P. Macé. Rapid algebraic resolution of 3d geometric contraints and control of their consistency. In *Fourth International Workshop on Automated Deduction in Geometry (ADG'02)*, 2002.

G. Sparr. An algebraic/analytic method for reconstruction from image correspondences. In *Proceedings of the 7th Scandinavian Conference on Image Analysis, Aalborg, Denmark*, pages 274–281, 1991a.

G. Sparr. Projective invariants for affine shapes of point configurations. In *Proceeding of the* DARPA–ESPRIT *workshop on Applications of Invariants in Computer Vision, Reykjavik, Iceland*, pages 151–170, March 1991b.

G. Sparr. Depth computations from polyhedral images. In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 378–386. Springer-Verlag, May 1992a.

G. Sparr. On the "reconstruction" of impossible objects. In *Proceedings Swedish Society for Automated Image Analysis, Uppsala, Sweden*, pages 109–112, 1992b.

G. Sparr. Simultaneous reconstruction of scene structure and camera locations from uncalibrated image sequences. In *Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria*, volume I, pages 328–333. IEEE Computer Society Press, August 1996.

G. Sparr. Euclidean and affine structure/motion for uncalibrated cameras from affine shape and subsidiary information. In *SMILE*, pages 187–207, 1998.

J. Stolfi. *Oriented Projective Geometry*. Academic Press, 1991.

P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 1100–1105, June 1997.

P. Sturm. Algorithms for plane-based pose estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA*, pages 1010–1017, June 2000.

P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado, USA*, pages 432–437, June 1999a.

P. Sturm and S.J. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In T. Pridmore and D. Elliman, editors, *Proceedings of the tenth British Machine Vision Conference, Nottingham, England*, pages 265–274. British Machine Vision Association, September 1999b.

P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In B. Buxton and R. Cipolla, editors, *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, volume 1065 of *Lecture Notes in Computer Science*, pages 709–720. Springer-Verlag, April 1996.

Z. Sun, A. M. Tekalp, N. Navab, and V. Ramesh. Short papers - shape extraction and analysis - interactive optimization of 3d shape and 2d correspondence using multiple geometric constraints via pocs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):562–568, 2002. ISSN 0162-8828.

G. Sunde. Specification of shapes by dimensions and other geometric constraints. In *IFIP WG 5.2 Geometric Modeling*, 1986.

I. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System.* PhD thesis, Department of Electrical Engineering, MIT, 1963.

D. Svedberg and S. Carlsson. Calibration, pose and novel views from single images of constrained scenes. *Pattern Recognition Letters*, 21(13–14):1125–1133, December 2000.

R. Szeliski and P.H.S. Torr. Geometrically constrained structure from motion : Points on planes. In *3D Structure from Multiple Images of Large-scale Environments SMILE'98*. Springer Verlag, June 1998.

C.J. Taylor. Reconstruction of articulated objects from point correspondences in a single image. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA*, pages 677–684, June 2000.

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

B. Triggs. Matching constraints and the joint image. In E. Grimson, editor, *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 338–343. IEEE Computer Society Press, June 1995.

B. Triggs. Factorization methods for projective structure and motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, San Francisco, California, USA*, pages 845–851, 1996.

B. Triggs. Autocalibration and the absolute quadric. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 609–614. IEEE Computer Society Press, June 1997.

B. Triggs. Autocalibration from planar scenes. In *Proceedings of the 5th European Conference on Computer Vision, Freiburg, Germany*, 1998.

B. Triggs. Plane + parallax, tensors and factorization. In *Proceedings of the 6th European Conference on Computer Vision, Dublin, Ireland*, pages 522–538. Springer-Verlag, 2000.

G. Trombettoni. *Algorithmes de maintien de solution par propagation locale pour les systèmes de contraintes.* Thèse de doctorat, Université de Nice–Sophia Antipolis, 1997.

G. Trombettoni. A Polynomial Time Local Propagation Algorithm for General Dataflow Constraint Problems. In *Proc. Constraint Programming CP'98, LNCS 1520 (Springer Verlag)*, pages 432–446, 1998.

G. Trombettoni and M. Wilczkowiak. Scene Reconstruction based on Constraints: Details on the Equation System Decomposition. In *Proc. International Conference on Constraint Programming, CP'03*, volume 2833 of *LNCS*, pages 956–961, Kinsale,Ireland, 2003. Springer.

R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida, USA*, pages 364–374, 1986.

J.R. Ullman. An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

F.A. van den Heuvel. 3D reconstruction from a single image using geometric constraints. ISPRS *Journal of Photogrammetry and Remote Sensing*, 53:354–368, 1998.

F.A. van den Heuvel and G. Vosselman. Efficient 3D-modeling of buildings using a priori geometric object information. In *Proceedings of the* SPIE *Conference on Videometrics V*, volume 3174, pages 38–49, 1997.

P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica : A Modeling Language for Global Optimization.* MIT Press, 1997.

S. Van Huffel and J. Vanderwalle. The total least squares problem: Computational aspects and analysis. In *Frontiers in Applied Mathematics*, volume 9. SIAM, 1991.

A. Verroust, F. Schonek, and D. Roller. Rule oriented method for parametrized computer aided design. *Computer Aided Design*, 24(6):531–540, 1992.

VXL. http://vxl.sourceforge.net/, 2003.

T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 541–555, May 2002.

M. Wilczkowiak, E. Boyer, and P. Sturm. Camera calibration and 3D reconstruction from single images using parallelepipeds. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, volume 1, pages 142–148. IEEE Computer Society Press, July 2001.

M. Wilczkowiak, E. Boyer, and P. Sturm. 3D modelling using geometric constraints: A parallelepiped based approach. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 2353 of *Lecture Notes in Computer Science*, pages 221–236. Springer-Verlag, May 2002.

M. Wilczkowiak, P. Sturm, and E. Boyer. The analysis of ambiguous solutions in linear systems and its application to computer vision. In *Proceedings of the 14th British Machine Vision Conference, Norwich, England*, pages 53–62, September 2003a.

M. Wilczkowiak, P. Sturm, and E. Boyer. Modélisation 3d à l'aide de contraintes géométriques. In *Journées ORASIS 2003, Gérardmer, France*, pages 77–86, May 2003b.

M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and 3d modelling. Research Report 5055, INRIA, Grenoble, France, 2003c.

M. Wilczkowiak, G. Trombettoni, C. Jermann, P. Sturm, and E. Boyer. Scene modeling based on constraint system decomposition techniques. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 1004–1010, October 2003d.

G. Xu, J.-I. Terai, and H.-Y. Shum. A linear algorithm for camera self-calibration, motion and structure recovery for multi-planar scenes from two perspective images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA*, June 2000.

L. Zelnik-Manor and M. Irani. Multi-view constraints on homographies. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 24(2):214–223, February 2002.

Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, September 1999.

Z. Zhang, P. Anandan, and H.Shum. What can be determined from a full and a weak perspective image? In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 680–688, 1999.

A. Zisserman, D. Liebowitz, and M. Armstrong. Resolving ambiguities in auto-calibration. *Philosophical Transactions of the Royal Society of London, SERIES A*, 356(1740):1193–1211, 1998.