UNIVERSITEIT • STELLENBOSCH • UNIVERSITY

Metric reconstruction of multiple rigid objects

by

Jan Hendrik de Vaal

Thesis presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

Department of Mathematical Sciences (Applied Mathematics)
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Prof B.M. Herbst

March 2009

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: ....
J.H. de Vaal

Date: ....

Abstract

Metric reconstruction of multiple rigid objects

J.H. de Vaal

Department of Mathematical Sciences (Applied Mathematics)
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Applied Mathematics)

March 2009

Engineers struggle to replicate the capabilities of the sophisticated human visual system. This thesis sets out to recover motion and 3D structure of multiple rigid objects up to a similarity. The motion of these objects are either recorded in a single video sequence, or images of the objects are recorded on multiple, different cameras. We assume a perspective camera model with optional provision for calibration information. The Structure from Motion (SfM) problem is addressed from a matrix factorization point of view. This leads to a reconstruction correct up to a projectivity — of little use in itself. Using techniques from camera auto-calibration the projectivity is upgraded to a similarity. This reconstruction is also applied to multiple objects through motion segmentation. The SfM system developed in this thesis is a batch-processing algorithm, requiring few frames for a solution and readily accepts images from very different viewpoints. Since a solution can be obtained with just a few frames, it can be used to initialize sequential methods with slower convergence rates, such as the Kalman filter. The SfM system is critically evaluated against an extensive set of motion sequences.

Uittreksel

Metric reconstruction of multiple rigid objects

J.H. de Vaal

Department of Mathematical Sciences (Applied Mathematics)
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Tesis: MScEng (Applied Mathematics)

Maart 2009

Ingeneurs worstel om die kapasiteit van die gesofistikeerde menslike visuele stelsel te repliseer. Die doelstelling van hierdie tesis is om die beweging en 3D struktuur van veelvoudige voorwerpe tot op 'n similariteit te rekonstrueer. Die beweging van hierdie voorwerpe word of in 'n enkele videoreeks opgeneem, of beelde van die voorwerpe word opgeneem op verskeie verskillende kameras. Ons aanvaar 'n perspektief kamera model met optionele voorsiening vir kalibrasie informasie. Die Struktuur vanuit Beweging (SvB) probleem word aangespreek vanuit 'n matriks faktoriseerings oogpunt. Dit lei tot 'n rekonstruksie korrek tot op 'n projektiwiteit — van min waarde op sy eie. Deur gebruik te maak van kamera auto-kalibrasie tegnieke word die projektiwiteit opgradeer na 'n similariteit. Hierdie rekonstruksie word ook op veelvoudige voorwerpe toegepas deur middel van beweging segmentasie. Die SvB stelsel wat ontwikkel is in hierdie tesis is 'n proseseerings algoritme wat min rame benodig vir 'n oplossing en aanvaar beelde van uit verskillende oogpunte. Aangesien 'n oplossing verkrygbaar is van slegs 'n paar rame, kan dit ook gebruik word om sekwensiele metodes met stadiger konvergensie te initialiseer, soos die Kalman filter. Die SvB stelsel word krities geëvalueer teen 'n omvattende stel bewegingsekwensies.

Acknowledgments

I would like to express my sincere gratitude to all the people and organizations who have contributed to making this work possible. I especially want to thank my study leader Prof Herbst for his guidance, ideas, motivation and patience. I also must mention my family, the National Research Foundation and TradeCube who all contributed to the funding of my post-graduate studies.

Dedications

to Mersini

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
CCS	Camera Coordinate System
CMS	Critical Motion Sequence
COBYLA	Constrained Optimization BY Linear Approximation
COP	Center Of Projection
DIAC	Dual Image of the Absolute Conic
EKF	extended Kalman filter
GUI	Graphical User Interface
IAC	Image of the Absolute Conic
LK	Lucas Kanade
LP	Linear Programming
OCS	Object Coordinate System
PAC	Potential Absolute Conic
SfM	Structure from Motion
SVD	Singular Value Decomposition
UKF	unscented Kalman filter
WCS	World Coordinate System

List of Symbols

Constants:

$\mathbf{0}$	vector with all components zero
$\mathbf{e}_4$	vector $[0, 0, 0, 1]^T$
I	identity matrix
$\widetilde{I}$	4×4 matrix derived from the identity matrix with $\widetilde{I}_{44} = 0$

Notational Conventions:

a	scalar value a
$\mathbf{a}$	homogeneous column vector $\mathbf{a}$
$\\|\mathbf{a}\\|$	L2-norm of vector $\mathbf{a}$
$\widetilde{\mathbf{a}}$	non-homogeneous column vector $\widetilde{\mathbf{a}}$
A	matrix A
$\\|A\\|$	Frobenius norm of matrix A
$\mathbf{x}$	2D features in the image coordinate system
$\mathbf{X}$	3D features in the world coordinate system
$\boldsymbol{l}$	2D line
$\mathbf{\Pi}$	plane
ω	conic
Q	quadric
R	rotation matrix
$\mathbf{t}$	translation vector
H	homography
P	projection matrix

Special

$\mathbb{R}^n$	n-dimensional Euclidean space
$\mathbb{P}^n$	n-dimensional projective space
$\mathbf{C}$	camera center
$\mathbf{O}$	object coordinate system (OCS) center
N	total number of features
F	total number of frames
K	camera calibration matrix
$\mathbf{\Pi}_\infty$	plane at infinity
ω_∞	image of the absolute conic (IAC)
ω^*_∞	dual image of the absolute conic (DIAC)
Ω_∞	absolute conic
Q^*_∞	absolute dual quadric
W	measurement matrix
M	motion matrix
S	structure matrix
$\mathcal{F}$	fundamental matrix
λ	projection depth
ϕ	potential IAC
Φ	potential absolute conic (PAC)
O	total number of motions
G	shape interaction matrix

Chapter 1

Introduction

The human visual system is a feat of biological engineering, equipped with high-resolution, high-definition colour stereo cameras and a sophisticated processing unit. It is able to perform the 3D reconstruction of multiple objects, processing lighting, detail, perspective, structure and motion, literally within the blink of an eye. We can fool the eye and the mind with optical illusions, but still we struggle to mimic the 3D reconstruction capabilities of this incredible system.

Engineers have been able to design a number of similar systems, including

- *Stereo vision* that reconstructs an object from two images obtained from two differently positioned, calibrated cameras.
- *Structure from Motion* (SfM) that reconstructs an object from its motion observed in multiple images, the images obtained either as a video sequence from a single camera or from multiple cameras observing the object. The cameras need not be calibrated.
- *Shape from Shading* that recovers the 3D shape of a surface from brightness values observed in a single black and white image.
- *Shape from Silhouette* that reconstructs an object as the smallest volume obtained from intersecting multiple cones, each cone generated by back-projecting the object's silhouette in one of multiple images.
- *Tomography reconstruction* that reconstructs an object as the integral of multiple projections. Each projection refers to cross-sectional information of the extent to which the intensity of an energy beam is reduced as it passes through the object and is obtained by illuminating the object from different directions.

Among these approaches to 3D reconstruction, SfM has the advantages of (1), multiple image redundancy that can be used to retrieve scene geometry more accurately and (2), the fact that it does not require calibrated cameras. It is therefore not surprising that it has found applications in a number of fields:

- The 3D reconstruction of an environment is crucial to the navigation of autonomous robots [3].
- In the film and gaming industry, motion capture is used extensively to reconstruct 3D human motion. This motion is then used to create computer generated characters with realistic movement. One example is the 2007 film Beowulf, loosely based on the old English poem, that is fully computer generated and was created with motion capture techniques [17].
- In archeology, SfM is used to preserve world cultural heritage. In the valley of Bamiyan, Afghanistan, approximately 1700 years ago, two large standing Buddha statues, one of the two being the tallest representation of standing Buddha in the world, were carved out of the sedimentary rock of the region. In March 2001 the Taleban government militia demolished the colossal statues. After the destruction a group from the Swiss Federal Institute of Technology in Zurich, completed the computer reconstruction of the Great Buddha from surviving photographic images, see [8]. This computer reconstruction can serve as a basis for a physical reconstruction.
- It can even be used in the treatment of cancer. The medical radiation treatment facility at iThemba Labs requires a precise and robust patient positioning system and a SfM system to verify correct positioning has been proposed [38].

Each SfM application has its own specific requirements and its own set of 3D reconstruction assumptions. Robot navigation, for example, typically takes place in a static environment with rigid structure, that differs from human motion capture where the motion of human facial features is complex and elastic. Therefore, to efficiently reconstruct a range of objects from various sources, two popular approaches to SfM were developed: the Kalman filter approach introduced by Azarbayejani and Pentland [1] and the matrix factorization approach introduced by Tomasi and Kanade [34]. The Kalman filter, developed by Rudolf Kalman [15], is a recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. The matrix factorization approach is (typically) a batch processing algorithm that factorizes a large matrix containing all of the features for all of the frames of a video sequence into two matrices, one representing 3D structure and the other 3D motion. Both these SfM approaches contain a feature tracker and a reconstruction process—the two essential components of a SfM system.

The first component, the feature tracker, as the name suggests, detects and tracks a number of 2D features from one frame to the next in a motion sequence. These features are what we ultimately represent in 3D. The features may be pixels, point-wise features defined by corners, distinct patches of textures, or parametric curves that form the outlines of objects or parts of objects. Pixel-based feature tracking methods that take all of the pixels within a frame into account are referred to as dense optical flow methods. Point-wise feature tracking, tracking a limited number of features instead of all of the pixels, is referred to as sparse optical flow methods.

The second component, the reconstruction process, recovers the 3D information of the features moving across the 2D images of a video sequence. From the small differences in the relative changes between the features in the video sequence, the 3D structure is derived. This explanation is of course an oversimplification of a difficult problem. If nothing is known of the objects that caused the observed 2D motion, or the cameras providing the images, the number of degrees of freedom of the problem is large, and the problem becomes hard, as will be explained in detail in subsequent chapters.

In order to make the problem tractable, some assumptions need to be made. These include the type of camera to be used, and the kind of motion being observed. One limiting assumption, for example, is the 3D to 2D projection of the camera that can be approximated using one of a number of camera models. Some of the most popular models in decreasing order of complexity are:

- The *perspective camera model* that provides a good approximation of a real camera and the human eye.
- The *paraperspective camera model* which is an approximation of the perspective camera model. It accounts for the scaling effect as an object moves toward or away from the camera as well as the different angles from which an object is viewed as it moves parallel to the camera's image plane.
- The *weak perspective camera model* that only accounts for the perspective scaling effect.
- The *orthographic camera model* that projects without accounting for either scaling or angle.

For the application presented in this thesis, the number of independently moving objects is limited. Some of the different types of objects one might want to consider include:

- A *rigid object,* which is the idealization of a solid body that only rotates and translates (rigid motion). It undergoes no deformation so that the distance between any two points on the object is constant.
- An *articulated object,* which is an object that consists of multiple rigid objects connected by joints.
- An *elastic object,* which is a solid body that deforms under stress. The distance between any two points on the object is not necessarily constant. This is a very difficult object to model, since the relative motion between projected features is not necessarily caused by the motion of the object.

In this thesis we only consider rigid objects.

As mentioned, each of the two SfM approaches has its limitations. The Kalman filter approach is limited in that it can only reconstruct a single rigid object and the motion between frames typically has to be small (this makes it ideal for real-time video). While the Kalman filter approach does not require a calibrated camera, the camera (or cameras) has to have a fixed calibration with no change in calibration parameters from frame to frame. The Kalman filter approach reconstructs up to a similarity. The first Kalman filter based approach used the extended Kalman filter (EKF) for reconstruction in the case of a perspective camera model and a single rigid object [1], and was improved by using the nonlinear unscented Kalman filter (UKF) by Venter [39].

The matrix factorization approach, on the other hand, is able to handle features from multiple rigid objects, large inter-frame motion and cameras with different calibration parameters. A straightforward matrix factorization, assuming a perspective camera, can do no better than a projective reconstruction which is very general, and not very useful since it allows severe deformation of an object. One has to resort to techniques from auto-calibration to upgrade the projective reconstruction to a similarity.

The original matrix factorization algorithm of Tomasi and Kanade [34] assumes an orthographic camera and a single rigid object. It was then extended to the weak perspective camera model [9], the paraperspective camera model [21] and finally to the perspective camera model, with the projective reconstruction limitation alluded to above, by Sturm and Triggs [31]. The matrix factorization algorithm was also applied to multiple rigid objects by Costaira and Kanade [4] assuming an orthographic camera model. The idea of auto-calibrating cameras to upgrade a projective reconstruction originated with Faugeras, Luong and Maybank [6]. First only cameras with fixed calibration were considered but later less restrictive constraints were investigated. Triggs [37] introduced the concept of the absolute dual quadric as a numerical device and investigated nonlinear solution methods, improving auto-calibration performance.

The objective of this thesis is to develop a full SfM implementation that reconstructs up to a similarity transformation, the structure and motion of multiple independently moving rigid bodies, assuming a perspective camera(s) which is not required to be calibrated. We note that the matrix factorization overcomes some of the limitations of the sequential Kalman filter approach; factorizing a few frames

- speeds up convergence
- allows for segmentation into different objects.

Matrix factorization results can also subsequently be used to initialise a Kalman filter algorithm.

Since the matrix factorization method is limited to a projective reconstruction for a perspective camera, we follow it with a metric rectification through auto-calibration techniques based on the absolute dual quadric. We further adopt the Lucas and Kanade (LK) [18] sparse optical flow feature tracker for our SfM system and focus on developing the 3D reconstruction component.

During the development of the SfM system an issue arose in the metric rectification process. The process consists of estimating the absolute dual quadric and is implemented as a constrained nonlinear optimization. This optimization fails to converge to the correct solution under certain conditions. Basically what happens is that the plane at infinity may intersect the object, i.e. the reconstructed object falls on both sides of the plane at infinity. This of course has to be corrected, and for that purpose the cheiral inequalities are used, as suggested by Hartley [11] [10]. We use the cheiral inequalities for two purposes: (1), to transform our initial projective reconstruction to a quasi-affine reconstruction that ensures the plane at infinity does not intercept the object and (2), as a constraint to the nonlinear optimization. Using the cheiral inequalities as a constraint to the nonlinear optimization in the calculation of the absolute dual quadric is novel, to the best of our knowledge. Hartley used the cheiral inequalities to construct a number of candidate pseudo-affine reconstructions to be refined to an affine reconstruction that was converted to their Euclidean reconstruction.

One other issue worth mentioning is adapting the motion segmentation algorithm of Costeira and Kanade [4], developed for the orthographic camera model, to a perspective camera model. It may come as a surprise, but as far as we know this has not been done. The modification is basically straightforward and only requires the calculation of a set of correctly scaled projection depths for all of the features. More detail is given in Chapter 4.

The SfM software implementation is developed in Python and thus cross-platform and comes with a graphical user interface (GUI) for user-friendly use.

It was developed in-house with contributions from open-source Python packages — e.g. OpenCV for Fundamental matrix calculations and feature tracking, CVXOPT for linear programming, SciPy for nonlinear optimization, and MayaVi for 3D plotting.

1.1 Overview of this work

The first two chapters give the theoretical background that is required to follow the arguments in the rest of the thesis. An understanding of projective geometry, its stratification, and knowledge of the perspective camera model that is used is crucial. An overview of projective geometry is given in Chapter 2, including a layout of the different strata and how to convert between them. The perspective camera model is explained in Chapter 3.

The next two chapters develop the 3D reconstruction algorithm for the SfM system. In Chapter 4 a matrix factorization algorithm is discussed that returns a projective reconstruction of image features. Next, in Chapter 5 an auto-calibration algorithm is presented, based on the absolute dual quadric to upgrade the projective factorization of Chapter 4 to the desired metric reconstruction. There is also a discussion of critical motion sequences, i.e. motion sequences that do not allow a complete solution. While the reconstruction algorithm discussed in Chapters 4 and 5 is limited to a single rigid object, Chapter 6 focuses on extending the algorithm to multiple, independently moving, rigid objects.

With the reconstruction algorithm fully developed, in Chapter 7 we examine the complication caused by the situation when the plane at infinity intersects the object. In Chapter 8 a variety of experiments are performed, ranging from the purely synthetic to real sequences illustrating the performance of the SfM system.

In the final chapter we review how we arrived at the proposed algorithms and the results obtained. This work is then concluded with a number of suggestions on future improvements.

Supplementary material is provided in the Appendix A where an overview is given of the resources provided on an accompanying CD. It contains, among other things, our SfM software and all the experimental sequences used in this work. Appendix A also provides a user guide to the SfM software, including descriptions of the most important modules and explaining how to use the GUI.

Chapter 2

Projective geometry

The 3D reconstructions studied in this work depend heavily on the camera model that is adopted, a perspective camera. The transformations involved in projecting 3D features onto the image plane are best described using projective geometry, with its underlying notion of *homogeneous coordinates*. In fact metric reconstructions from multiple views are impossible without a reasonably complete understanding of projective geometry. In this chapter the basic ideas that will be used in the rest of the work are given. Without it, it will be hard to follow the rest of the discussion. For a more thorough explanation the reader should consult [22], [13] and [6].

2.1 Projective spaces

A point in the projective n-space, $\mathbb{P}^n$, is represented by a $(n+1)$ homogeneous vector with coordinates, $\mathbf{x} = [x_1, \ldots, x_{n+1}]^T$, and at least one of its elements is not equal to zero. That means that all non-zero multiples of elements in $\mathbb{P}^n$ are identified, i.e. if $\mathbf{x}$, $\mathbf{y} \in \mathbb{P}^n$ then $\mathbf{x}$ and $\mathbf{y}$ are the same if, and only if, $\mathbf{x} = \lambda\mathbf{y}$ for some non-zero real number λ.

A collineation is a mapping between projective spaces that preserve collinearity, i.e. points connected by a straight line are mapped to points again connected by a straight line. A collineation from $\mathbb{P}^m$ to $\mathbb{P}^n$ is represented by a $(m+1) \times (n+1)$ matrix H. Points are transferred $\mathbf{x} \to \mathbf{x}' = H\mathbf{x}$. Observe that H and λH, with λ non-zero, represent the same collineation.

Note that there is a one-to-one correspondence between the Euclidean space

$\mathbb{R}^n$ and the projective space $\mathbb{P}^n$.

$$\text{For } \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n+1} \end{bmatrix} \in \mathbb{P}^n \Rightarrow \mathbf{x} = \begin{bmatrix} \frac{p_1}{p_{n+1}} \\ \frac{p_2}{p_{n+1}} \\ \vdots \\ \frac{p_n}{p_{n+1}} \end{bmatrix} \in \mathbb{R}^n.$$

Alternatively,

$$\text{for } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n \Rightarrow \mathbf{p} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \in \mathbb{P}^n.$$

Note that we have not said anything about p_{n+1} being non-zero. In fact $\mathbf{p} \in \mathbb{P}^n$ with $p_{n+1} = 0$ is perfectly well-defined. It is just the corresponding points in $\mathbb{R}^n$ that are problematic. This is part of the power of projective geometry, and it will prove useful to define the points $\mathbf{p} \in \mathbb{P}^n$ with $p_{n+1} = 0$ as the plane at infinity.

2.2 The projective plane

The projective plane is the projective space $\mathbb{P}^2$. A point in $\mathbb{P}^2$ is represented by a homogeneous 3-vector $\mathbf{x} = [x, y, w]^T$. Line $\boldsymbol{l}$ is also represented by a homogeneous 3-vector, i.e. $\boldsymbol{l} = [l_1, l_2, l_3]^T$. A point $\mathbf{x}$ is on a line $\boldsymbol{l}$ if, and only if,

$$\boldsymbol{l}^T \mathbf{x} = 0. \tag{2.1}$$

Alternatively, given a point $\mathbf{x}$, one can also state that a line $\boldsymbol{l}$ passes through $\mathbf{x}$ if, and only if, (2.1) holds. This interchangeability of lines and points is known as the principle of duality.

A line $\boldsymbol{l}$ passing through points $\mathbf{x}_1$ and $\mathbf{x}_2$ is given by the vector product $\boldsymbol{l} = \mathbf{x}_1 \times \mathbf{x_2}$. This can also be written as,

$$\boldsymbol{l} = [\mathbf{x}_1]_\times \mathbf{x}_2 \text{ with } [\mathbf{x}_1]_\times = \begin{bmatrix} 0 & w_1 & -y_1 \\ -w_1 & 0 & x_1 \\ y_1 & -x_1 & 0 \end{bmatrix}.$$

The dual formulation gives the intersection point of two lines, $\mathbf{x} = \boldsymbol{l}_1 \times \boldsymbol{l}_2$.

All the lines passing through a specific point form a pencil of lines. The two lines, $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ are distinct elements of the pencil. All the other lines can be obtained through $\boldsymbol{l} = \lambda_1 \boldsymbol{l}_1 + \lambda_2 \boldsymbol{l}_2$ for non-zero scalars λ_1 and λ_2 with only the ratio $\frac{\lambda_1}{\lambda_2}$ of importance.

2.3 Projective 3-space

Projective 3-space is the projective space of $\mathbb{P}^3$. A point in $\mathbb{P}^3$ is represented by a 4-vector $\mathbf{X} = [X, Y, Z, W]^T$. In $\mathbb{P}^3$ the dual entity of a point is a plane $\mathbf{\Pi}$, also represented by a 4-vector, i.e. $\mathbf{\Pi} = [\pi_1, \pi_2, \pi_3, \pi_4]^T$. A point $\mathbf{X}$ is located on a plane $\mathbf{\Pi}$ if, and only if,

$$\mathbf{\Pi}^T \mathbf{X} = 0.$$

A line is given by the combination of two points, $\lambda_1 \mathbf{X}_1 + \lambda_2 \mathbf{X}_2$, or the intersection of two planes, $\mathbf{\Pi}_1 \cap \mathbf{\Pi_2}$. Note that two parallel planes intersect in a well-defined line, a line on the plane at infinity, in $\mathbb{P}^3$. The plane at infinity is discussed in detail later in this chapter.

2.4 Projective transformations

A 2D homography maps lines to lines ($\mathbb{P}^2 \rightarrow \mathbb{P}^2$) and is given by a non-singular 3×3 homogeneous matrix H. A point $\mathbf{x}$ transforms under a homography with

$$\mathbf{x}' = H\mathbf{x}. \tag{2.2}$$

Given a point $\mathbf{x}'$ on line $\boldsymbol{l}'$ where $\mathbf{x}'$ is transformed by 2.1, we find that

$$\begin{aligned} 0 &= \boldsymbol{l}'^T \mathbf{x}' \\ &= \boldsymbol{l}'^T H\mathbf{x} \\ &= (H^T \boldsymbol{l}')^T \mathbf{x}, \end{aligned}$$

implying $\mathbf{x}$ is on line $\boldsymbol{l} = H^T \boldsymbol{l}'$. We therefore conclude that lines transform under a homography as

$$\boldsymbol{l}' = H^{-T} \boldsymbol{l}. \tag{2.3}$$

Reasoning similarly, the transformation of 3D points and planes in $\mathbb{P}^3$ is given by:

$$\begin{aligned} \mathbf{X}' &= H\mathbf{X} \\ \mathbf{\Pi}' &= H^{-T} \mathbf{\Pi}, \end{aligned} \tag{2.4}$$

where 3D homography H is a non-singular 4×4 homogeneous matrix.

2.5 Conics and quadrics

2.5.1 Conic

A conic is a curve in $\mathbb{P}^2$ described by the locus (collection of points which share a property) of all points $\mathbf{x}$ satisfying the homogeneous quadratic equation

$$S(\mathbf{x}) = \mathbf{x}^T \omega \mathbf{x} = 0, \tag{2.5}$$

where ω is a 3×3 symmetric matrix defined up to scale. Therefore, a conic has 5 degrees of freedom. We do not distinguish between the matrix ω and the curve it defines, and simply refer to the conic ω. In Euclidean geometry, proper conics (conics that have full rank) are one of three types: hyperbola, ellipse or parabola.

2.5.2 Line-conic intersection

Let $\mathbf{x}$ and $\mathbf{x}'$ be two points defining a line. Any point on this line can be represented by $\mathbf{x} + \lambda\mathbf{x}'$. This point lies on a conic if, and only if,

$$\begin{aligned} S(\mathbf{x} + \lambda\mathbf{x}') &= (\mathbf{x} + \lambda\mathbf{x}')\,\omega(\mathbf{x} + \lambda\mathbf{x}') = 0 \\ &= \mathbf{x}^T\omega\mathbf{x} + \lambda\mathbf{x}^T\omega\mathbf{x}' + \lambda\mathbf{x}'^T\omega\mathbf{x} + \lambda^2\mathbf{x}'^T\omega\mathbf{x}' = 0. \end{aligned}$$

This can be written as

$$S(\mathbf{x}) + 2\lambda S(\mathbf{x}, \mathbf{x}') + \lambda^2 S(\mathbf{x}') = 0 \tag{2.6}$$

where

$$S(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\omega\mathbf{x}' = S(\mathbf{x}', \mathbf{x}).$$

This means, in general, there are two intersection points between a conic and a line. These intersections can be real or complex and can be obtained by solving (2.6) for λ.

2.5.3 Tangent to a conic

The two intersecting points of a conic and a line coincide if the discriminant of (2.6) is zero. Since it is an quadratic equation in λ, the discriminant can be written as

$$4S(\mathbf{x}, \mathbf{x}')\,S(\mathbf{x}, \mathbf{x}') - 4S(\mathbf{x})\,S(\mathbf{x}') = 0.$$

If $\mathbf{x}$ is considered fixed and to belong to the conic $S(\mathbf{x}) = 0$, the equation becomes,

$$S(\mathbf{x}, \mathbf{x}')\,S(\mathbf{x}, \mathbf{x}') = 0.$$

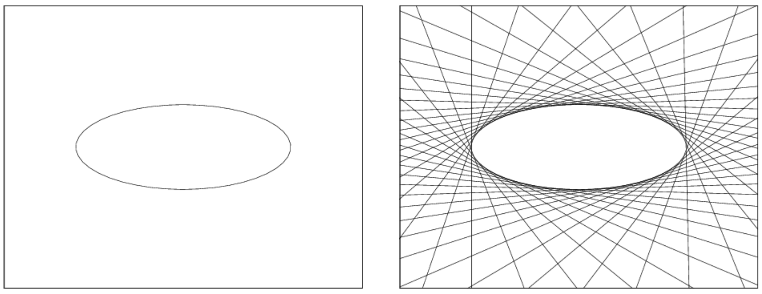

(i) conic ω (ii) dual conic ω^*

Figure 2.1: A conic ω and its dual ω^*.

This implies

$$S(\mathbf{x}, \mathbf{x}') = 0 = \mathbf{x}^T \omega \mathbf{x}' = \left(\omega^T \mathbf{x}\right)^T \mathbf{x}' = \left(\omega^T \mathbf{x}'\right)^T \mathbf{x},$$

which is linear in the coefficients of $\mathbf{x}'$. This, according to (2.1), means there is only one tangent for $\mathbf{x}$ and $\mathbf{x}'$, therefore the same point on the conic. Then the tangent $\boldsymbol{l}$ is represented by

$$\boldsymbol{l} = \omega^T \mathbf{x} = \omega \mathbf{x}, \tag{2.7}$$

since ω is symmetric.

2.5.4 Dual conic

According to the duality principle one can also define a dual conic ω^* as the lines tangent to a conic. It is represented by the adjoint matrix ω^* of the conic matrix ω, and is given by,

$$S\left(\boldsymbol{l}^T\right) = \boldsymbol{l}^T \omega^* \boldsymbol{l} = 0,$$

where ω^* is again a symmetric, homogeneous matrix, also with 5 degrees of freedom.

When $\mathbf{x}$ belongs to a conic, i.e. $\mathbf{x}^T \omega \mathbf{x} = 0$, the tangent, $\boldsymbol{l} = \omega \mathbf{x}$, satisfies

$$\boldsymbol{l}^T \omega^{-1} \boldsymbol{l} = 0,$$

in the case when ω is non-singular. Thus the dual conic is represented by the adjoint matrix $\omega^* = \omega^{-1}$ (assuming ω is full rank).

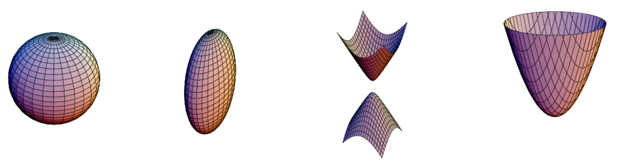

(i) sphere (ii) ellipsoid (iii) hyperboloid (iv) paraboloid

Figure 2.2: Some quadric examples.

2.5.5 Transformation of conic and dual conic

We consider that transformation of a conic under a 2D homography, $\mathbf{x}' = H\mathbf{x}$ and $\boldsymbol{l}' = H^{-T}\boldsymbol{l}$. For $\mathbf{x}$ on the conic (2.5) it follows that

$$\begin{array}{ccccc} 0 & = & \mathbf{x}^T\omega\mathbf{x} & = & \mathbf{x}'^T H^{-T}\omega H^{-1}\mathbf{x}' \\ 0 & = & \boldsymbol{l}^T\omega^*\boldsymbol{l} & = & \boldsymbol{l}'^T H\omega^* H^T\boldsymbol{l}' \end{array},$$

and thus

$$\begin{array}{c} \omega' = H^{-T}\omega H^{-1} \\ \omega^{*\prime} = H\omega^* H^T \end{array}. \tag{2.8}$$

This also implies that $(\omega')^* = (\omega^*)'$.

2.5.6 Quadric

In projective $\mathbb{P}^3$ space a similar concept to the conic exists, the quadric Q. A quadric is a surface defined by the locus of all $3D$ points $\mathbf{X}$ that satisfy

$$\mathbf{X}^T Q\mathbf{X} = 0,$$

where Q is a 4×4 symmetric matrix defined up to scale. A quadric therefore has 9 degrees of freedom. Note that the intersection of a plane with a quadric is a conic.

2.5.7 Dual quadric

Similar to the dual conic, the dual quadric Q^* is defined as the planes tangent to the quadric, and is written as

$$\mathbf{\Pi}^T Q^*\mathbf{\Pi} = 0,$$

where Q^* is a 4×4 symmetric matrix. The dual quadric also has 9 degrees of freedom.

2.5.8 Tangent to a quadric

Similar to (2.7), the tangent plane to a conic for point $\mathbf{X}$ on the conic is

$$\mathbf{\Pi} = Q\mathbf{X}.$$

2.5.9 Relation between quadric and dual quadric

When $\mathbf{X}$ varies along the quadric it satisfies $\mathbf{X}^T Q\mathbf{X} = 0$ and therefore the tangent plane $\mathbf{\Pi}$ to the quadric at $\mathbf{X}$ satisfies $\mathbf{\Pi}^T Q^* \mathbf{\Pi} = 0$. This again shows, that the dual quadric is given by $Q^* = Q^{-1}$ (again provided Q is full rank).

2.5.10 Transformation of quadric and dual quadric

The transformations for quadrics and dual quadrics follow as in the case of the conic and dual conic, and is therefore given by

$$\begin{aligned} Q' &= H^{-T} Q H^{-1} \\ Q^{*\prime} &= H Q^* H^T, \end{aligned} \tag{2.9}$$

where H is now a 3D homography. Again it is clear, $(Q^*)' = (Q')^*$.

2.6 The stratification of 3D geometry

We usually perceive the world as embedded in a $3D$ Euclidean space, a place where measurements of distances and angles are well-defined. This viewpoint is not mandatory, especially for artificial systems like robots. It is sometimes desirable, or only possible, to deal with a more general, and therefore simpler, geometric structure. These geometric structures can be thought of as different geometric strata. The simplest being projective, then affine, next metric and finally Euclidean structure. These strata can be considered as subgroups of each other, e.g. the metric group is a subgroup of the affine group and both are subgroups of the projective group.

Acting on the elements of these strata, some properties of the strata are left invariant by these transformation groups. More precisely, an invariant is a property of a geometric configuration that is not altered by a transformation under a specific group. Uncovering these entities allow us to upgrade the structure of the geometry to a higher strata

In the following paragraphs the different strata are shortly discussed along with their associated groups of transformations, and their invariants.

2.6.1 Projective stratum

This is the least structured, or most general stratum and therefore the one with the least number of invariants. A projective transformation of $\mathbb{P}^3$ is represented by the 4×4 non-singular homogeneous matrix

$$H_p = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} = \begin{bmatrix} A & \mathbf{b} \\ \mathbf{v}^T & k \end{bmatrix}, \tag{2.10}$$

with A a non-singular 3×3 matrix and $\mathbf{b}$ and $\mathbf{v}$ both 3-vectors. Since H_p is homogeneous, only the ratios between its 16 elements are significant. Therefore it has 15 degrees of freedom.

For projective transformations, the most fundamental invariant is the cross-ratio. Let $\mathbf{X}_i = \mathbf{X} + \lambda_i \mathbf{X}'$, for $i = 1, \ldots, 4$, be four different collinear points. The cross-ratio is then defined as

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

The cross-ratio does not dependent on the choice of reference points $\mathbf{X}$ and $\mathbf{X}'$ and, as stated, is invariant under the group of projective transformations of $\mathbb{P}^3$. A similar cross-ratio can be defined for four lines intersecting in a point or four planes intersecting in a common line.

A number of further invariant properties can be be expressed in terms cross-ratio constructions, e.g. concurrency, lines intersecting in a single point, collinearity, points on a straight line and intersection of geometric objects.

2.6.2 Affine strata

The affine group is located in between the projective and metric group with more structure than the first and less than the latter. The affine stratum differs from the projective by identifying a special plane, the plane at infinity $\mathbf{\Pi}_\infty$. The plane at infinity is defined as all the 3D homogeneous points $\mathbf{X} = [X,\, Y,\, Z,\, W]^T$ with $W = 0$ and thus $\mathbf{\Pi}_\infty$ has the canonical form $\mathbf{e}_4 = [0,\, 0,\, 0,\, 1]^T$.

An affine transformation is given by the 4×4 non-singular homogeneous matrix

$$H_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & \mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.11}$$

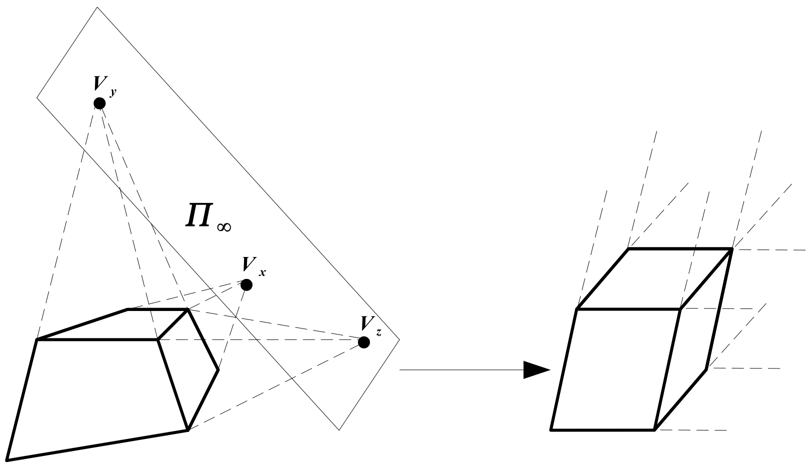


Figure 2.3: Projective (left) and affine (right) structures which are equivalent to a cube under their respective ambiguities. The vanishing points obtained from lines which are parallel in the affine stratum constrain the position of the plane at infinity in the projective representation.

with 3×3 matrix A non-singular. It should be clear that affine transformations of $\mathbb{P}^3$ has 12 degrees of freedom.

The plane at infinity is an invariant under affine transformations and is the fundamental invariant of the affine stratum, $\mathbf{\Pi}_\infty = H_A^{-T}\mathbf{\Pi}_\infty$ or $H_A^T\mathbf{\Pi}_\infty = \mathbf{\Pi}_\infty$. Note that the positions of points at infinity can change but these points stay on the plane at infinity. This has a useful interpretation: Since parallel planes intersect on the plane at infinity, after an affine transformation they still intersect on the plane at infinity $\mathbf{\Pi}_\infty$. Consequently, affine transformations map parallel planes to parallel planes.

From projective to affine

Starting with the plane at infinity $\mathbf{\Pi}_\infty$ in its canonical position given by $\mathbf{e}_4 = [0,\ 0,\ 0,\ 1]^T$, under a general projective transformation it is mapped to a finite position, i.e. $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T,\ 1\right]^T$ when the last element is scaled to one. This is, for example, the reason why parallel lines intersect at a finite point under the projective transformation induced by our cameras. The problem is to find a transformation that maps the projective stratum onto an affine stratum. It should be clear that this can only be done up to an arbitrary affine transformation.

The idea is to find the projective representation of the plane at infinity in the scene, as illustrated in Figure 2.3. Since we know that we are looking at a cube under a projective transformation in the figure, one can identify three vanishing points, i.e. the three points on the plane at infinity, by locating the intersection

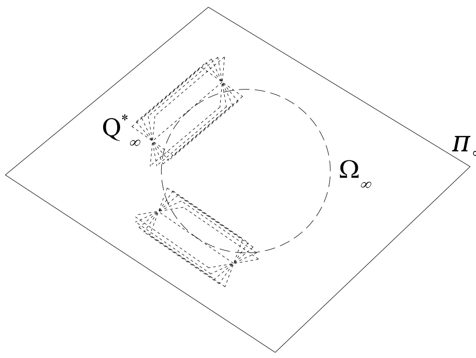


Figure 2.4: The absolute conic Ω_∞ and the absolute dual quadric Q^*_∞ in 3D space.

of what used to be parallel lines. These three points define the projective representation of the plane at infinity. With the projective representation of the plane at infinity $\mathbf{\Pi}_\infty$ and its canonical form $\mathbf{e}_4$ known, we now have to find the transformation that maps the one to the other.

Since

$$\mathbf{e}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = H^{-T}\mathbf{\Pi}_\infty \text{ or } H^T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \mathbf{\Pi}_\infty,$$

the fourth row of H is determined. Thus a natural choice for the projective to affine transformation, H_{PA} is

$$H_{PA} = \begin{bmatrix} I & \mathbf{0} \\ \boldsymbol{\pi}^T & 1 \end{bmatrix}$$

where π is the first three rows of $\mathbf{\Pi}_\infty$ when the last component is scaled to 1. However any transformation of the form

$$H_{PA} = \begin{bmatrix} A & \mathbf{0} \\ \boldsymbol{\pi}^T & 1 \end{bmatrix} \tag{2.12}$$

with A non-singular also maps $\mathbf{\Pi}_\infty$ to $\mathbf{e}_4$.

2.6.3 Metric stratum

This stratum corresponds to the set of similarity transformations. These transformations consist of a scaling, an orthonormal rotation and a translation. It corresponds to the Euclidean transformations but complements that with a scaling. When no absolute yardstick is available, this is the highest level of geometric structure that can be attained.

The transformation is represented in Euclidean coordinates by

$$\begin{aligned}\widehat{\mathbf{X}}' &= \sigma \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \widehat{\mathbf{X}} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \\ &= \sigma R \widehat{\mathbf{X}} + \mathbf{t} \end{aligned} \tag{2.13}$$

with r_{ij} the elements of the orthonormal matrix R. The coefficients r_{ij} are related by six independent constraints that can be written as,
$R^T R = RR^T = I$. This leaves three degrees of freedom that define the axis– and angle of rotation. Rewriting (2.13) in homogeneous coordinates as
$\mathbf{X}' = H_M \mathbf{X}$, the transformation matrix has the form

$$H_M = \begin{bmatrix} \sigma r_{11} & \sigma r_{12} & \sigma r_{13} & t_1 \\ \sigma r_{21} & \sigma r_{22} & \sigma r_{23} & t_2 \\ \sigma r_{31} & \sigma r_{32} & \sigma r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sigma R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

A similarity transformation has 7 degrees of freedom, 3 for rotation, 3 for translation and 1 for scale.

In the metric case the fundamental invariants are *relative lengths* and *relative angles*. Similar to the affine case, they are related by an invariant geometric entity. Besides leaving the plane at infinity unchanged, similarity transformations leave the *absolute conic*, $\boldsymbol{\Omega}_\infty$, invariant.

The absolute conic is a specific conic located on the plane at infinity that is a virtual (conic of imaginary points), circle conic. The absolute conic is described, in the metric stratum, by all 3D points $\mathbf{X} = [X,\, Y\, Z,\, W]^T$ that satisfy

$$X^2 + Y^2 + Z^2 = 0 \text{ and } W = 0. \tag{2.14}$$

Therefore, for Ω_∞ in the metric stratum on Π_∞, Ω_∞ has the canonical form

$$\omega = I.$$

This means that for points with $W = 0$ on $\boldsymbol{\Pi}_\infty$, the defining equation is given by

$$[X,\, Y,\, Z]\, I\, [X,\, Y,\, Z]^T = \mathbf{0}\,.$$

It is often more practical to represent the absolute conic with its dual entity, Q^*_∞, the absolute dual quadric. Figure 2.4 illustrates the geometric concept of the absolute dual quadric, i.e. the planes tangent to Ω_∞ so that Ω_∞ is the "rim" of Q^*_∞. The absolute dual quadric is a degenerate dual quadric, i.e. the 4×4

homogeneous matrix representing the dual quadric does not have full rank (a rank of 3 instead of 4). Thus the absolute dual quadric has 8 degrees of freedom, one less than full rank quadrics because of its zero determinant. The canonical form of Q^*_∞ in the metric stratum is

$$Q^*_\infty = \widetilde{I} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}.$$

One way to show that Q^*_∞ is a dual of Ω_∞ is to consider the absolute conic as the limit of a series of squashed ellipsoids, namely quadrics represented by the matrix

$$Q = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & k \end{bmatrix}.$$

As $k \to \infty$, the quadrics become increasingly close to the plane at infinity, and the only points the limit contain are $[X,\, Y,\, Z,\, 0]^T$ with $X^2+Y^2+Z^2 = 0$. These points are points on the absolute conic. However the dual of Q is the quadric

$$Q^* = Q^{-1} = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & k^{-1} \end{bmatrix},$$

which in the limit become the absolute dual quadric $Q^*_\infty = \widetilde{I}$.

Note that $\mathbf{\Pi}_\infty$ is the null-space of Q^*_∞,

$$Q^*_\infty \mathbf{\Pi}_\infty = \mathbf{0}. \tag{2.15}$$

It can easily be shown for the canonical case,

$$\widetilde{I}\mathbf{e}_4 = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = \mathbf{0}.$$

It can now be verified that similarity transformations leave the absolute conic and its associated entities unchanged, using (2.8) and (2.9):

$$\begin{aligned} H_M \tilde{I} H_M^T &= \begin{bmatrix} \sigma R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \sigma R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \\ &= \widetilde{I} \end{aligned}$$

and

$$\begin{aligned} \omega &= I = \sigma^{-1} R^{-T} I R^{-1} \sigma^{-1} \\ \omega^* &= I = \sigma R I R^T \sigma, \end{aligned}$$

taking into account that we are dealing with homogeneous quantities. Con-

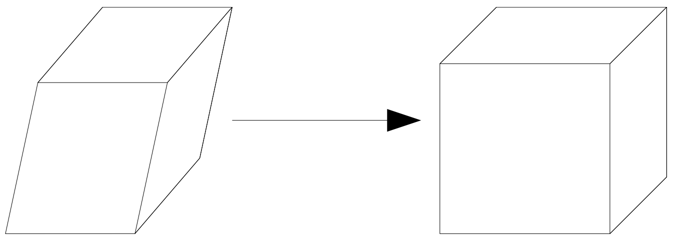


Figure 2.5: Affine (left) and metric (right) representation of a cube. The right angles and the identical lengths in the different directions of a cube give enough information to upgrade the structure from affine to metric.

versely it is easy to show that the projective transformations that leave the absolute dual quadric unchanged, form the group of similarity transformations,

$$\begin{aligned}\begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} &= H_P \widetilde{I} H_P^T \\ &= \begin{bmatrix} A & \mathbf{b} \\ \mathbf{v}^T & k \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} A^T & \mathbf{v} \\ \mathbf{b}^T & k \end{bmatrix} \\ &= \begin{bmatrix} AA^T & A\mathbf{v} \\ \mathbf{v}^T A^T & \mathbf{v}^T\mathbf{v} \end{bmatrix}.\end{aligned}$$

Therefore $AA^T = I$ and $\mathbf{v} = \mathbf{0}$ which exactly corresponds to the constraints for a similarity transformation. The same could be done for the absolute conic and the plane at infinity.

From projective or affine to metric

It is often required to upgrade a projective or affine representation to metric. This is done by retrieving the absolute conic or one of its associated entities. Since the absolute conic is located at the plane of infinity it is easier to retrieve it once the plane is identified, i.e. the affine structure has been recovered. It is however possible to retrieve both entities at the same time. The absolute dual quadric, Q^*_∞, is especially suited for this purpose since it encodes both entities at once.

Every known angle or ratio imposes a constraint on the absolute conic. Knowing enough constraints allows for the conic to be uniquely determined, e.g. In Figure 2.5 a cube is transformed so that its sides are orthogonal and all have the same length. Once the absolute conic is identified the geometry can be upgraded from projective or affine to metric by bringing it to its canonical (metric) form.

The procedure to go from projective to affine has already been explained in the affine stratum section (1.2.2) so the discussion is now limited to the affine to metric case. In this case there must be an affine transformation that brings the absolute conic to its canonical position; or, conversely, from its canonical position to its position under the affine transformation. Using (2.9) and (2.11) it follows that the absolute dual quadric under the affine transformation is given by:

$$Q_\infty^* = \begin{bmatrix} A & \mathbf{a} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} A^T & \mathbf{0} \\ \mathbf{a}^T & 1 \end{bmatrix} = \begin{bmatrix} AA^T & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}.$$

Then the affine conic representation of the absolute conic and its dual have the form

$$\omega^* = AA^T \text{ and } \omega = A^{-T}A^{-1}.$$

One possible choice for the transformation from affine to metric, H_{AM} is

$$H_{AM} = \begin{bmatrix} A^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.16}$$

so that

$$\begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} A^{-1}AA^TA^{-T} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} A^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} AA^T & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} A^{-T} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

where a valid A can be obtained from Q_∞^* through Cholesky factorization.

Combining (2.12) and (2.16),

$$H_{PM} = H_{PA}H_{AM} = \begin{bmatrix} I & \mathbf{0} \\ \pi^T & 1 \end{bmatrix} \begin{bmatrix} A^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} A^{-1} & \mathbf{0} \\ \pi^T & 1 \end{bmatrix}$$

2.6.4 Euclidean stratum

Euclidean stratum does not differ much from the metric stratum. The difference is that scales are fixed and therefore not only can relative lengths be measured, but also absolute lengths. The Euclidean transformation is represented in Euclidean coordinates by

$$\begin{aligned} \widetilde{\mathbf{X}}' &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \widetilde{\mathbf{X}} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \\ &= R\widetilde{\mathbf{X}} + \mathbf{t} \end{aligned} \tag{2.17}$$

Stratum	dof	Transformation	Invariants
projective	15	$H_P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$	cross-ratio
affine	12	$H_A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$	relative distances, along direction parallelism, *plane at infinity*
metric	7	$H_M = \begin{bmatrix} \sigma r_{11} & \sigma r_{12} & \sigma r_{13} & t_1 \\ \sigma r_{21} & \sigma r_{22} & \sigma r_{23} & t_2 \\ \sigma r_{31} & \sigma r_{32} & \sigma r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	relative distances, angles, *absolute conic*
Euclidean	6	$H_E = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	absolute distances

Table 2.1: The number of degrees of freedom, transformations and invariants corresponding to the different geometric strata (the coefficients r_{ij} form orthonormal matrices).

with R an orthonormal matrix. The Euclidean transformation then has the homogeneous form of $\mathbf{X}' = H_E\mathbf{X}$ with

$$H_E = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

This transformation has 6 degrees of freedom, 3 for rotation and 3 for translation. If R is a rotation matrix, $\det R = 1$, then the transformation represents a rigid motion in space.

2.6.5 Overview of the different strata

The various strata are summarized in Table 2.1. Equivalent shapes for a cube over the different geometric strata are displayed in Figure 2.6.

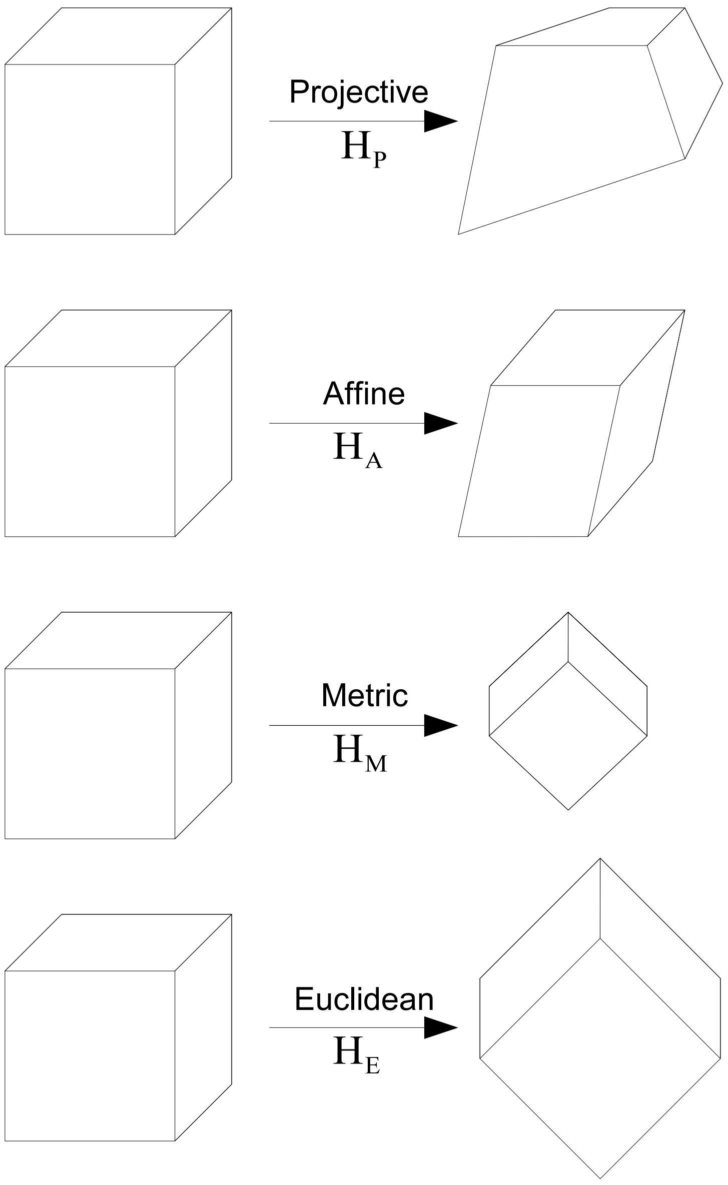


Figure 2.6: Shapes which are equivalent to a cube for the different geometric ambiguities.

Chapter 3

Perspective camera model

This study is based on the perspective camera model, also known as the pin-hole camera, which is an idealized mathematical model of real cameras. It is used since it models most cameras, including the human eye, relatively well except for some factors e.g. geometric distortions of the types that are caused by lenses.

The perspective camera is a central projection camera and is illustrated in Figure 3.1. This means the projection process is completely determined by choosing a finite perspective projection center **C** and an image plane. The projection of a scene point **X** is then obtained as the intersection of the line passing through both **X** and **C**, with the image plane. All cameras modeling

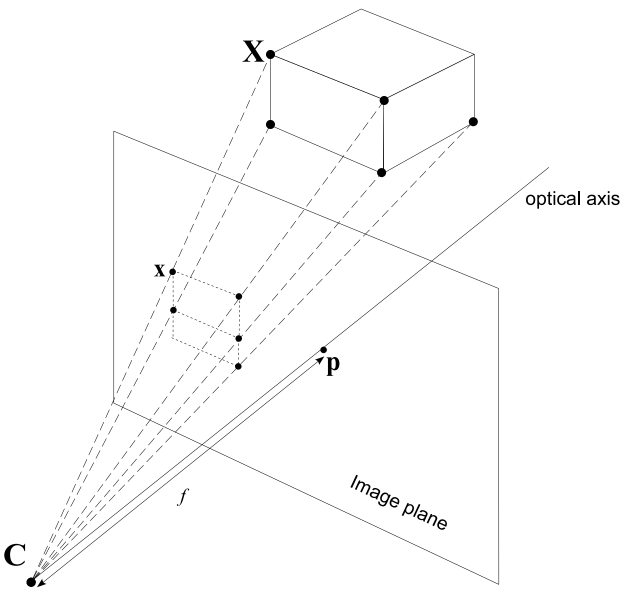


Figure 3.1: Perspective projection with a camera center **C**, focal length f and principal point **p** in the image plane.

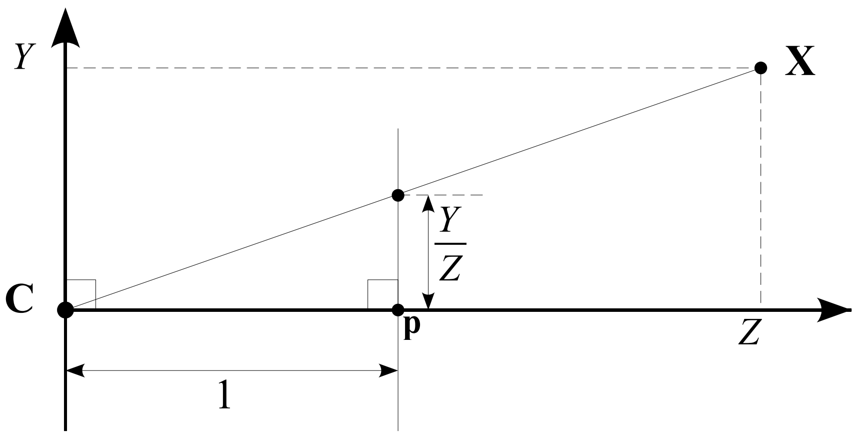


Figure 3.2: Perspective camera projection from a view parallel to the camera X-axis. $\mathbf{C}$ is the camera center and $\mathbf{p}$ the principal point. The camera center is placed at the coordinate origin and $Z = 1$ is the image plane.

central projection are specializations of the general projective camera model.

Using homogeneous coordinates we denote $\mathbf{X} = [X, Y, Z, W]^T$ as a 3D point and $\mathbf{x} = [x, y, w]^T$ as its 2D projection. The general projective camera is represented by the mapping

$$\mathbf{x} = P\mathbf{X}, \tag{3.1}$$

where the projection matrix P is an arbitrary 3×4 homogeneous matrix of rank 3. The rank 3 requirement arises, because any lesser rank maps onto a line or point and not the required image plane. As a consequence of homogeniality, the general projective camera has 11 degrees of freedom.

3.1 Perspective camera properties

Consider a 3D point $\mathbf{X}$ in the world coordinate system (WCS) and $\mathbf{x}$ its 2D projection in pixel coordinates under the mapping $\mathbf{x} = P\mathbf{X}$ with P the camera projection matrix and $\mathbf{C}$ the camera coordinate system (CCS) camera center. In the special case of Figure 3.2 with the camera center at the origin and the image plane at $Z = 1$, it is clear from similar triangles that $\mathbf{X}$ centrally projects to $\mathbf{x} = [X/Z, Y/Z, 1]^T$. According to [13] the perspective camera matrix can be decomposed, in metric space, as

$$P = KR\left[\begin{array}{c|c} I & -\widetilde{\mathbf{C}} \end{array}\right] \tag{3.2}$$

with $\widetilde{\mathbf{C}}$ the non-homogeneous coordinates of the camera center in the WCS, R the 3×3 rotation matrix between the CCS and the WCS and K the 3×3

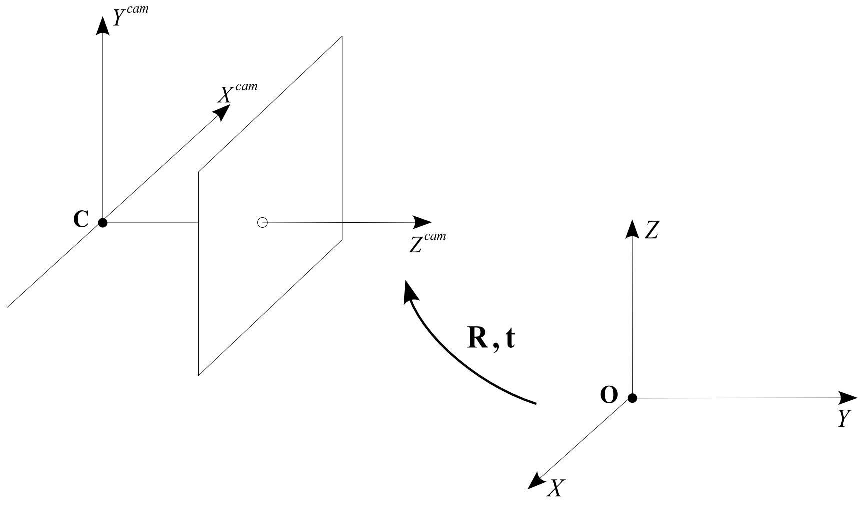


Figure 3.3: The Euclidean transformation between the world and camera coordinate system.

upper-triangular camera calibration matrix.

The decomposition in (3.2) can be rewritten as

$$KR\left[\begin{array}{c|c} I & -\widetilde{\mathbf{C}} \end{array}\right] = K\left[\begin{array}{c|c} R & -R\widetilde{\mathbf{C}} \end{array}\right] = K\left[\begin{array}{c|c} R & \mathbf{t} \end{array}\right]$$

where 3-vector $\mathbf{t}$ is the origin of the world coordinate system in terms of the camera coordinate system, seen in Figure 3.3. The motion of the world coordinate system relative to the camera is fully expressed with rotation matrix R and translation vector $\mathbf{t}$ so that we may write, in Euclidean coordinates,

$$\widetilde{\mathbf{X}}^{cam} = R\widetilde{\mathbf{X}} + \mathbf{t} = R\left(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}}\right)$$

expressing the position of 3D WCS point $\widetilde{\mathbf{X}}$ in the camera coordinate system.

The pixel coordinates of the image and $\mathbf{x}$ are not necessarily in Euclidean coordinates, with equal scales in both axes' directions. A typical example is a CCD camera with non-square pixels. The camera calibration matrix K is a 2D affine transformation matrix that contain the parameters that relate the CCS to the image coordinate system. These parameters are intrinsic to a camera and therefore called its internal parameters. Calibration matrix K has the general form

$$K = \begin{bmatrix} \alpha & s & x_0 \\ & r\alpha & y_0 \\ & & 1 \end{bmatrix}. \tag{3.3}$$

The calibration parameters are:

- $\alpha = fm_x$, representing the focal length f (distance between the center of projection, COP, and the image plane) in terms of the pixel dimension in the x-direction, m_x.
- $r = \frac{m_y}{m_x}$, the aspect ratio between m_x and m_y, the number of pixels per unit distance in image coordinates in the x and y direction.
- x_0, y_0, with $[x_0,\, y_0,\, 1]^T$ the principal point of the camera. The principal point is the camera center or COP in terms of the image coordinates.
- s, the skew parameter.

For a camera with fixed optics these parameters are identical for all images. For cameras with zoom and focusing capabilities the focal length and possibly the principal point can vary between images.

The perspective camera is defined as a special case of the general projection camera for which the left hand 3×3 submatrix of the camera matrix P is non-singular. Assigning B to the left 3×3 submatrix of P, we can rewrite (3.2) as

$$P = \left[\begin{array}{c|c} B & \mathbf{p}_4 \end{array}\right] \tag{3.4}$$

where $\mathbf{p}_4$ is the 4th column of P and

$$B \left[\begin{array}{c|c} I & B^{-1}\mathbf{p}_4 \end{array}\right] = B \left[\begin{array}{c|c} I & -\widetilde{\mathbf{C}} \end{array}\right].$$

It is clear from (3.2) and (3.4) that one can factorize $B = KR$ using RQ decomposition (QR decomposition with some algebraic manipulation), where K is the upper triangular camera calibration matrix of (3.3) and R the orthogonal rotation matrix. We now show that the non-singularity of B ensures that all perspective camera's have a finite center represented by homogeneous 4-vector $\mathbf{C}$.

First it has to be shown that camera center $\mathbf{C}$ is the one-dimensional right null-space of the camera matrix P. Consider a line in 3-space containing a point $\mathbf{C}$, any homogeneous 4-vector for which $P\mathbf{C} = \mathbf{0}$, and any other point $\mathbf{A}$. Points on this line are presented by the join $\mathbf{X}(\lambda) = \lambda\mathbf{A} + (1-\lambda)\mathbf{C}$. Under the mapping $\mathbf{x} = P\mathbf{X}$, points on this line are projected to

$$\mathbf{x} = P\mathbf{X}(\lambda) = \lambda P\mathbf{A} + (1-\lambda) P\mathbf{C} = \lambda P\mathbf{A}$$

since $P\mathbf{C} = \mathbf{0}$. This means all points on the line are mapped to the same point $P\mathbf{A}$, therefore the line must be a ray through the camera center.

It follows that $\mathbf{C}$ is the homogeneous representation of the camera center and the one-dimensional right null-space for P, since for all choices of $\mathbf{A}$ the line $\mathbf{X}(\lambda)$ is a ray through the camera center. We can now see from (3.4) and $P\mathbf{C} = \mathbf{0}$ that $\mathbf{C}$ has the form

$$\mathbf{C} = \begin{bmatrix} -B^{-1}\mathbf{p}_4 \\ 1 \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{C}} \\ 1 \end{bmatrix}.$$

This clearly shows that the non-singularity restriction ensures that all perspective camera's have finite centers.

3.2 Actions of the perspective camera

An overview is given of the action of the perspective camera on points, lines, planes, conics and quadrics.

3.2.1 On points

forward projection

It has already been shown that points $\mathbf{X}$ in 3-space are projected with P to image points $\mathbf{x} = P\mathbf{X}$.

back projection

Since all points on the ray through $\widetilde{\mathbf{C}}$ and $\mathbf{x}$ project onto $\mathbf{x}$, it is impossible to recover $\mathbf{X}$ given $\widetilde{\mathbf{C}}$ and $\mathbf{x}$. But we can at least recover the ray.

We know two points on this ray. The first is the camera center $\mathbf{C} = \left[\widetilde{\mathbf{C}}^T, 1\right]^T$. When writing P as $P = [B \,|\, \mathbf{p}_4]$, the camera center $\widetilde{\mathbf{C}}$ is given by $\widetilde{\mathbf{C}} = -B^{-1}\mathbf{p}_4$. The second known point is the point where the back projected ray intersects the plane at infinity, $\mathbf{X}_\infty = \left[\left(B^{-1}\mathbf{x}\right)^T, 0\right]^T$. Writing the line as the join of these two points on the ray,

$$\mathbf{X}(\mu) = \begin{bmatrix} -B^{-1}\mathbf{p}_4 \\ 1 \end{bmatrix} + \mu \begin{bmatrix} B^{-1}\mathbf{x} \\ 0 \end{bmatrix} = \begin{bmatrix} B^{-1}(\mu\mathbf{x} - \mathbf{p}_4) \\ 1 \end{bmatrix}.$$

3.2.2 On lines

forward projection

The camera center $\mathbf{C}$ and a 3-space line, not through $\mathbf{C}$, define a plane. Its image is the intersection between this plane and the image plane. It is well known that two planes intersect in a line, therefore, a 3-space line projects to a line in the image.

This is easily shown algebraically by denoting $\mathbf{A}$, $\mathbf{B}$ points in 3-space and $\mathbf{a}$, $\mathbf{b}$ their images under P. Then the points $\mathbf{X}(\mu) = \mathbf{A} + \mu\mathbf{B}$ on the line, which is the join of $\mathbf{A}$ and $\mathbf{B}$ in 3-space, project to the points

$$\mathbf{x}(\mu) = P(\mathbf{A} + \mu\mathbf{B}) = P\mathbf{A} + \mu P\mathbf{B} = \mathbf{a} + \mu\mathbf{b}$$

which is the joining of $\mathbf{a}$ and $\mathbf{b}$ in the image.

back projection

The set of points that map to a line in the image is a plane in space defined by the camera center and the image line, as was shown with the forward projection. We will now show that the set of points mapped under P to a line $\boldsymbol{l}$ in the image, is the plane $P^T\boldsymbol{l}$.

A point lies on $\boldsymbol{l}$ if, and only if, $\mathbf{x}^T\boldsymbol{l} = 0$. Therefore, a 3-space point $\mathbf{X}$ maps to $\boldsymbol{l}$ if, and only if, $\mathbf{X}^TP^T\boldsymbol{l} = 0$. This means that $\mathbf{X}$ lies on the plane $\mathbf{\Pi} = P^T\boldsymbol{l}$, the back projection of line $\boldsymbol{l}$.

3.2.3 On planes

The imaging equation $\mathbf{x} = P\mathbf{X}$ is a map from a point in the world coordinate system to a point in the image coordinates. We are free to choose the world coordinate system. Suppose we choose it so that the XY-plane corresponds to a plane $\mathbf{\Pi}$ in the scene. This means points on the scene plane $\mathbf{\Pi}$ have zero Z-coordinates. Denoting the columns of P as $\mathbf{p}_i$, $i = 1 \ldots 4$, the image of a point on $\mathbf{\Pi}$ is given by

$$\mathbf{x} = P\mathbf{X} = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.$$

This shows that the map between points $\mathbf{x}_{\Pi} = [X, Y, 1]^T$ on $\mathbf{\Pi}$ and their image $\mathbf{x}$ is a simple planar homography (plane to plane projective transformation), $\mathbf{x} = H\mathbf{x}_{\Pi}$. Where H is a 3×3 matrix of rank 3 (rank 3 is guaranteed since P is rank 3).

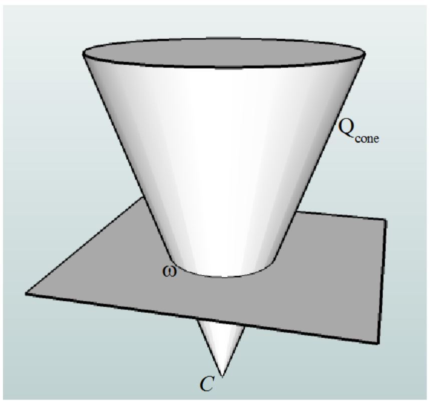


Figure 3.4: A conic ω back-projects to a degenerate quadric (cone) Q_{cone} with camera center C as the cone vertex (null space).

3.2.4 On conics

We now show that a conic ω back-projects under P to a cone. A cone is a degenerate quadric Q, i.e. a quadric that does not have full rank,

$$Q = P^T \omega P.$$

The projection is illustrated in Figure 3.4.

A point $\mathbf{x}$ lies on ω if $\mathbf{x}^T \omega \mathbf{x} = 0$. A 3-space point $\mathbf{X}$ maps to $\mathbf{x} = P\mathbf{X}$ which lies on the conic ω if $\mathbf{X}^T P^T \omega P \mathbf{X} = 0$. Thus, if we take that $Q = P^T \omega P$ does represents a quadric, then $\mathbf{X}$ lies on the quadric if $0 = \mathbf{X}^T Q \mathbf{X}$ with $\mathbf{X}^T Q \mathbf{X} = \mathbf{X}^T P^T \omega P \mathbf{X}$. It follows that $\mathbf{X}$ lies on the quadric if, and only if, $\mathbf{X}$ maps to a point on the conic so that $\mathbf{Q}$ is the back projection of conic ω.

Note that the null-vector, or cone vertex, of Q is the camera center $\mathbf{C}$ since

$$Q\mathbf{C} = P^T \omega \left(P\mathbf{C}\right) = \mathbf{0}.$$

3.2.5 On quadrics

Once more we show that the forward projection of dual quadric Q^* is the dual conic ω^* given by

$$\omega^* = P Q^* P^T. \tag{3.5}$$

This is derived from the observation that lines tangent to a conic ω belonging to the dual of the conic ω^*, or conic outline, satisfy $\boldsymbol{l}^T \omega^* \boldsymbol{l} = 0$. These lines back project to planes $\mathbf{\Pi} = P^T \boldsymbol{l}$, as explained in section (3.2.2), and are tangent to the quadric Q and belong to the dual Q^* of the quadric if, and only if,

$\mathbf{\Pi}^T Q^* \mathbf{\Pi} = 0$. It follows that for each line tangent to ω with back projected plane tangent to Q,

$$0 = \mathbf{\Pi}^T Q^* \mathbf{\Pi} = \boldsymbol{l}^T P Q^* P \boldsymbol{l} = \boldsymbol{l}^T \omega^* \boldsymbol{l}.$$

3.3 Perspective camera calibration and the image of the absolute conic

We now show that there exists a very important relation between the camera calibration matrix K and the projection of the absolute conic Ω_∞ in an image. It will be shown that this relation is $\omega_\infty = \left(KK^T\right)^{-1}$, where ω_∞ is a conic known as the image of the absolute conic (IAC). It is later established that this relation is essential for the auto-calibration of a camera.

First we must examine the map between the plane at infinity $\mathbf{\Pi}_\infty$ and the image plane. Points on the plane at infinity may be written as $\mathbf{X}_\infty = \left[\mathbf{d}^T, 0\right]^T$, and are imaged by the perspective camera, $P = KR\left[I \mid -\widetilde{\mathbf{C}}\right]$ as

$$\mathbf{x} = P\mathbf{X}_\infty = KR\left[I \mid -\widetilde{\mathbf{C}}\right]\begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix} = KR\mathbf{d}.$$

This shows the mapping between $\mathbf{\Pi}_\infty$ and the image plane is given by the planar homography $\mathbf{x} = H\mathbf{d}$ with

$$H = KR. \tag{3.6}$$

It is evident from (3.6) that the homography is independent of the position of the camera center, $\widetilde{\mathbf{C}}$, and depends only on the camera's internal calibration and its orientation with respect to the world coordinate system.

With the absolute conic Ω_∞ residing on $\mathbf{\Pi}_\infty$, we can compute its image under the homography H. Equation (2.8) shows that under the homography $\mathbf{x} = H\mathbf{x}$ a conic ω maps as $\omega' = H^{-T}\omega H^{-1}$. Since Ω_∞ is in canonical form, it is given by the conic $\omega = I$ on $\mathbf{\Pi}_\infty$ and is mapped to

$$\omega_\infty = (KR)^{-T} I \,(KR)^{-1} = K^{-T} R R^{-1} K^{-1} = K^{-T} K^{-1}. \tag{3.7}$$

So the image and the dual image of the absolute conic (DIAC) is given by,

$$\begin{aligned} \omega_\infty &= \left(KK^T\right)^{-1} \\ \omega_\infty^* &= KK^T. \end{aligned} \tag{3.8}$$

The image and dual image of the absolute conic is illustrated in Figure 3.5.

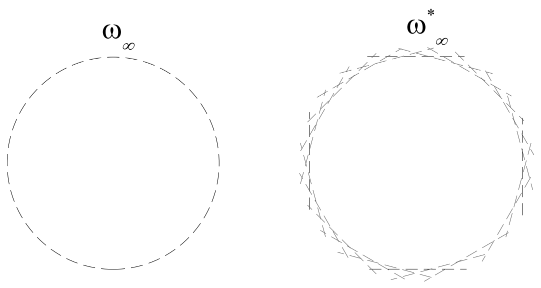


Figure 3.5: The image of the absolute conic ω_∞ and its dual ω_∞^* in the image plane.

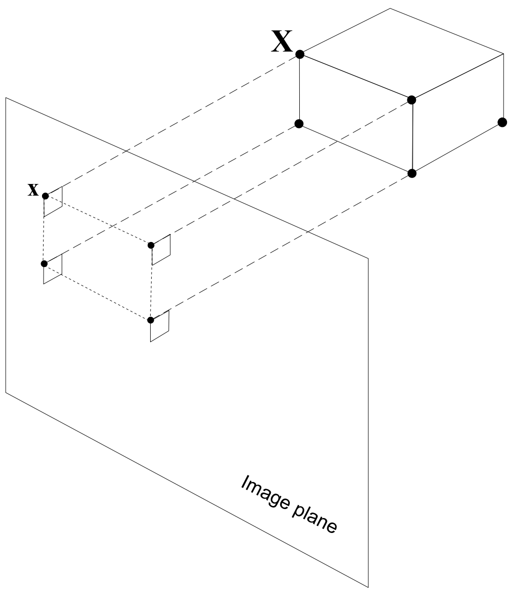


Figure 3.6: Orthographic camera projecting perpendicularly onto its image plane.

3.4 Orthographic camera, an approximation to the perspective camera

The orthographic camera model is the simplest of the affine camera models. The affine camera models are defined by a center of projection (COP) located on the plane at infinity and the last row the projection matrix P equal to [0, 0, 0, 1]. The orthographic camera model is the special case where the normal of the viewing plane (the camera direction) is parallel to one of the axes of the world coordinate system. The WCS is typically chosen so that the Z-axis is parallel to the viewing plane of the camera and thus all the 3D points project perpendicularly onto the image plane by simply dropping their camera coordinate system Z-coordinates.

The orthographic projection is illustrated in Figure 3.6. The general orthographic camera matrix is then of the form

$$P = \begin{bmatrix} \mathbf{r}_1^T & t_1 \\ \mathbf{r}_2^T & t_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.9}$$

where $\mathbf{r}_1$ and $\mathbf{r}_2$ are the first two rows of rotation matrix R, and t_1 and t_2 the first two components of translation vector $\mathbf{t}$. The orthographic camera has only 5 degrees of freedom: 3 for rotation and 2 for translation.

The reason for consideration of the orthographic camera is that it is an acceptable approximation when an object's thickness and its change in depth between frames is small relative to the object's distance from the camera. In these cases the depth recovery is difficult and very sensitive to noise, making the orthographic camera more reliable than some more complex models.

Chapter 4

Matrix factorization method for projective reconstruction

Matrix factorization as a means for 3D reconstruction from a video sequence was originally introduced by Tomasi and Kanade [34]. They developed it for the case of a single object, motion within a single plane, and an orthographic camera. Subsequently, Tomasi and Kanade [35, 33, 36] extended the planar motion case to one with arbitrary 3D motion for a single object under orthographic projection. With this change to arbitrary motion, they extended the camera coordinate system only formulation by introducing an additional world coordinate system. All the feature vectors were now measured relative to a world origin. This algorithm has been used as the basis for a number of algorithms and adapted to different types of motions, number of objects and camera configurations.

4.1 Problem statement

Suppose we have a set of $i = 1, \ldots, N$ $3D$ features visible in $f = 1, \ldots, F$ perspective images as image measurements. All measured features in this work are points, i.e. not lines or curves. The projection of a 3D point i in image f is visible in Figure 4.1. The goal is to recover the 3D structure, i.e. the 3D coordinates (in the world coordinate system) of features, and motion from the measured features.

We find it convenient to think of, and formulate the problem accordingly, a stationary camera observing a moving scene. It simplifies some of the representations and is easily expandable to the case of multiple moving objects that will be encountered later. Of course in reality it is only the relative motion

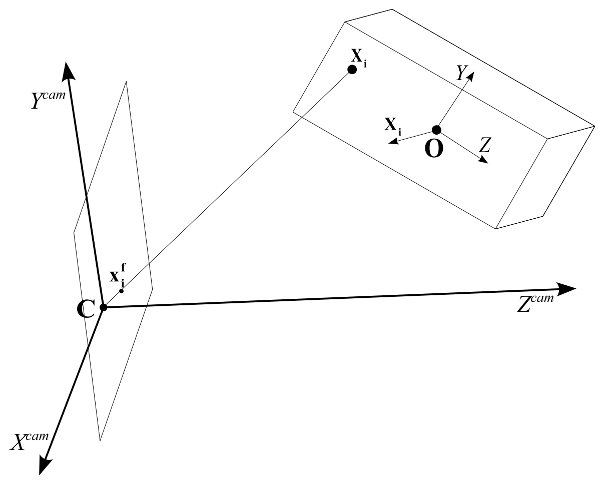


Figure 4.1: The camera and the object coordinate system with their respective coordinate centers $\mathbf{C}$ and $\mathbf{O}$. Object point $\mathbf{X}_i$ project to image point $\mathbf{x}_{fi}$ in frame f.

between camera and object that matters. A simple description of a static camera observing a scene with a moving object, is with two coordinate systems. A static camera coordinate system (CCS) with its origin the camera center, and a moving world coordinate system fixed to the object, the object coordinate system (OCS). The OCS origin is located at the object centroid. Both these coordinate systems are shown in Figure 4.1.

We denote 3D features on the object with $\mathbf{X}_i = [X_i, Y_i, Z_i, W_i]^T$, $i = 1, \ldots, N$. If $\mathbf{x}_i^f$ is the projection of $\mathbf{X}_i$ onto the f-th frame then

$$\mathbf{x}_i^f = P^f \mathbf{X}_i. \tag{4.1}$$

where P^f is the (unknown) projection matrix. In fact, given $\mathbf{x}_i^f$, $f = 1, \ldots, F$ and $i = 1, \ldots, N$, the problem is to recover P^f, $f = 1, \ldots, F$ and $\mathbf{X}_i$, $i = 1, \ldots, N$.

At the first level the problem is straightforward. Gathering $\mathbf{x}_i^f$ for all 3D features and frames in a big measurement matrix W, it should be possible to write (4.1) for all 3D features and frames as the factorization of W,

$$W = MS.$$

Where M is a matrix containing the motion information and S a matrix containing the 3D structure information. Thus the problem amounts to a matrix factorization.

This however, is not straightforward for a perspective projection. Therefore we begin with explaining the relatively straightforward situation of feature projection under an orthographic camera.

4.2 Orthographic camera

Before examining the projective matrix factorization algorithm, we first analyze the simpler case of the orthographic camera. This conveys the primary ideas in a simpler setting.

Suppose we take N features, $\mathbf{x}_i = [x_i,\, y_i,\, 1]^T$, for $i = 1, \ldots, N$, projected according to (4.1) with P the projection matrix of an orthographic camera given in (3.9). The feature vectors are all normalized so that the last element is equal to one, therefore they equal the actual image measurements. These features are tracked over F frames and collected in a single $2F \times N$ measurement matrix of the form

$$W = \begin{bmatrix} x_1^1 & \cdots & x_N^1 \\ y_1^1 & \cdots & y_N^1 \\ \vdots & & \vdots \\ x_1^F & \cdots & x_N^F \\ y_1^F & \cdots & y_N^F \end{bmatrix}.$$

Each column of matrix W is then the trajectory of one feature over the frame sequence and each grouping of two rows, all the features for that frame. We can then represent W as the matrix product

$$\begin{aligned} W &= MS \\ M &= \begin{bmatrix} \boldsymbol{\rho}_1^{1T} \\ \boldsymbol{\rho}_2^{1T} \\ \vdots \\ \boldsymbol{\rho}_1^{FT} \\ \boldsymbol{\rho}_2^{FT} \end{bmatrix} \\ S &= \begin{bmatrix} \mathbf{X}_1, & \cdots, & \mathbf{X}_N \end{bmatrix}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\rho}_i^{f\,T}$, $i = 1, \ldots, 2$ and $f = 1, \ldots, F$, is the i-th row of the f-th projection matrix. The $2F \times 4$ matrix M and $4 \times N$ matrix S now provide a compact representation of the reconstructed motion and structure, respectively.

4.2.1 Structure and motion from factorization

Since M is $2F \times 4$ and S is $4 \times N$, W has a maximum rank of 4. In reality, W is constructed from noisy measurements, e.g. noise in the feature tracking, and its rank can be much higher. The closest rank-4 matrix to noisy W in the Frobenius norm can be obtained from singular value decomposition[1] (SVD).

The SVD decomposes W as $W = U\Sigma V^T$. By denoting Σ_4 as the 4×4 diagonal matrix of the four biggest singular values, revealing the most of the matrix, U_4 as the $2F \times 4$ truncated left singular matrix and V_4^T as the $4 \times N$ truncated right singular matrix,

$$W_4 = U_4 \Sigma_4 V_4^T.$$

Matrix W_4 is now the closest rank 4 approximation to noisy W. Denoting

$$\begin{aligned} \widehat{S} &= U_4 \Sigma_4^{\frac{1}{2}} \\ \widehat{M} &= \Sigma_4^{\frac{1}{2}} V_4^T, \end{aligned}$$

we have a structure and motion factorization

$$W_4 = \widehat{M}\widehat{S}$$

that correspond to (4.2).

Unfortunately this factorization is not the ideal, since for any 4×4 non-singular matrix H, $M' = \widehat{M}H$ and $S' = H^{-1}\widehat{S}$ is also a possible solution as

$$M'S' = \widehat{M}HH^{-1}\widehat{S} = W_4.$$

Since the orthographic camera is an affine camera, this means the reconstruction is only an affine reconstruction, i.e. reconstructed up to an affine transformation from the desired metric reconstruction.

Fortunately for the orthographic case the calculation of the exact solution, exact up to a similarity transformation from the original, is reasonably simple. Constraints on the metric motion matrix M can be used to determine the rectifying homography H.

4.2.2 Exact solution

We require the rectifying homography H to upgrade the affine reconstruction $W_4 = \widehat{M}\widehat{S}$ to the metric reconstruction $W_4 = MS = \left(\widehat{M}H\right)\left(H^{-1}\widehat{S}\right)$. We denote the rectifying homography H as the concatenation of two blocks,

$$H = \left[H_R | \mathbf{h}_t\right].$$

[1]For more information consult [28]

The first submatrix H_R is a 4×3 matrix related to the rotational component and the second block $\mathbf{h}_t$ is a vector related to translation. Since

$$M = \widehat{M}H = \left[\widehat{M}H_R|\widehat{M}\mathbf{h}_t\right], \tag{4.3}$$

we can impose separate constraints on the rotation and translation component to solve for H.

Rotation constraints

From (4.2),

$$M = \begin{bmatrix} \boldsymbol{\rho}_1^{1T} \\ \boldsymbol{\rho}_2^{1T} \\ \vdots \\ \boldsymbol{\rho}_1^{FT} \\ \boldsymbol{\rho}_2^{FT} \end{bmatrix} \text{ with } \begin{bmatrix} \boldsymbol{\rho}_{1_f}^{fT} \\ \boldsymbol{\rho}_2^{fT} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^{fT} & t_1^f \\ \mathbf{r}_2^{fT} & t_2^f \end{bmatrix}, \text{ for } f = 1, \ldots, F.$$

Since rotation is orthonormal,

$$\begin{bmatrix} \mathbf{r}_1^{fT} \\ \mathbf{r}_2^{fT} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^f & \mathbf{r}_2^f \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ for } f = 1, \ldots, F \tag{4.4}$$

where $\mathbf{r}_1^T$, $\mathbf{r}_2^T$ are the first and second row of rotation matrix R.

Combining (4.3) and (4.4) yields

$$\begin{aligned} \widehat{\mathbf{m}}_{2j-1}^T H_R H_R^T \widehat{\mathbf{m}}_{2j-1} &= 1 \quad \forall j = 1, \ldots, F \\ \widehat{\mathbf{m}}_{2j}^T H_R H_R^T \widehat{\mathbf{m}}_{2j} &= 1 \quad \forall j = 1, \ldots, F \\ \widehat{\mathbf{m}}_{2j-1}^T H_R H_R^T \widehat{\mathbf{m}}_{2j} &= 0 \quad \forall j = 1, \ldots, F \end{aligned}$$

where $\widehat{\mathbf{m}}_j^T$ is row j of matrix M. This over-constrained system can be solved for entries of the 4×4 matrix $H_R H_R^T$.

To obtain H_R from $H_R H_R^T$ we employ the SVD. Note that $H_R H_R^T$ has rank 3 since H_R is 4×3 and is symmetric. Therefore, given

$$H_R H_R^T = U_3 \Sigma_3 V_3^T,$$

$$H_R = U_3 \Sigma_3^{\frac{1}{2}}.$$

Translation constraints

Under the orthographic camera model, the projection of the object centroid corresponds to the centroid of the projected object. Therefore,

$$M\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}_i,$$
$$\text{i.e.}\quad M\bar{X} = \overline{\mathbf{w}},$$

where $\mathbf{w}_i$ is the i-th column of W_4. Since the object coordinate system has an arbitrary origin, we can choose

$$\bar{X} = \left[\mathbf{0}^T, 1\right]^T$$

which corresponds to centering the object around the object coordinate system origin. It follows from (4.3) that

$$\widehat{M}\mathbf{h}_t = \left[\widehat{M}H_R|\widehat{M}\mathbf{h}_t\right]\begin{bmatrix}\mathbf{0}\\1\end{bmatrix} = \overline{\mathbf{w}}$$

This is an over-constrained system from which we solve $\mathbf{h}_t$,

$$\mathbf{h}_t = \left(\widehat{M}^T\widehat{M}\right)^{-1}\widehat{M}^T\overline{\mathbf{w}}$$

where $\widehat{M}^+ = \left(\widehat{M}^T\widehat{M}\right)^{-1}\widehat{M}^T$ is the pseudo-inverse of $\widehat{M}$. This completes the calculation of rectifying homography H and therefore metric reconstructed M and S.

4.3 Perspective camera

Now we switch to the perspective camera. Recall our matrix factorization $W = MS$, written component-wise as

$$\mathbf{x}_i^f = P^f\mathbf{X}_i, \tag{4.5}$$

for $i = 1, \ldots, N$ and $f = 1, \ldots, F$. The projection matrix P is now the perspective camera matrix as defined in (3.2). Even though the image projection equation looks similar to the orthographic case, the Tomasi Kanade orthographic factorization algorithm can not be applied directly to the perspective camera. Since the perspective camera is a projective camera, this method is called projective matrix factorization.

Working in homogeneous coordinates, all coordinates are defined up to nonzero scale factors. These scale factors are unknown but actually need to be taken

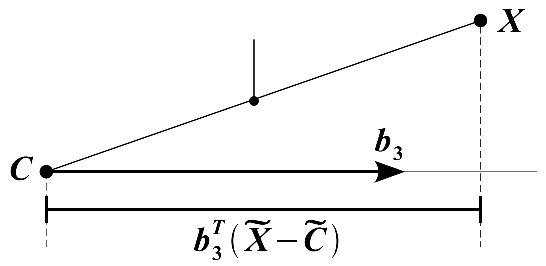


Figure 4.2: If the camera matrix $P = [B \,|\, \mathbf{p}_4]$ is normalized so that $\|\mathbf{b}_3\| = 1$ and $\det B > 0$, and $\mathbf{x} = \lambda\,[x,\,y,\,1]^T = P\mathbf{X}$, where $\mathbf{X} = [X,\,Y,\,Z,\,1]^T$, then λ is the depth of the 3D feature $\mathbf{X}$ from the camera center in the direction of the principal ray, $\mathbf{b}_3$, of the camera.

into account when we place the multiple perspective projection equations in the matrix factorization form of $W = MS$. The reason is that the scale factors values are now dependent on both a 3D feature and a perspective projection matrix, with all of the scale factors for the projected features in a single frame dependent on the scaling of the same projection matrix and the scale factors for the projection of a 3D feature across all of the frames dependent on the scaling of a single 3D feature. Extracting these unknown scale factors and explicitly writing them as λ_i^f, the image feature vectors are now redefined with their last coordinate equal to unity, $\mathbf{x}_i^f = \left[x_i^f,\,y_i^f,\,1\right]^T$. The x_i^f and y_i^f values now represent the actual measured image coordinates. We now have (4.5) in the explicit form

$$\begin{aligned} P^f\mathbf{X}_i &= \lambda_i^f \begin{bmatrix} x_i^f \\ y_i^f \\ 1 \end{bmatrix} \\ &= \lambda_i^f \mathbf{x}_i^f. \end{aligned}$$

We now show that with a specific normalization these scale factors have a specific meaning, they are the depths of the $3D$ features referred to as *projective depths.*

4.3.1 Depth of features

We examine the camera illustrated in Figure 4.2 with camera center $\mathbf{C} = \left[\widetilde{\mathbf{C}}^T,\,1\right]^T$ and camera matrix written in the form $P = [B \,|\, \mathbf{p}_4]$, projecting a WCS feature $\mathbf{X} = [X,\,Y,\,Z,\,1]^T = \left[\widetilde{\mathbf{X}}^T,\,1\right]^T$ to image feature $\mathbf{x} = \lambda\,[x,\,y,\,1]^T = P\mathbf{X}$.

Then for $\boldsymbol{\rho}_3^T$ the 3rd row of P

$$\lambda = \boldsymbol{\rho}_3^T\mathbf{X} = \boldsymbol{\rho}_3^T\left(\mathbf{X} - \mathbf{C}\right),$$

since $P\mathbf{C} = \mathbf{0}$ for the camera center. It now follows that

$$\lambda = \boldsymbol{\rho}_3^T (\mathbf{X} - \mathbf{C}) = \mathbf{b}_3^T \left(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}} \right),$$

with $\mathbf{b}_3^T$ the third row of B representing the principal ray direction. Therefore, λ can be interpreted as the dot product between the principal ray direction and the depth of the feature $\mathbf{X}$ from the camera center $\mathbf{C}$. If the camera matrix is normalized so that $\det B > 0$ and $\|\mathbf{b}_3\| = 1$, $\mathbf{b}_3$ is a unit vector pointing in the positive axial direction, then λ is the actual projective depth of feature $\mathbf{X}$ from camera center $\mathbf{C}$ in the positive principal ray direction.

For the sake of completeness, we present the general case. For any homogeneous WCS feature $\mathbf{X} = [X,\, Y,\, Z,\, W]^T = W\left[\frac{X}{W},\, \frac{Y}{W},\, \frac{Z}{W},\, 1\right]^T = W\left[\widetilde{\mathbf{X}}^T,\, 1\right]^T$,[2] and for a camera matrix P that is not normalized, the projective depths are defined as

$$\lambda = \boldsymbol{\rho}_3^T (\mathbf{X} - \mathbf{C}) = W\mathbf{b}_3^T \left(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}} \right).$$

The actual depth of a general $\mathbf{X}$ in front of the principal plane now becomes

$$\operatorname{depth}(\mathbf{X}; P) = \frac{\operatorname{sign}(\det B)\, \lambda}{W \left\|\mathbf{b}_3^T\right\|}. \tag{4.6}$$

4.3.2 Projective factorization

With the camera matrix $P = [B \,|\, \mathbf{p}_4]$ normalized so that $\left\|\mathbf{b}_3^T\right\| = 1$ and $\det B > 0$, $\mathbf{x} = [x,\, y,\, 1]^T$ and $\mathbf{X} = [X,\, Y,\, Z,\, 1]^T$ the new measurement matrix W is of size $3F \times N$ and has the form,

$$\begin{aligned} W &= \begin{bmatrix} \lambda_1^1 \mathbf{x}_1^1 & \lambda_2^1 \mathbf{x}_2^1 & \cdots & \lambda_N^1 \mathbf{x}_N^1 \\ \lambda_1^2 \mathbf{x}_1^2 & \lambda_2^2 \mathbf{x}_2^2 & \cdots & \lambda_N^2 \mathbf{x}_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^F \mathbf{x}_1^F & \lambda_2^F \mathbf{x}_2^F & \cdots & \lambda_N^F \mathbf{x}_N^F \end{bmatrix} \\ &= MS, \end{aligned} \tag{4.7}$$

with the the unknown projective depths λ_i^f the actual depths from camera center $\mathbf{C}$ in the positive direction of the principal ray. The matrices resulting from the

[2]Note that here W is the last component of WCS feature $\mathbf{X} = [X,\, Y,\, Z,\, W]^T$ and not the measurement matrix.

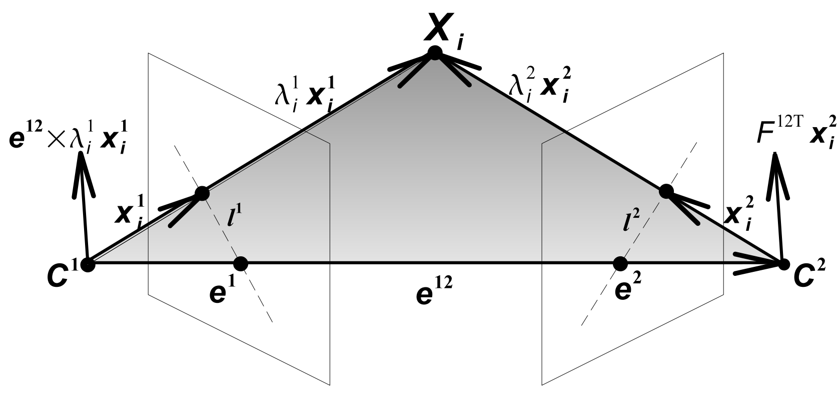


Figure 4.3: Motivation for the projective depth equation.

factorization are still mostly similar to the orthographic case,

$$
\begin{aligned}
M &= \begin{bmatrix} P^1 \\ P^2 \\ \vdots \\ P^F \end{bmatrix} \\
S &= \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_N \end{bmatrix}
\end{aligned}
$$

with motion matrix M now $3F \times 4$ and structure matrix S still $4 \times N$.

It is clear from (4.7) that before factorization is possible these unknown projective depths need to be determined.

4.3.3 Projective depth estimation

To recover the projective depths we give an intuitive argument. Visualize all image features as $3D$ vectors embedded in Euclidean space, e.g. $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$ in Figure 4.3. In Figure 4.3 we have two cameras $\mathbf{C}^1$ and $\mathbf{C}^2$ with fundamental matrix $\mathcal{F}^{12}$, both observing Euclidean WCS feature $\mathbf{X}_i$.

The fundamental matrix $\mathcal{F}$ is a rank-2, 3×3 matrix which relates corresponding points in stereo images. The line $\boldsymbol{l} = \mathcal{F}\mathbf{x}^f$ describes the epipolar line on which the corresponding point $\mathbf{x}^g$ must lie in the other image. Therefore, for all pairs of points it holds that $\mathbf{x}^{g\,T} \mathcal{F}\mathbf{x}^f = 0$.

$\mathbf{X}_i$ is projected in image 1 to a feature represented by the $3D$ vector $\mathbf{x}_i^1$ in the coordinate system centered at $\mathbf{C}^1$, and in image 2 as the feature with $3D$ vector $\mathbf{x}_i^2$ in a coordinate system centered at $\mathbf{C}^2$. The baseline vector $\mathbf{e}^{12} = \widetilde{\mathbf{C}}^1 - \widetilde{\mathbf{C}}^2$ is also displayed, along with its two image intersecting epipoles (intersection of the baseline with the image plane) $\mathbf{e}^1$ and $\mathbf{e}^2$ and the epipolar line in each image, $\boldsymbol{l}^1$ and $\boldsymbol{l}^2$. With the correct projective depths, λ_i^1 and λ_i^2, $\mathbf{X}_i = \lambda_i^1 \mathbf{x}_i^1$

and $\mathbf{X}_i = \lambda_i^2 \mathbf{x}_i^2$ ($\mathbf{X}_i$ expressed in the appropriate coordinate system). $\lambda_i^1 \mathbf{x}_i^1$and $\lambda_i^2 \mathbf{x}_i^2$ along with $\mathbf{e}^{12}$ now form a closed triangle on their projective plane, with $\mathbf{X}_i$ the triangle apex.

The vectors $\mathbf{e}^{12} \times \mathbf{x}_i^1$ and $\mathbf{e}^{12} \times \mathbf{x}_i^2$ are normal to the epipolar plane and, with $\mathbf{e}^{12}$ and $\lambda_i^1 \mathbf{x}_i^1$ scaled as they are, the area of the epipolar plane triangle is given by $\frac{1}{2} \left|\mathbf{e}^{12} \times \left(\lambda_i^1 \mathbf{x}_i^1\right)\right|$. On the other hand, it follows from epipolar geometry that the image 1 epipolar line $\boldsymbol{l}^1$, the projection of 3-space line $\mathcal{F}^{12^T}\left(\lambda_i^2 \mathbf{x}_i^2\right)$, is the projection of the epipolar plane in image 1. Since any feature $\left[x_{i,}^1\, y_i^1\right]^T$ on the epipolar line in image 1 is normal to the image 1 epipolar line $\boldsymbol{l}^1$ (i.e. $\left[x_i^1,\, y_2^1\right] \boldsymbol{l}^1 = 0$), and the $3D$ vectors representing all the features on the epipolar line lie on the projective plane - the vector $\mathcal{F}^{12^T}\left(\lambda_i^2 \mathbf{x}_i^2\right)$ is also normal to the epipolar plane.

Thus $\mathcal{F}^{12^T}\left(\lambda_i^2 \mathbf{x}_i^2\right)$ and $\mathbf{e}^{12} \times \left(\lambda_i^1 \mathbf{x}_i^2\right)$ are proportional, since they are in the same direction. Then for the correct scaling of $\mathcal{F}^{12}$, they also have the same size, and the area of the triangle is given by both $\frac{1}{2} \left|\mathbf{e}^{12} \times \left(\lambda_i^1 \mathbf{x}_i^1\right)\right|$ and $\frac{1}{2} \left|\mathcal{F}^{12^T}\left(\lambda_i^2 \mathbf{x}_i^2\right)\right|$. The relation

$$\left(\mathcal{F}^{12^T} \mathbf{x}_i^2\right) \lambda_i^2 = \left(\mathbf{e}^{12} \times \mathbf{x}_i^1\right) \lambda_i^1$$

follows, which can be rewritten for the general case of frame f and g,

$$\left(\mathcal{F}^{fg^T} \mathbf{x}_i^g\right) \lambda_i^g = \left(\mathbf{e}^{fg} \times \mathbf{x}_i^f\right) \lambda_i^f. \tag{4.8}$$

Note that in projective space the vectors in Figure 4.3 have meaning and that (4.8) holds.

From (4.8) two observations can be concluded:

- Equality up to scale: The epipolar line in image f from $\mathcal{F}^{fg^T} \mathbf{x}_i^g$ is the line through the feature $\mathbf{x}_i^f$ and the epipole $\mathbf{e}^f$ in image f. This is just the standard epipolar constraint.

- Equality of factors: if the correct projective depths are used along with the correct scaling of $\mathcal{F}^{fg}$ and $\mathbf{e}^{fg}$, the two terms are exactly equal (not just up to scale). In this case the equation allows us to solve the projective depths in projective space using fundamental matrices and epipoles.

Solving equation (4.8) in terms of λ_i^g,

$$\lambda_i^g = \frac{\left\|\mathbf{e}^{fg} \times \mathbf{x}_i^f\right\|^2}{\left(\mathbf{e}^{fg} \times \mathbf{x}_i^f\right)^T \left(\mathcal{F}^{fg^T} \mathbf{x}_i^g\right)} \lambda_i^f. \tag{4.9}$$

This equation can be recursively chained together to give estimates for the complete set of depths for feature i for all of the required frames, requiring a known initial value, e.g. a known λ_i^1.

The strategy is then to simulate a chain of $F-1$ stereo cameras for all of the F frames. This allows for the estimation of the minimum number of fundamental matrices and epipoles to create the necessary set of homogeneous equations to solve all the projective depths. The fundamental matrices are calculated from corresponding feature pairs in a stereo view. RANSAC estimation is used to calculate the fundamental matrices in this work. The epipoles are then calculated from its property as the nullspace of a fundamental matrix. The stereo camera chain setup can be either sequential or parallel. With a sequential setup we sequentially pair frames starting from the first two and ending with the last two so that we have the set of unique fundamental matrices $\{\mathcal{F}^{ij}\}$ for $i = 1, \ldots, F-1$ and $j = 2, \ldots, F$. With a parallel setup, a single reference frame is chosen and each each frame is paired with the reference frame (except the reference frame with itself) e.g. with the first frame the reference frame we have the set of fundamental matrices $\{\mathcal{F}^{1j}\}$ for $j = 2, \ldots, F$. Solving the chains of equation (4.9) for all N features in each frame we have a complete set of projective depths up to an overall scale factor. We still require a starting projective depth for the chain though - here we choose an arbitrary initial value of $\lambda_i^1 = 1$.

There is however one flaw in this argument. For an exact equality in our equations we require a correct scaling of fundamental matrix F^{ij} and $\mathbf{e}^{fg}$ and since we have no knowledge of the camera matrices or the distance between the camera centers, they can be recovered only up to an unknown scale factor. So we don't know the scale factors in equation (4.8) and (4.9) after all!

Fortunately this does not turn out to be a problem. When our minimal set of $N(F-1)$ non-redundant depth recovery equations are used, the overall scale factor for each image absorbs the relative scale of the fundamental matrix $\mathcal{F}$ and epipole $\mathbf{e}$ used for that image. Note that the arbitrary overall scale factor also absorbs our arbitrary initial value for $\lambda_i^1 = 1$.

4.3.4 Structure and motion from factorization

Once we have obtained the correctly scaled projective depths, the measurement matrix W of equation (4.7) should in theory again have a maximum rank of 4. The SVD factorization, similarly to the Tomasi and Kanade orthographic factorization method, can now be applied, $W = U\Sigma V^T$. The best rank-4 approximated projective measurement matrix in the Frobenius norm is given by

$$W_4 = U_4 \Sigma_4 V_4^T$$

with Σ_4 the 4×4 diagonal matrix of the four biggest singular values, U_4 the $3F \times 4$ truncated left singular matrix and V_4^T the $4 \times N$ truncated right singular matrix. W_4 is now factorised,

$$\begin{aligned} &W_4 = \widehat{M}\widehat{S} \text{ with} \\ &\widehat{M} = U_4 \Sigma_4^{\frac{1}{2}} \\ &\widehat{S} = \Sigma_4^{\frac{1}{2}} V_4^T \ , \end{aligned} \tag{4.10}$$

Unfortunately, just as in the orthographic case the factorization is *not* unique. For any 4×4 invertible matrix H,

$$M'S' = \left(\widehat{M}H\right)\left(H^{-1}\widehat{S}\right) = \widehat{M}\widehat{S} = W_4,$$

is also a possible solution. Unlike orthographic projection case, the perspective camera is a projective camera and the reconstruction is now only up to projective transformation (not affine). This time the solution is not as simple as merely applying the motion matrix M constraints. Determining the rectifying homography H for the projective to the metric reconstruction is a complex process thoroughly discussed in the next chapter.

4.4 Matrix factorization shortcomings

The Tomasi Kanade matrix factorization has a number of shortcomings, mostly as a result of the use of a single measurement matrix for all the image features in all the frames:

- The set of image features that are reconstructed must be visible in all the frames, i.e. it can not handle occlusion. This can be partially overcome by splitting the image sequence into subsequences.
- Additional images or image features cannot be added recursively.
- The projective factorization method can only reconstruct up to an arbitrary projective transformation.

4.5 Algorithm outline

The matrix factorization method steps are summarized in Algorithm 4.1. The complexity of the method is dominated by the SVD computation which is $\mathcal{O}\left((m+n)^3\right)$.

Algorithm 4.1 Projective matrix factorization algorithm

Objective

Given a set of N features over F perspective images $\left\{\mathbf{x}_i^f\right\}$, compute a projective reconstruction $\left\{P^f, \mathbf{X}_i\right\}$ of the image features.

Algorithm

```
1: Simulate $F-1$ stereo cameras and estimate the fundamental matrices and
   epipoles.
2: Determine the scale factors $\lambda_i^f$ using equation (4.9).
3: Build the rescaled measurement matrix $W$.
4: Compute the SVD of the measurement matrix $W$.
5: From the SVD, compute projective shape and motion.
```

Chapter 5

Auto-calibration method for metric rectification

Auto-calibration or self-calibration is the process of determining internal camera parameters directly from multiple uncalibrated images, see e.g. [13]. The idea of auto-calibrating a camera originated with Faugeras, Luong and Maybank [6] and avoids the onerous task of calibrating cameras using special calibration objects. In our application it allows us to upgrade, or rectify, our projective factorization, e.g. obtained in chapter 4, to the desired metric factorization that is within a similarity transformation of the original scene.

It was shown in earlier chapters that the identification of the absolute conic allows for an upgrade to metric structure since it is the only conic invariant to similarity transformations of the metric stratum. It was also shown that the absolute conic projects to image of the absolute conic in each image. Based on this, the root of the auto-calibration algorithm is the fact that a camera moves rigidly, i.e. similarity transformations, and so the absolute conic is fixed under camera motion when it is projecting in each image. Conversely, if a single conic can be determined from the intersection of cones back-projecting from a conic in each image, this conic is the absolute conic and the rectifying homography can be inferred.

We proceed to first state the auto-calibration problem and then solve it with an algorithm based on the absolute dual quadric.

5.1 Problem statement

We have a projective reconstruction $\left\{P^f, \mathbf{X}_i\right\}$ for a video sequence of $f = 1, \ldots, F$ frames and $i = 1, \ldots, N$ features. We wish to determine the rectifying homography H such that $\left\{P^f H,\, H^{-1}\mathbf{X}_i\right\} = \left\{P^f_M,\, \mathbf{X}_{Mi}\right\}$ is the metric reconstruction. It will be shown that the solution requires placing known constraints, for $f = 1 \ldots F$, on camera calibration matrix K^f where $P^f_M = K^f \left[R^f \,|\, \mathbf{t}^f\right] = P^f H$. These constraints are related to constraints on the inverse of the projection of the absolute conic in an image, the dual image of the absolute conic (DIAC) ω^*_∞(and image of the absolute conic (IAC) ω^f_∞). This leads to equations on the known projective cameras P^f that solve the unknown homography H.

We begin by presenting the required form of the rectifying homography. This is followed by deriving the auto-calibration equations necessary for solving the homography.

The rectifying homography

The perspective projection matrices P^f, $\mathbf{x}^f_i = P^f\mathbf{X}_i$, are related to metric P^f_M, where $\mathbf{x}^f_i = P^f_M\mathbf{X}_{Mi}$ with

$$P^f_M = P^f H, \text{ for } f = 1 \ldots F, \tag{5.1}$$

where H is an unknown 4×4 homography of 3-space.

Since we desire a metric reconstruction, we are not concerned with the absolute rotation, translation or scale of the reconstruction. The OCS is chosen to coincide with the first camera, so that $R^1 = I$ and $\mathbf{t}^1 = \mathbf{0}$. Then R^f and $\mathbf{t}^f$ determine the metric transformation between the f-th camera and the first, with $P^1_M = K^1 \left[I \,|\, \mathbf{0}\right] = \left[K^1 \,|\, \mathbf{0}\right]$. Similarly, for the projective reconstruction, we choose the canonical camera for the first view, so that $P^1 = [I \,|\, \mathbf{0}]$.

H is a projective transformation, $H : \mathbb{P}^3 \rightarrow \mathbb{P}^3$, with the general form (see (2.10)),

$$H = \begin{bmatrix} A & \mathbf{b} \\ \mathbf{v}^T & k \end{bmatrix}.$$

From (5.1)

$$\begin{aligned} P^1_M &= \left[K^1 \,|\, \mathbf{0}\right] \\ &= [I \,|\, \mathbf{0}]\, H \\ &= [I \,|\, \mathbf{0}] \begin{bmatrix} A & \mathbf{b} \\ \mathbf{v}^T & k \end{bmatrix} \\ &= [A \,|\, \mathbf{b}], \end{aligned}$$

which implies that $A = K^1$ and $\mathbf{b} = \mathbf{0}$. In addition, since H is non-singular, k must be non-zero, so we assume $k = 1$ (this fixes the scale of the reconstruction).

This shows that H is of the form

$$H = \begin{bmatrix} K^1 & \mathbf{0} \\ \mathbf{v}^T & 1 \end{bmatrix}.$$

The plane at infinity $\mathbf{\Pi}_\infty$ has the canonical metric form of $\mathbf{e}_4 = [0,\, 0,\, 0,\, 1]^T$. Its projective representation is given by (see (2.4)),

$$\begin{aligned} \mathbf{\Pi}_\infty &= H^{-T}\mathbf{e}_4 = \begin{bmatrix} {K^1}^{-T} & -\,{K^1}^{-T}\mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -\,{K^1}^{-T}\mathbf{v} \\ 1 \end{bmatrix}. \end{aligned} \tag{5.2}$$

Suppose we write $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T,\, 1\right]^T$, then, according to (5.2), $\boldsymbol{\pi} = -\,{K^1}^{-T}\mathbf{v}$ and similarly $\mathbf{v}^T = -\boldsymbol{\pi}^T K^1$.

In summary,

$$H = \begin{bmatrix} K^1 & \mathbf{0} \\ -\boldsymbol{\pi}^T K^1 & 1 \end{bmatrix}, \tag{5.3}$$

where K^1 is the upper-triangular camera calibration matrix of the first camera and the plane at infinity in the projective reconstruction is given by $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T,\, 1\right]^T$.

It follows from (5.3) that to transform a projective reconstruction to metric requires 8 parameters (8 degrees of freedom), 3 entries for $\boldsymbol{\pi}$ and 5 entries for K. This agrees with the requirements of finding metric structure - specifying the plane at infinity and the absolute conic, which have 3 and 5 degrees of freedom respectively.

The basic auto-calibration equations

We denote the known projective reconstruction cameras with $P^f = \left[A^f \,|\, \mathbf{b}^f\right]$. From (5.1) and (5.3) it follows that

$$\begin{aligned} P^f_M &= P^f H \\ &= \begin{bmatrix} A^f & | & \mathbf{b}^f \end{bmatrix} \begin{bmatrix} K^1 & \mathbf{0} \\ -\boldsymbol{\pi}^T K^1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} A^f K^1 - \mathbf{b}^f \boldsymbol{\pi}^T K^1 & | & \mathbf{b}_f \end{bmatrix}. \end{aligned}$$

If we now write $P_M^f = K^f \left[R^f \,|\, \mathbf{t}^f\right] = \left[K^f R^f \,|\, K^f \mathbf{t}^f\right]$, we obtain,

$$K^f R^f = \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right) K^1.$$

We can eliminate R^f by multiplication with $\left(K^f R^f\right)^T$from the right,

$$\begin{array}{rcl} K^f \, {K^f}^T & = & K^f R^f \, {R^f}^T \, {K^f}^T \\ & = & \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right) K^1 \, {K^1}^T \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right)^T . \end{array}$$

In (3.8) it was noted that $\omega_\infty^{*f} = K^f \, {K^f}^T$, where ω_∞^{*f} is the DIAC (dual image of the absolute conic) in frame f. This substitution gives the basic equations for auto-calibration,

$$\begin{array}{rcl} \omega_\infty^{*f} & = & \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right) \omega_\infty^{*1} \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right) \\ \omega_\infty^{f} & = & \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right)^{-1} \omega_\infty^{1} \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right)^{-1} , \end{array} \tag{5.4}$$

remembering that $\omega_\infty^f = {\omega_\infty^{*f}}^{-1} = {K^f}^{-T} \, {K^f}^{-1}$ is the IAC (image of the absolute conic) in frame f. These equations relate the unknown entries of ω_∞^{*f} and ω_∞^f, for $f = 1 \ldots F$, and parameters of $\boldsymbol{\pi}$ with the parts A^f and $\mathbf{b}^f$ of the known projective cameras.

The art of auto-calibration is to exploit the constraints on K^f, such that one of the entries in ω_∞^{*f} or ω_∞^f is zero. This, along with (5.4), generates equations on the images of the absolute conic in different images. All auto-calibration methods are variations on solving these equations. A popular approach to solving these equations is based on the absolute dual quadric, examined in the next section.

5.2 Auto-calibration using the absolute dual quadric

A method based on the absolute dual quadric was first proposed by Heyden and Aström [14], although they did not give the geometric meaning of their constraint. Previous methods focused explicitly on the plane at infinity and the absolute conic. Triggs [37] removed scale factors as additional unknowns by writing P^1 in canonical form and suggested nonlinear solution methods. Pollefeys, Koch and van Gool [25] extended this method by showing that it could be applied to different cameras (cameras with different intrinsic calibration parameters).

The absolute dual quadric Q_∞^* was discussed in Section (2.6.3) and is a degenerate dual quadric (a cone) represented by a 4×4 symmetric, singular and homogeneous matrix of rank 3. It is the dual entity of the traditional absolute conic that is simpler to use. Since the absolute dual quadric encodes both the absolute conic Ω_∞ and plane at infinity $\mathbf{\Pi}_\infty$, required for metric geometry, it allows us to go directly from a projective to a metric reconstruction. A more stratified approach is possible with first, a projective to affine rectification based on finding $\mathbf{\Pi}_\infty$, followed by an affine to metric rectification based on finding Ω_∞. The drawback of this stratified approach is that it turns out to be an onerous task to automatically find $\mathbf{\Pi}_\infty$.

It is clear from (3.5) that Q^*_∞ projects to the DIAC ω^*_∞. If we further consider (3.8),

$$K^f\ {K^f}^T = \omega^{*f}_\infty = P^f Q^*_\infty\ {P^f}^T. \tag{5.5}$$

The idea of camera calibration using Q^*_∞ is to transfer intrinsic calibration constraints from ω^{*f}_∞ onto Q^*_∞ via the known projective reconstruction cameras P^f. In this manner the matrix representing Q^*_∞ in the projective reconstruction is determined from known constraints on K^f. We then use (5.5) with the known projection matrices to solve K^f for all $f = 1, \ldots, F$.

Once Q^*_∞ is determined, the rectifying homography H, which we ultimately seek, can also be found. It is clear from (2.9) that the absolute dual quadric in metric form,

$$\widetilde{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

transforms with H to a projective reconstruction with

$$Q^*_\infty = H\widetilde{I}H^T. \tag{5.6}$$

Note that this implies that the homography H is found through a matrix decomposition of absolute dual quadric. Thus the auto-calibration is based on specifying constraints on K^f to determine Q^*_∞, and then from Q^*_∞ determine H.

For the sake of completeness, note that if we multiply out (5.6) for H given by (5.3), then

$$Q^*_\infty = H^T\widetilde{I}H = \begin{bmatrix} K^1\ {K^1}^T & -K^1\ {K^1}^T\boldsymbol{\pi} \\ -\boldsymbol{\pi}^T K^1\ {K^1}^T & \boldsymbol{\pi}^T K^1\ {K^1}^T\boldsymbol{\pi} \end{bmatrix} = \begin{bmatrix} \omega^{*1}_\infty & -\omega^{*1}_\infty\boldsymbol{\pi} \\ -\boldsymbol{\pi}^T\omega^{*1}_\infty & \boldsymbol{\pi}^T\omega^{*1}_\infty\boldsymbol{\pi} \end{bmatrix}. \tag{5.7}$$

This clearly shows how Q^*_∞ encodes the absolute conic and plane at infinity. When combining (5.7) with (5.5) and denoting $P^f = \left[A^f \mid \mathbf{b}^f\right]$ we again obtain

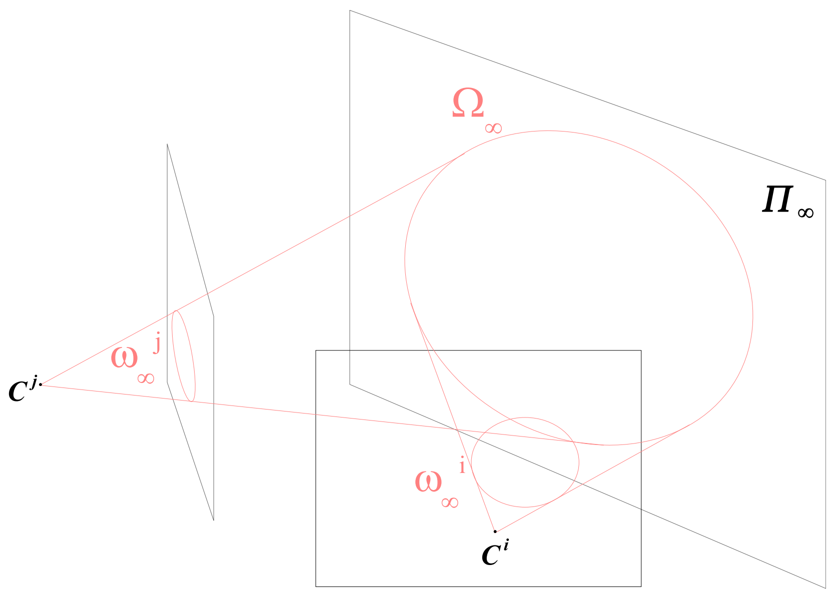


Figure 5.1: The absolute conic Ω_∞ projecting as IAC ω_∞ in images. Each projection is the result of a cone, with the camera center $\mathbf{C}$ as cone vertex, intersecting an image plane.

the auto-calibration equation of (5.4),

$$\omega_\infty^{*f} = P^f Q_\infty^* {P^f}^T = \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right) \omega_\infty^{*1} \left(A^f - \mathbf{b}^f \boldsymbol{\pi}^T\right).$$

Figure 5.1 illustrates the concept of the absolute conic and its projection in the images.

5.2.1 Constraints on the elements of Q_∞^*

The objective is to estimate Q_∞^* in the projective reconstruction from constraints on ω_∞^{*f} (or ω_∞^{f}) as given in (5.5). There are two ways to acquire these element constraints:

- Place constraints on the camera calibration parameters.
- Assume that the cameras have a constant calibration, i.e. they all have the same K.

We next discuss the two cases separately.

Camera calibration constraints

Since $\omega^*_\infty = KK^T$ and $\omega_\infty = (\omega^*_\infty)^{-1} = K^{-T}K^{-1}$, for the camera calibration matrix given by (3.3),

$$\omega^*_\infty = \begin{bmatrix} \alpha^2 + s^2 + x_0^2 & sr\alpha + x_0y_0 & x_0 \\ sr\alpha + x_0y_0 & r^2\alpha^2 + y_0^2 & y_0 \\ x_0 & y_0 & 1 \end{bmatrix} \tag{5.8}$$

and

$$\omega_\infty = \frac{1}{r^2\alpha^4}\begin{bmatrix} r^2\alpha^2 & -sr\alpha & -x_0r^2\alpha^2 + y_0sr\alpha \\ -sr\alpha & \alpha^2 + y_0^2 & r\alpha sx_0 - \alpha^2y_0 - s^2y_0 \\ -x_0r^2\alpha^2 + y_0sr\alpha & r\alpha sx_0 - \alpha^2y_0 - s^2y_0 & r^2\alpha^4 + \alpha^2y_0^2 + (r\alpha x_0 - sy_0)^2 \end{bmatrix}.$$

These matrices are simplified significantly if we assume zero skew , i.e. $s = 0$,

$$\omega^*_\infty = \begin{bmatrix} \alpha^2 + x_0^2 & x_0y_0 & x_0 \\ x_0y_0 & r^2\alpha^2 + y_0^2 & y_0 \\ x_0 & y_0 & 1 \end{bmatrix}$$

and

$$\omega_\infty = \frac{1}{r^2\alpha^2}\begin{bmatrix} r^2 & 0 & -r^2x_0 \\ 0 & 1 & -y_0 \\ -r^2x_0 & -y_0 & r^2\alpha^2 + y_0^2 + r^2x_0^2 \end{bmatrix}. \tag{5.9}$$

We can now clearly show how constraints on K imply constraints on ω^*_∞ . This leads to equations for the elements of Q^*_∞. As an example, assume the principal point is known. The image coordinate system can then be changed so that the principal point coincides with the origin, i.e. $x_0 = 0$, $y_0 = 0$. The DIAC in (5.8) now becomes

$$\omega^*_\infty = \begin{bmatrix} \alpha^2 + s^2 & sr\alpha & 0 \\ sr\alpha & r^2\alpha^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

The zero entries in (5.10) generate linear equations on Q^*_∞,

$$\left(P^f Q^*_\infty {P^f}^T\right)_{13} = 0 \text{ and}$$
$$\left(P^f Q^*_\infty {P^f}^T\right)_{23} = 0.$$

There is also the very useful case for ω_∞, in (5.9), where zero skew alone generates a linear equation.

The possible equations that arise from constraints on ω^*_∞ is summarized in

Condition	Constraint	Type	# constraints
zero skew	$\omega^*_{\infty 12}\omega^*_{\infty 33} = \omega^*_{\infty 13}\omega^*_{\infty 23}$	quadratic	F
p.p. at origin	$\omega_{\infty 13} = \omega_{\infty 23} = 0$	linear	2F
zero skew, p.p. at origin	$\omega^*_{\infty 12} = 0$	linear	F
fixed aspect ratio, zero skew, p.p. at origin	$\frac{\omega^{*f}_{\infty 11}}{\omega^{*f}_{\infty 22}} = \frac{\omega^{*g}_{\infty 11}}{\omega^{*g}_{\infty 22}}$	quadratic	F-1
known aspect ratio, zero skew, p.p. at origin	$r^2\omega^*_{\infty 11} = \omega^*_{\infty 22}$	linear	F

Table 5.1: Auto-calibration constraints derived from the DIAC. (p.p. is the principal point)

Condition	Constraint	Type	# constraints
zero skew	$\omega_{\infty 12} = 0$	linear	F
p.p. at origin	$\omega_{\infty 13} = \omega_{\infty 23} = 0$	linear	2F
fixed aspect ratio, zero skew	$\frac{\omega^{f}_{\infty 11}}{\omega^{f}_{\infty 22}} = \frac{\omega^{g}_{\infty 11}}{\omega^{g}_{\infty 22}}$	quadratic	F-1
known aspect ratio, zero skew	$\omega_{\infty 11} = r^2\omega_{\infty 22}$	linear	F

Table 5.2: Auto-calibration constraints derived from the IAC. (p.p. is the principal point)

Table 5.1, and on ω_∞ in Table 5.2.

Constant camera constraint

If the cameras have identical internal parameters, i.e. $K^f = K^g$ for all $1 \leq f, g \leq F$, then $\omega^{*f}_\infty = \omega^{*g}_\infty$ which expands to $P^f Q^*_\infty {P^f}^T = P^g Q^*_\infty {P^g}^T$. However since these are homogeneous quantities, the equality only holds up to a non-zero scale. This still generates an extra five quadratic equations on the parameters of Q^*_∞,

$$\frac{\omega^{*f}_{\infty 11}}{\omega^{*g}_{\infty 11}} = \frac{\omega^{*f}_{\infty 12}}{\omega^{*g}_{\infty 12}} = \frac{\omega^{*f}_{\infty 13}}{\omega^{*g}_{\infty 13}} = \frac{\omega^{*f}_{\infty 23}}{\omega^{*g}_{\infty 23}} = \frac{\omega^{*f}_{\infty 33}}{\omega^{*g}_{\infty 33}}. \tag{5.11}$$

5.2.2 Required properties for Q^*_∞

In an arbitrary projective frame, the determined Q^*_∞ has to satisfy a number of properties in order to be valid. It must be singular, positive semi-definite and have $\mathbf{\Pi}_\infty$ as its null-vector.

Singularity

The absolute dual quadric is a degenerate quadric with a rank of 3. The singularity property can be enforced by placing the nonlinear constraint, $\det Q^*_\infty = 0$.

$\mathbf{\Pi}_\infty$ null-vector

From (2.15), $\mathbf{\Pi}_\infty$ is the null-vector of Q^*_∞. Given the parametrization $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T, 1\right]^T$ and

$$Q^*_\infty = \begin{bmatrix} \omega^{*1}_\infty & -\omega^{*1}_\infty \boldsymbol{\pi} \\ -\boldsymbol{\pi}^T \omega^{*1}_\infty & \boldsymbol{\pi}^T \omega^{*1}_\infty \boldsymbol{\pi} \end{bmatrix},$$

this property can be enforced by placing the constraint

$$Q^*_\infty \mathbf{\Pi}_\infty = \mathbf{0}.$$

Positive semi-definiteness

The absolute dual quadric is positive semi-definite (or negative semi-definite if the sign is reversed), i.e. Q^*_∞ is Hermitian ($Q^*_\infty = {Q^*_\infty}^H$) and all its eigenvalues are non-negative. This is obvious for the canonical form and is easily extended to an arbitrary frame. This condition is related to $\omega^*_\infty = PQ^*_\infty P^T$ that should be positive definite. If ω^*_∞ is not positive definite, it cannot be decomposed with Cholesky factorization to compute the camera calibration matrix. This condition can be enforced by placing the constraint,

$$\text{for all eigenvalues } \lambda_i,\ i = 1, 2, 3, \text{ of } Q^*_\infty,\ \lambda_i \geq 0.$$

5.2.3 Cheiral inequalities for Q^*_∞

According to (4.6) the depth of a 3D feature $\mathbf{X} = [X, Y, Z, W]^T$ in front of the image plane of camera $P = [B \mid \mathbf{p}_4]$, where it was projected as $\mathbf{x} = [x, y, w]^T$, is given by

$$\operatorname{depth}(\mathbf{X}; P) = \frac{\operatorname{sign}(\det B)\, w}{W \left\|\mathbf{b}_3^T\right\|}.$$

A feature $\mathbf{X}$ lies in front of a camera P if, and only if, $\operatorname{depth}(\mathbf{X}; P) > 0$. If we are only interested in if the feature is in front or behind the image plane, it is only necessary to consider

$$\operatorname{sign}\left(\operatorname{depth}\left(\mathbf{X};P\right)\right) = wW \det B. \tag{5.12}$$

The sign of depth $(\mathbf{X}; P)$ is referred to as the *cheirality* of feature $\mathbf{X}$, i.e. it indicates whether a feature is in front or behind a camera.

In the metric frame, all valid 3D features projected to the camera image are in front of the camera and therefore have a positive cheirality. Our goal to transfer the positive cheiral requirements for cameras $P^f H$ and 3D features $H^{-1}\mathbf{X}_i$, mapped by the the rectifying planar homography H, to constraints on the parameters of Q^*_∞. These constraints are called the cheiral inequalities.

We now derive an expression, in terms of H, for the sign $\left(\operatorname{depth}\left(H^{-1}\mathbf{X}; PH\right)\right)$ equation, where $\{P,\, \mathbf{X}\}$ is a projective reconstruction and H any projective planar homography.

sign $\left(\operatorname{depth}\left(H^{-1}\mathbf{X}; PH\right)\right)$ in terms of any homography H

We define $\mathbf{C} = [c_1,\, c_2,\, c_3,\, c_4]^T$ to be the vector where

$$c_i = (-1)^i \det \widehat{P}^{(i)} \tag{5.13}$$

and where $\widehat{P}^{(i)}$ is the matrix obtained by removing the i-th column from P, e.g. $\widehat{P}^{(4)} = B$. We also denote

$$P_{\mathbf{v}} = \begin{bmatrix} P \\ \mathbf{v}^T \end{bmatrix},$$

the 4×4 matrix made up of 3×4 matrix P augmented with a final row $\mathbf{v}^T$. The co-factor expansion of the determinant along the last row gives $\det P_{\mathbf{v}} = \mathbf{v}^T\mathbf{C}$, for $\mathbf{C}$ defined as in (5.13) and $\mathbf{v}^T$ any vector. If we were to choose $\mathbf{v} = \boldsymbol{\rho}_i$, the i-th row of P, then

$$\boldsymbol{\rho}_i^T\mathbf{C} = \det P_{\boldsymbol{\rho}_i} = 0.$$

Since this is true for all i, it follows $P\mathbf{C} = 0$, which means $\mathbf{C}$ is the null-vector of P and therefore its camera center.

We can now reformulate (5.12) as

$$\operatorname{sign}\left(\operatorname{depth}\left(\mathbf{X};P\right)\right) = w\left(\mathbf{e}_4^T\mathbf{X}\right)\left(\mathbf{e}_4^T\mathbf{C}\right), \tag{5.14}$$

where $\mathbf{e}_4 = [0,\, 0,\, 0,\, 1]^T$. Note that $\mathbf{e}_4$ is the vector representing the metric form of the plane at infinity and that $\mathbf{X}$ lies on the plane at infinity if, and only if, $\mathbf{e}_4^T\mathbf{X} = 0$.

Consider any projective planar transformation $\mathbf{H}$ so that $P'\mathbf{X}' = PHH^{-1}\mathbf{X}$.

What happens now to $\mathbf{C}$? We see that

$$\det\left((PH)_{H^T\mathbf{v}}\right) = \mathbf{v}^T H \mathbf{C}_{PH} = \det P_{\mathbf{v}} \det H = \mathbf{v}^T \mathbf{C} \det H,$$

where $\mathbf{v}^T$ is again any $4-$vector. Since the last expression is true for all $\mathbf{v}$, it follows that $H\mathbf{C}_{PH} = \mathbf{C}\det H$, or

$$\mathbf{C}_{PH} = H^{-1}\mathbf{C}\det H. \tag{5.15}$$

Now apply (5.15) to (5.14):

$$\begin{aligned}\operatorname{sign}\left(\operatorname{depth}\left(H^{-1}\mathbf{X}; PH\right)\right) &= w\left(\mathbf{e}_4^T H^{-1}\mathbf{X}\right)\left(\mathbf{e}_4^T \mathbf{C}_{PH}\right) \\ &= w\left(\mathbf{e}_4^T H^{-1}\mathbf{X}\right)\left(\mathbf{e}_4^T H^{-1}\mathbf{C}\right)\det H.\end{aligned}$$

One may interpret $\mathbf{e}_4$ as the plane at infinity $\mathbf{\Pi}_\infty$ in the projective frame mapped to infinity, $\mathbf{\Pi}_\infty = \mathbf{e}_4^T H^{-1}$ with $\mathbf{e}_4^T\mathbf{X}' = \Pi_\infty^T H H^{-1}\mathbf{X}$. Finally denoting $\delta = \operatorname{sign}(\det H)$, we obtain

$$\operatorname{sign}\left(\operatorname{depth}\left(H^{-1}\mathbf{X}; PH\right)\right) = w\left(\mathbf{\Pi}_\infty^T\mathbf{X}\right)\left(\mathbf{\Pi}_\infty^T\mathbf{C}\right)\delta.$$

Cheiral inequality constraints

We now know $\operatorname{sign}\left(\operatorname{depth}\left(H^{-1}\mathbf{X}_i; P^f H\right)\right) = w_i^f\left(\mathbf{\Pi}_\infty^T\mathbf{X}_i\right)\left(\mathbf{\Pi}_\infty^T\mathbf{C}^f\right)\delta$, where $\left\{P^f, \mathbf{X}_i\right\}$ is a projective reconstruction and H any projective planar homography. Now our goal is to find the constraints on our metric rectifying homography H that ensures that the depth $\left(H^{-1}\mathbf{X}_i; P^f H\right) > 0$ for all f and i.

We know that $w_i^f > 0$ for image features $\mathbf{x}_i^f = \left[x_i^f, y_i^f, w_i^f\right]^T$ and that we chose $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T, 1\right]^T$ with $\boldsymbol{\pi}$ the same as in (5.3) and (5.7). The sign of the depth equation then becomes

$$\operatorname{sign}\left(\operatorname{depth}\left(H\mathbf{X}_i, P^f H^{-1}\right)\right) = \left(\begin{bmatrix}\boldsymbol{\pi}\\1\end{bmatrix}^T\mathbf{X}_i\right)\left(\begin{bmatrix}\boldsymbol{\pi}\\1\end{bmatrix}^T\mathbf{C}^f\right)\delta > 0 \tag{5.16}$$

that must hold for all pairs (i, j).

Since we are free to multiply $\left[\boldsymbol{\pi}^T, 1\right]^T$ by -1 and have either $\delta = -1$ or $\delta = 1$, the following inequalities follow directly from (5.16):

$$\begin{aligned}\mathbf{X}_i^T\begin{bmatrix}\boldsymbol{\pi}\\1\end{bmatrix} &> 0 \quad \text{for all } i \\ \delta\,\mathbf{C}^{f\,T}\begin{bmatrix}\boldsymbol{\pi}\\1\end{bmatrix} &> 0 \quad \text{for all } f.\end{aligned} \tag{5.17}$$

These inequalities are called the *cheiral inequalities* and were first applied by Hartley to 3D reconstruction and camera calibration in [10] and [11]. A valid rectifying homography H then have to satisfy these inequalities for all features i and cameras f with either $\delta = 1$ or $\delta = -1$. These equations are useful since π is contained in the parametrization of Q^*_∞ and therefore these constraints can be applied without even calculating H. To enforce this condition, the cheiral inequalities can, and should, be added as an extra constraint to the constrained numerical optimization of Q^*_∞.

5.2.4 Counting arguments

The number of constraints required to determine a full metric reconstruction in our problem comes down to the number of constraints required to solve the absolute dual quadric. The absolute dual quadric has 8 essential parameters, taking into account that it is represented by a symmetric 4×4 matrix that is homogeneous and singular. This, as expected, coincides with the number of parameters for the rectifying homography, 5 for the absolute conic and 3 for the plane at infinity.

Consider F views and suppose k of the internal calibration parameters are known in all the views and l of the internal calibration parameters are fixed over all the views although unknown. Then $k + l \leq 5$, where 5 is the maximum number of calibration parameters in the camera calibration matrix K, where $KK^T = \omega^*_\infty = PQ^*_\infty P^T$. A fixed and known calibration parameter (correctly chosen according to Table 5.1 and 5.2) provides one constraint per view via the condition $\omega^{*f}_\infty = P_f Q^*_\infty P_f^T$, for a total of Fk constraints. A fixed but unknown parameter (again chose correctly according to Table 5.1 and 5.2) provides one fewer constraint, since just the value of the unknown parameter is missing. Thus l fixed parameters provide a total of $l\,(F-1)$ constraints. The number of views and constraints requirement for calibration is then

$$Fk + (F-1)\,l \geq 8. \tag{5.18}$$

Table 5.2.4 gives a few examples of constraint sets and the minimum number of views required. Note that it is possible to solve Q^*_∞ for constant cameras with unknown calibration using a minimum of 3 images. On the other hand, the variable camera can be solved using a minimum of 9 images with at least one fixed calibration parameter, or 8 images for one known parameter.

It is important to note that some known or fixed parameters only provide a constraint if another parameter is known, use Table 5.1 and 5.2 as a guide. Further note that using the minimum number of images is not recommended.

Constraints	Known	Fixed	min # images
zero skew	s		8
fixed aspect ratio, zero skew	s	r	5
known aspect ratio, zero skew	s, r		4
only focal length unknown	s, r, x_0, y_0		2
standard auto-calibration problem		s, r, x_0, y_0, α	3

Table 5.3: Examples of minimum views required for auto-calibration.

5.2.5 Methods for solving Q^*_∞

There are basically two approaches to solving Q^*_∞: the linear approach and the nonlinear approach. In our experience the nonlinear approach is, by far, more robust.

linear-approach

The linear approach is fairly straightforward. Since Q^*_∞ is a 4×4 symmetric matrix, it has 10 unknown elements which are written as a vector $\mathbf{x}$. The linear constraints on Q^*_∞ are assembled as rows in a matrix so that a matrix equation of the form $A\mathbf{x} = 0$ follows. The solution to $\mathbf{x}$ is then the null space of A, $\mathbf{x} \in \text{null}(A)$. This is then typically solved by obtaining a least squares solution via the SVD.

The singularity condition of Q^*_∞ is not implicitly enforced here but can be obtained postiori by setting the smallest (4th) singular value to zero. The main difficulty of this approach is the difficulty in enforcing that Q^*_∞ is positive semi-definite. It is not clear how to enforce this constraint.

nonlinear approach

The recommended method is the nonlinear optimization approach. For the known or fixed elements of $K^f\ {K^f}^T$ with all the projective cameras P^f known, we iteratively minimize the sum

$$e = \sum_{f=1}^{F} \left\| K^f\ {K^f}^T - P^f Q^*_\infty\ {P^f}^T \right\|^2 \tag{5.19}$$

to solve unknown absolute dual quadric Q^*_∞, where $\|M\|$ is the Frobenius norm of a matrix M, and $K^f\ {K^f}^T$ and $P^f Q^*_\infty\ {P^f}^T$ are normalized to have unit Frobenius norm. The motivation behind the unit Frobenius normalization is to eliminate the unknown scale factors. The sum (5.19) will now be carefully explained.

In the sum (5.19) not all matrix elements are necessarily included. The one extreme is if all the K^f are the same, i.e. all the cameras have the same calibration, then all matrix elements are included since

$$\begin{aligned} \omega_\infty^{*1} &= K^1\, {K^1}^T = K^f\, {K^f}^T \\ \text{and } Q_\infty^* &= \begin{bmatrix} \omega_\infty^{*1} & -\omega_\infty^{*1}\boldsymbol{\pi} \\ -\boldsymbol{\pi}^T\omega_\infty^{*1} & \boldsymbol{\pi}^T\omega_\infty^{*1}\boldsymbol{\pi} \end{bmatrix}. \end{aligned}$$

On the other hand we have the extreme where none of the calibration parameters for K^f and therefore $K^f\, {K^f}^T$ is known, or fixed. In this case none of the elements of $K^f\, {K^f}^T$ can be included in (5.19) and Q_∞^* is unsolvable. For the rest, suppose some calibration parameter is known. Tables 5.1 gave a summary of the known calibration parameters that lead to known entries in $K^f\, {K^f}^T$ and only those matrix entries will be included in (5.19).

Since the unknown elements of $K^f\, {K^f}^T$ are not included in the sum (5.19) and the fixed entries of K^f are included in the parameters of K^1, there are 8 unknown parameters of Q_∞^* to be estimated, 5 for K^1 and 3 for $\mathbf{\Pi}_\infty$:

- α - The focal length in frame 1 in terms of the image x-direction pixel dimension with an initialization value of 1.
- r - The 1st frame pixel aspect ratio with an initialization value of 1.
- s - The 1st frame skew with an initialization value of 0.
- $(x_0,\, y_0)$ - The camera principal point in frame 1 with an initialization value of (0, 0)
- $\boldsymbol{\pi}$ - The 3-vector from $\mathbf{\Pi}_\infty = \left[\boldsymbol{\pi}^T,\, 1\right]^T$ with an initialization value of $[0,\, 0,\, 0]^T$

Then to construct Q_∞^* for the sum (5.19):

$$K^1 = \begin{bmatrix} \alpha & s & x_0 \\ & r\alpha & y_0 \\ & & 1 \end{bmatrix},$$

$$\omega_\infty^{*1} = K^1\, {K^1}^T \quad \text{and} \quad Q_\infty^* = \begin{bmatrix} \omega_\infty^{*1} & -\omega_\infty^{*1}\boldsymbol{\pi} \\ -\boldsymbol{\pi}^T\omega_\infty^{*1} & \boldsymbol{\pi}^T\omega_\infty^{*1}\boldsymbol{\pi} \end{bmatrix},$$

according to the parametrization in (5.7). The given initialization corresponds to $\omega_\infty^{*1} = I$ and $Q_\infty^* = \widetilde{I}$, i.e. their canonical forms, see [37].

Note that if a known or fixed calibration parameter does not deliver a known entry in the DIAC $\omega_\infty^* = K^f\ {K^f}^T$, it could possibly deliver a known entry in the IAC $\omega_\infty = {K^f}^{-T}\ {K^f}^{-1}$. Consult Table 5.2 for a summary of the calibration parameters that deliver known entries in the IAC. The alternative sum of

$$e' = \sum_{f=1}^{F} \left\| {K^f}^{-T}\ {K^f}^{-1} - \left(P^f Q_\infty^*\ {P^f}^T\right)^{-1} \right\|^2$$

is then minimized (or a combination of both sums). An example is the often used assumption of zero skew, $s = 0$, that gives a zero entry in ${K^f}^{-T}\ {K^f}^{-1}$ as seen in 5.9. The zero skew assumption is especially useful when no calibration parameters are known.

The minimization problem is most effectively solved using a constrained optimization procedure. All known constraints on Q_∞^* should be added as optimization constraints. This also helps to limit critical motions that will be discussed later. These constraints include:

- known and fixed calibration parameters
- highly nonlinear singularity, $\det Q_\infty^* = 0$
- highly nonlinear positive semi-definiteness, $\lambda_i \geq 0$ for λ_i, $i = 1, 2, 3$, an eigenvalue of Q_∞^*
- null-vector constraint, $Q_\infty^* \begin{bmatrix} \boldsymbol{\pi} \\ 1 \end{bmatrix} = \mathbf{0}$
- the cheiral inequalities, $\mathbf{X}_i^T \begin{bmatrix} \boldsymbol{\pi} \\ 1 \end{bmatrix} > 0$ and $\delta\ {\mathbf{C}^f}^T \begin{bmatrix} \boldsymbol{\pi} \\ 1 \end{bmatrix} > 0$ for all i and f

All the conditions of Q_∞^* are implicitly met in this approach, since they are added as constraints to the constrained optimization of the cost function.

In this work the optimization is implemented with the COBYLA (Constrained Optimization BY Linear Approximations) algorithm [26]. The constraints are all then rewritten so that they are added as a sequence of functions that all must be $>= 0$ at each optimization iteration.

5.2.6 Camera calibration

Once Q_∞^* is determined, ω_f^* is calculated using (5.5) for all frames $f = 1, \ldots, F$. Since $K^f\ {K^f}^T = \omega_\infty^{*f}$, the upper triangular camera calibration matrix K^f for each frame f is determined through a simple Cholesky factorization. This completes the auto-calibration part of the algorithm.

Algorithm 5.1 Metric rectification using auto-calibration based on Q^*_∞.

Objective

Given a projective reconstruction $\{P^f, \mathbf{X}_i\}$ from a set of matched features $\mathbf{X}_i$ across a set of views with constraints on the camera calibration matrices K^f, compute a metric reconstruction of the features and cameras.

Algorithm

```
1: Use $\omega^{*f}_\infty = P^f Q^*_\infty {P^f}^T$ along with constraints on the elements of $\omega^{*f}_\infty$ from
   $K^f$, constraints derived from properties of $Q^*_\infty$ and constraints supplied by
   the required properties of $Q^*_\infty$ to estimate $Q^*_\infty$.
2: Decompose $Q^*_\infty$ as $H\widetilde{I}H^{-1}$ with assistance from eigenvalue decomposition,
   where $\widetilde{I}$ is the matrix $\mathrm{diag}(1, 1, 1, 0)$.
3: Apply $H^{-1}$ to the features and $H$ to the projection matrices to acquire a
   metric reconstruction.
```

5.3 Extracting the rectifying homography

Given an already determined Q^*_∞ in a specific projective frame, we now wish to determine the rectifying homography H. It was shown in (5.6) that the absolute dual quadric has the decomposition

$$Q^*_\infty = H^T \widetilde{I} H.$$

Extracting H is therefore the simple matter of decomposing Q^*_∞.

This decomposition is easily achieved by either parametrising Q^*_∞ according to (5.7) and constructing H according to (5.3), or with the eigenvalue decomposition of $Q^*_\infty = E\Lambda E^{-1}$. Then

$$H = E\, \Lambda'^{\frac{1}{2}}$$

with Λ' the matrix Λ with its 0 eigenvalue replaced by 1. Note that H is the rectifying homography for cameras, $P_m = PH$, and H^{-1} the rectifying homography for features, $\mathbf{X}_M = H^{-1}\mathbf{X}$.

5.4 Algorithm outline

The steps in the auto-calibration algorithm based on the absolute dual quadric is summarized in Algorithm 5.1. We now have a method, summarized in Algorithm 5.2, to reconstruct from features in perspective images the 3D structure and motion of a single rigid object up to a similarity.

Algorithm 5.2 Metric reconstruction of a single rigid object.

Objective

Given a set of N features over F perspective images $\left\{\mathbf{x}_i^f\right\}$ belonging to a rigid object, recover the 3D structure and motion of the object up to a similarity.

Algorithm

```
1: Determine a projective reconstruction $\{P^f, \mathbf{X}_i\}$ for the image features using
   the projective matrix reconstruction algorithm ( Algorithm 4.1).
2: Rectify the projective reconstruction to a metric reconstruction with a trans-
   formation obtained through auto-calibration based on the absolute dual
   quadric $Q^*_\infty$ (Algorithm 5.1).
```

5.5 Critical motion sequences

It is important to note that a metric reconstruction from a video sequence is not necessarily guaranteed. A sequence of camera motions that does not allow the complete determining of the rectifying homography H is termed a critical motion sequence (CMS). This is the case when a motion sequence does not lead to a unique solution to the auto-calibration problem. The resulting metric 3D reconstruction is then degenerate and falls somewhere between projective and metric.

Critical motion sequences have been systematically classified by Sturm [29] in the case of constant internal parameters. This classification has been extended to more general calibration constraints, i.e. varying focal length, in [30, 22]. A geometric analysis by Pollefeys and van Gool, [23], is also worth mentioning.

Metric reconstruction depends on identifying the absolute conic Ω_∞ which is fixed under similarity transformations. An auto-calibration solution method based on the absolute dual quadric was proposed, (from (5.5))

$$\omega^{*f}_\infty = P^f Q^*_\infty \, {P^f}^T = K^f \, {K^f}^T \,, \qquad (5.20)$$

for image frame $f = 1, \ldots, F$, where K^f is the camera calibration matrix for frame f. Problems arise when the absolute conic is not the only full-rank virtual conic to satisfy the constraints on the camera calibration parameters in (5.20), when projecting in each image. If that is the case then there is no way to distinguish the absolute conic.

It is clear from (5.20) that the possibility of a motion sequence being critical depends on the number of constraints that are imposed for auto-calibration. The extremes are all parameters known, in which case almost no critical motions exist; and no constraints at all in which case all motion sequences are critical.

Known intrinsic camera parameters	Type of IAC	Type of cone
α, r, x_0, y_0, s	centered circle, known radius	absolute cone
r, x_0, y_0, s	centered circle	circular cone
r, s	circle	elliptic cone
x_0, y_0, s	centered ellipse, axis aligned with image	elliptic cone
s	ellipse, axis aligned with image	elliptic cone
-	ellipse	elliptic cone

Table 5.4: Possible types of IAC and re-projection cones for a set of known intrinsic camera parameters.

When there is an ambiguity in the intrinsic camera parameters an equivalent ambiguity exists in ω_∞ and therefore other conics ϕ exist which are potential ω_∞. Further, as seen in Section 3.2.4.1, for every potential IAC ϕ there exists a reprojection cone – and then possibly a potential absolute conic Φ, formed from all of the reprojection cones intersecting a plane in space with the same conic.

We know the absolute conic Ω_∞ projects as a circle, centered at the point closest to the center of projection, on the image plane when it is viewed embedded in metric space (before it is transformed by camera calibration to pixel coordinates). This projection is a similarity transformation from the plane at infinity to the image plane. Then in each image, the IAC $\omega_\infty = K^{-T}K^T$ is represented by an ellipse centered at the principal point, the aspect ratio between its axes similar to that of the pixel coordinates and the skew the deviation of its axes from the image axes. Note that both Ω_∞ and ω_∞ are virtual conics and are not visible. If some intrinsic parameters are known, it allows us to transform the IAC to constrain its shape, and in some cases the shape of the reprojection cone. The different possibilities are summarized in Table 5.4. The first column contains the camera calibration parameters which are assumed known, the second and third column gives the corresponding type of IAC and virtual re-projection cone.

The goal is to only provide a overview of critical motion sequences and most results will be stated without proof. For a more thorough analysis the reader should consult the sources previously mentioned. To provide this overview we follow Sturm's approach, and look for all motion sequences that allows a specific potential absolute conic (PAC) Φ to solve (5.5) for all frames. The PAC Φ is defined as:

- a full-rank virtual conic that is not Ω_∞
- that projects an virtual ellipse (conic) ϕ^f, $f = 1, \dots F$, in the image plane of camera f.

Note that the conic ϕ^f is a potential IAC. This task is typically split in two parts, looking for PAC on the plane at infinity and looking for PAC outside the plane at infinity.

5.5.1 Potential absolute conic on the plane at infinity

The problem in this case is relatively simple. Consider Φ is on the plane at infinity. According to (3.6), geometric entities on the plane at infinity $\mathbf{\Pi}_\infty$ project according to $H = KR$, thus depending only on the camera orientation and not position. For example points $\mathbf{X}_\infty = \left[\mathbf{d}^T, 0\right]^T$ on $\mathbf{\Pi}_\infty$ project according to $\mathbf{x} = KR\mathbf{d}$, independent of position.

Let the image plane be viewed embedded in metric space, instead of in pixel coordinates, so that

$$\begin{gathered}
\mathbf{x}_i'^f = K^{f\,-1}\mathbf{x}_i^f \\
\omega'_\infty = K^{f\,T}\omega_\infty K^f = K^{f\,T}\left(K^{f\,-T}K^{f\,-1}\right)K^f = I = \omega'^*_\infty \\
\phi'^f = K^{f\,T}\phi K^f.
\end{gathered}$$

The homography between the plane at infinity and the image plane of camera f is now given by $H^f = R^f$ and the projection of the potential absolute conic Φ to a conic in each 3D image plane is given , according to (2.8) (and the fact that conics are symmetric), by

$$\phi'^f = R^{f\,-T}\,\Phi\,R^{f\,-1} = R^f\Phi\,R^{f\,T}. \tag{5.21}$$

We now determine all rotations R^f for which Φ is not the absolute conic.

Let $\mathbf{x}$ be an eigenvector of Φ with eigenvalue λ. From (5.21) we can write $R^f\Phi = \phi'^f R^f$ and it follows,

$$R^f\lambda\mathbf{x} = R^f\Phi\mathbf{x} = \phi'^f R^f\mathbf{x} = \lambda R^f\mathbf{x},$$

that $\mathbf{x}$ is also an eigenvector of ϕ'^f with eigenvalue λ. This is valid for all eigenvectors $\mathbf{x}$ of Φ. Thus, it follows that R^f conserves the eigenspaces and ϕ'^f and Φ have the same eigenvalues for all f.

There are three possible eigenvalue combinations: a triple eigenvalue, one double eigenvalue and one single eigenvalue,

or three single eigenvalues.

- *one triple eigenvalue*: Only the absolute conic itself corresponds to a triple eigenvalue, so it is ignored.
- *one double eigenvalue and one single eigenvalue*: ϕ^f and Φ are circles. The eigenspace corresponds to a plane $\mathbf{\Pi}$ and a line $\boldsymbol{l}$ perpendicular[1] to plane $\mathbf{\Pi}$. Since the rotation R conserves Π and $\boldsymbol{l}$, it must be either the identity transformation I, a rotation about $\boldsymbol{l}$ by an arbitrary angle, or a rotation by 180^o about a line on Π, incident with $\boldsymbol{l}$.
- *three single eigenvalues*: ϕ^f and Φ are ellipses. The eigenspace consists of three mutually orthogonal lines. Besides the identity transformation, only rotations of 180^o around any of these lines leave Φ unchanged.

Depending on the known intrinsic parameters and the constraints they place on a potential IAC (Table 5.4), the set of critical motions is determined from one of the three possible eigenvalue combinations.

5.5.2 Potential absolute conic outside the plane at infinity

In this case the problem is more complicated. Consider Φ not on the plane at infinity. Contrary to Φ on the plane at infinity, the projection of conics now depend on both camera orientation and position. The problem can however be separated in two parts. First all camera positions are determined. For Φ a PAC, all possible camera positions are vertices of cones that contain Φ. Then the possible camera orientations that leave each cone unchanged are determined.

As seen in Table 5.4, for some intrinsic constraints there are constraints on the possible cones. For the case where all intrinsic parameters are known, the cone is an absolute cone (i.e. has a triple eigenvalue). And the case where all intrinsic parameters are known but the focal length, the cone is a circular cone (i.e. double and single eigenvalue). For the rest the cone is an elliptic cone (i.e. three single eigenvalues).

The possible orientations of the cameras now depend on the shape of the virtual cones. From [29]:

- Any rotation around the vertex of an absolute cone leave the cone globally unchanged.
- Arbitrary rotations around the main axis (the line from the vertex of a cone to the center of its base) or rotations by 180^o about axes perpendicular but incident with the main axis leaves a circular cone unchanged.
- Rotations by 180^o about an axis of the elliptic cone leaves it unchanged.

[1]Eigenvectors of symmetric matrices corresponding to different eigenvalues are orthogonal.

Description	$\#\mathbf{t}$	$\#R$	Ambiguity	Ω_∞ dof
Planar motion	∞^2	∞	scaling axis perpendicular to plane	1
Pure translation	∞^3	1	affine transformation	5
Orbital motion	∞	1	projective distortion along rotation axis	2
Pure rotation	1	∞^3	arbitrary position for plane at infinity	3

Table 5.5: Practically important critical motion sequences for constant intrinsic parameters.

Description	$\#\mathbf{t}$	$\#R$	Ambiguity	Ω_∞ dof
Translation and rotation about optical axis	∞^3	$2 \times \infty$	scaling optical axis	1
Hyperbolic and/or elliptic motion	$2 \times \infty$	$2 \times \infty$	one extra solution	-
Forward motion	∞	$2 \times \infty$	projective distortion along optical axis	2
Pure rotation	1	∞^3	arbitrary position for plane at infinity	3

Table 5.6: Practically important critical motion sequences for varying focal length.

5.5.3 Practically important critical motion sequences

We are now interested in specific descriptions of critical motion sequences that may have practical significance. It is useful since these motions can then be avoided when acquiring an image sequence for auto-calibration and reconstruction. In Table 5.5.3 we provide details on CMS cases where all intrinsic parameters are constant and in Table 5.5.3 for CMS cases where all intrinsic parameters are known except for focal length. The number of possible positions and orientations that the critical motion can consist of is presented, along with the ambiguity inherent in the motion and the number of degrees of freedom on the absolute conic Ω_∞. Note that ∞^i indicates an infinite number in i dimensions and $i \times \infty$ an infinite number for i cases.

5.5.4 Reducing ambiguities

It is possible to reduce the degree of ambiguity in a scene if prior knowledge of the scene structure or camera is available. A known or partially known camera calibration should always be used if available. Another way of discarding ambiguous solutions is to analyze reconstructions with respect to physical contradictions, e.g. to see whether a reconstructed point lies behind a camera by which it is actually seen. However, the best way to counter critical motion sequences, is to avoid them by using motion sequences that are clearly “far” from critical. Take the orbital motion sequence which is adequate for modeling but critical if the camera is not calibrated; including rotations about the optical axis turns this sequence into non-critical.

Chapter 6

Motion Segmentation

So far we have assumed a scene contains a single moving object. We now focus on dynamic scenes consisting of multiple moving objects. Figure 6.1 illustrates a scene consisting of two objects, each with their associated object coordinate systems. Motion segmentation is the segmentation of individual objects, with unique motions, from a background. It is essential to the understanding and reconstruction of dynamic scenes. The terms “motion segmentation” and “motions” are preferred to “object segmentation” and “objects” since as long as two separate objects move as a single rigid object they obey our definition of a sin-

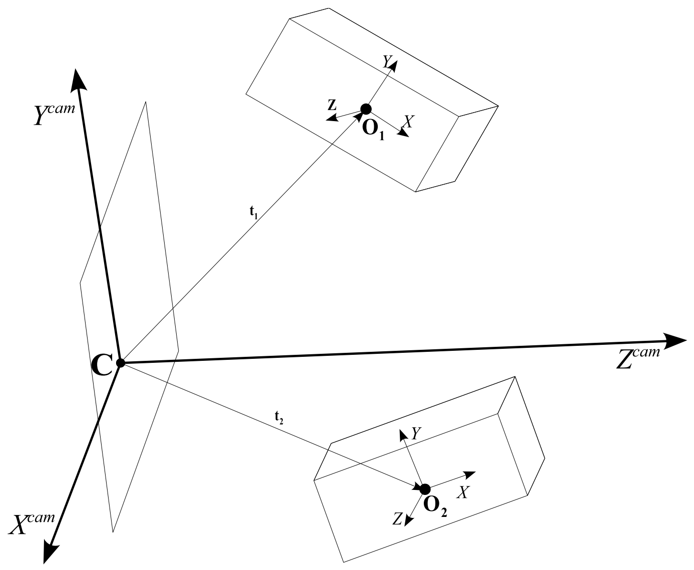


Figure 6.1: The coordinate system of two bodies relative to a camera coordinate system.

gle rigid object. Thus, "object" segmentation is based on separate, independent motion.

Ever since Tomasi and Kanade in [36] introduced the matrix factorization method based on the idea that the trajectories of general motion under an affine projection span a 4-dimensional linear manifold, this geometric constraint has become the basis for motion segmentation. The first algorithm was presented by Costeira and Kanade in [4] for the segmentation of image features captured by motion tracking.

6.1 Problem statement

Suppose we have a scene with two motions, each assigned their own defined object coordinate systems (OCS) centered at their respective feature centroid. Each motion has N_i, $i = 1, 2$, features so that a total of $N = N_1 + N_2$ are tracked over F perspective camera images. These features are scaled with their correct set of projective depths, calculated according to the matrix factorization algorithm of Chapter 4, for projective reconstruction and collected in the $3F \times N$ measurement W from (4.7).

Suppose that we somehow know the classification of these features and could permute the columns of matrix W so that the first N_1 columns belong to motion 1 and the last N_2 belong to motion 2. Matrix W then has the canonical form

$$W^* = \left[\begin{array}{c|c} W_1 & W_2 \end{array} \right]. \tag{6.1}$$

Each of the submatrices W_k, $k = 1, 2$, is the measurement matrix for motion k and is reduced SVD decomposed to form a rank-4 perspective measurement matrix

$$W_k \quad = \quad U_k \Sigma_k V_k^T \ .$$

Each of the 2 motions, according to Chapter 4 and 5, has a projective reconstruction and rectifying homography H_k to form a metric reconstruction,

$$W_k = M_k S_k = \widehat{M}_k H_k H_k^{-1} \widehat{S}_k,$$

for $k = 1, \ldots, 2$. Equation (6.1) then has the canonical factorization

$$\begin{aligned} W^* &= [M_1 \,|\, M_2] \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix} \\ \text{with} \quad [M_1 \,|\, M_2] &= [U_1 \,|\, U_2] \begin{bmatrix} \Sigma_1^{\frac{1}{2}} & \\ & \Sigma_2^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} H_1 & \\ & H_2 \end{bmatrix} \\ \text{and} \quad \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix} &= \begin{bmatrix} H_1^{-1} & \\ & H_2^{-1} \end{bmatrix} \begin{bmatrix} \Sigma_1^{-\frac{1}{2}} & \\ & \Sigma_2^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} V_1^T & \\ & V_2^T \end{bmatrix}. \end{aligned} \tag{6.2}$$

By denoting

$$\begin{aligned} M^* = [M_1 \,|\, M_2]\,,\ S^* &= \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix},\ H^* = \begin{bmatrix} H_1 & \\ & H_2 \end{bmatrix}, \\ U^* = [U_1 \,|\, U_2]\,,\ \Sigma^* &= \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \ \text{and} \ {V^*}^T = \begin{bmatrix} V_1^T & \\ & V_2^T \end{bmatrix} \end{aligned}$$

we can express the matrix factorization and metric reconstruction for multiple motions as

$$\begin{aligned} W^* &= M^* S^* \\ M^* &= U^* \Sigma^{*\frac{1}{2}} H^* \\ S^* &= H^{*-1} \Sigma^{*-\frac{1}{2}} V^{*T}, \end{aligned} \tag{6.3}$$

which is similarly to the single object case.

From (6.2) and Section 4.3.4 it is clear that W_1 and W_2 each have a rank of 4 and therefore that W^* has a theoretical rank of 8, assuming of course the motion vectors span 3D. Consequently a noisy W has a rank of at least 8.

In reality we do not know which features belong to which motion and so W, with its columns a mixture of motion 1 and 2, can not be placed in its canonical form W^*. We can still however apply the reduced SVD to W to obtain $W = U\Sigma V^T$. It appears that all that has to be done now to calculate the metric reconstruction is to find the rectifying homography H^*. There is, however, a fundamental flaw in this reasoning.

A rectifying homography H is computed from the motion of a single object, that is, by assuming that the measurement matrix consists of features from a single object. This is evident in the projective factorization of Section 4.3.4 and especially equation (4.10) where a rank greater than 4 (maximum rank for a single 3-space rigid motion) would result in projection matrices that are not of the required shape of 3×4 and reconstructed features which are not in 3-space. Thus the required conditions are only available after motion segmentation which involves obtaining W^*. Another problem is that we do not necessarily know how many motions there are and therefore how many classifications of features are required to avoid mixed motions and erroneous results.

Consequently, the motion segmentation process reduces to two objectives:

- Determine how many motions there are in the scene and therefore the rank constraint that applies.
- Discover which features belong to which motion so that the correct columns of W can be grouped to allow for the metric reconstruction for each motion object.

6.2 Motion Subspaces

We return to our N features tracked over F frames. Suppose all the features have already been scaled with the correct set of projective depths so that $\lambda_i^f \mathbf{x}_i^f = \left[\lambda_i^f x_i^f,\, \lambda_i^f y_i^f,\, \lambda_i^f\right]^T$ is the i-th feature in the f-th frame. Stacking all of ith feature's scaled coordinates over the F frames vertically (the ith column of measurement matrix W), we have single vector of the form

$$\mathbf{w}_i = \left[\lambda_i^1 x_i^1,\, \lambda_i^1 y_i^1,\, \lambda_i^1,\, \ldots,\, \lambda_i^F x_i^F,\, \lambda_i^F y_i^F,\, \lambda_i^F\right]^T \tag{6.4}$$

that represents the image motion, or trajectory, of the i-th feature in a $3F$-dimensional space (or over F frames in 3D space).

Taking a step back, consider the perspective projection of the ith 3D feature in frame f. This projection is given by

$$\lambda_i^f \mathbf{x}_i^f = P^f \mathbf{X}_i = \left[B^f \mid -B^f \widetilde{\mathbf{C}}^f\right] \mathbf{X}_i = X_i \mathbf{a}^f + Y_i \mathbf{b}^f + Z_i \mathbf{c}^f + \mathbf{d}^f$$

where

$$\mathbf{X}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix},\; B^f = \left[\mathbf{a}^f,\, \mathbf{b}^f,\, \mathbf{c}^f\right],\; \text{ and } \mathbf{d}^f = -B^f \widetilde{\mathbf{C}}^f.$$

If we define $3F$-dimensional vectors $\mathbf{m}_0 = \left[\mathbf{a}^{1\,T},\, \cdots\; \mathbf{a}^{F\,T}\right]^T$, $\mathbf{m}_1 = \left[\mathbf{b}^{1\,T},\, \cdots\; \mathbf{b}^{F\,T}\right]^T$, $\mathbf{m}_2 = \left[\mathbf{c}^{1\,T},\, \cdots\; \mathbf{c}^{F\,T}\right]^T$ and $\mathbf{m}_3 = \left[\mathbf{d}^{1\,T},\, \cdots\; \mathbf{d}^{F\,T}\right]^T$, it follows from (6.4) that

$$\mathbf{w}_i = X_i \mathbf{m}_0 + Y_i \mathbf{m}_1 + Z_i \mathbf{m}_2 + \mathbf{m}_3,$$

$i = 1, \ldots, N$. This result shows that the set of N trajectories $\{\mathbf{w}_i\}$, $i = 1, \ldots, N$, belongs to a 4-dimensional subspace spanned by the vectors $\{\mathbf{m}_0,\, \mathbf{m}_1,\, \mathbf{m}_2,\, \mathbf{m}_3\}$. This is true for perspective cameras with the image features scaled with the correct projective depths.

It follows that each independent motion occupies a unique 4-dimensional subspace spanned by vectors $\{\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3\}$. Therefore, the process of segmenting N features into O separate motions is equivalent to grouping the N features into O distinct 4-dimensional subspaces.

6.3 Number of Motions

It has been reported that estimating the number of motions is more difficult than segmentation itself [4, 7]. At first glance, one can determine the number of independent motions from the rank of the measurement matrix. In practice however, measurement noise greatly increases the rank estimated from real world measurements, throwing off motion estimates. Many studies have been done for estimating the number of motions automatically, but as stated in [32], none of them is really satisfactory. This is because the number of motions is not well-defined; one moving object can also be viewed as multiple objects moving similarly, and there is no rational way to unify similarly moving objects into one except using heuristic thresholds or ad-hoc criteria.

Gear [7] attempted a complicated statistical analysis for estimating the number of motions and concluded that individual points were likely to be judged as independently moving. This makes sense because saying that each point is moving independently (but coincidentally in unison) is always a more likely interpretation than saying that the motions of different points are constrained, as long as the judgment is based on statistical likelihood alone.

It is assumed in this work that the number of motions are determined from manual observation, and provided as an input. The required rank is then given by

$$r = 4O, \tag{6.5}$$

where O is the number of independent motions.

6.4 Subspace separation

Consider the general noisy measurement matrix W. In terms of the trajectory vectors discussed, it can be expressed as

$$W = [\mathbf{w}_1, \mathbf{w}_2 \cdots \mathbf{w}_N],$$

where the N trajectories $\{\mathbf{w}_i\}$ belong to a r-dimensional subspace $\mathcal{L} \subset \mathbb{R}^{3F}$. The problem is that shape and motion interact in W. It is indicated in (6.2) that the example measurement matrix's rank is generated by two subspaces, $\mathcal{L}_1$

and $\mathcal{L}_2$ of rank r_1 and r_2 respectively. In the example the measurement matrix is rank-8. In general it would be rank-$4O$ given by (6.5) for O subspaces, and the subspaces rank-4 each. These two subspaces are represented by a block diagonal S^*. However, the recovered shape space V^T, obtained from the SVD of non-canonical W, is a linear combination of the two subspaces and has lost the block diagonal structure.

Matrix V^T can be interpreted as consisting of N column vectors $\mathbf{v}_i$,

$$V^T = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_N \end{bmatrix}.$$

Each vector $\mathbf{v}_i$ is unit length since V^T is orthonormal and has r components (is a r-vector) from the reduced SVD, where r is the rank of W. Vector $\mathbf{v}_i$ is a new representation for trajectory $\mathbf{w}_i$, which was linearly transformed from its redundant $\mathbb{R}^{3F}$ space to $\mathbb{R}^r$ space while preserving the subspace property.

There is however a construct that preserves the original subspace structure, *shape interaction matrix G*. Matrix G is a $N \times N$ matrix defined

$$G = VV^T. \tag{6.6}$$

Each of the matrix entries are an "interaction" between rank reduced trajectories,

$$G_{ij} = \mathbf{v}_i^T \mathbf{v}_j. \tag{6.7}$$

The entry G_{ij} in terms of column vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ from measurement matrix W is then given by

$$G_{ij} = \mathbf{w}_i^T U \Sigma^{-2} U^T \mathbf{w}_j. \tag{6.8}$$

Equation (6.8) is obtained when inserting

$$\mathbf{v}_i = \Sigma^{-1} U^T \mathbf{w}_i$$

in equation (6.7) with

$$\mathbf{w}_i = U \Sigma \mathbf{v}_i$$

from the SVD of W.

Matrix G is uniquely computable from W without segmentation, since V is uniquely obtained from the SVD of W. Another useful property of G is that permuting the columns of W does not change the set of values $\{G_{ij}\}$, only their arrangement. Swopping columns k and l of W result in swopping columns k and l of V^T which now swop both the rows and columns k and l for symmetric G, but not its entry values. The swopping clearly also goes the other way and swopping rows and columns k and l in G will result in a swopping of

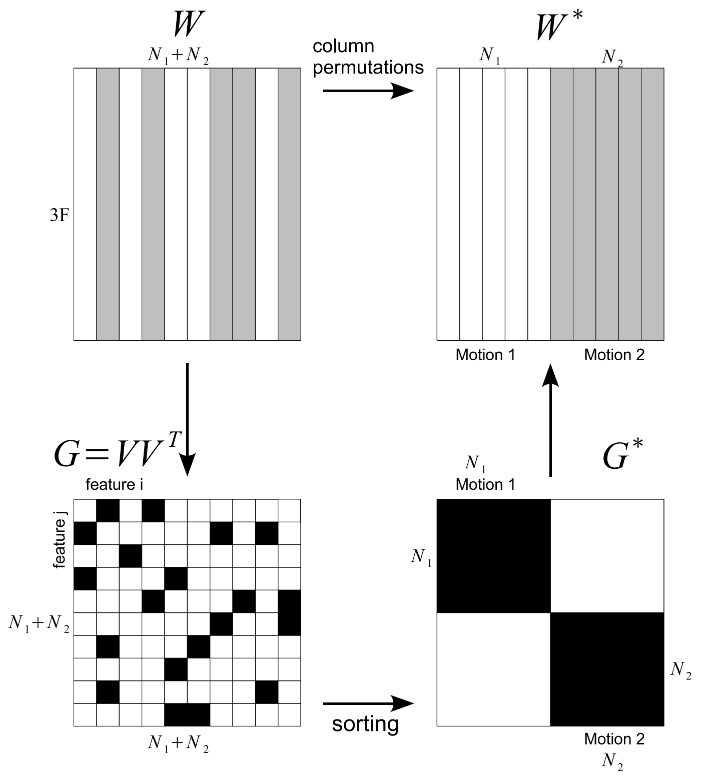


Figure 6.2: Segmentation process

columns k and l in W. Therefore, if we can successfully swop the entries of G so that it is in block-diagonal form with each block representing a motion, the corresponding swopping of columns in W would place it in canonical form W^*. This relationship between sorting G and permuting the columns of W is illustrated in Figure 6.4.

According to the subspace separation theorem stated in [16] we can use G to separate subspaces, since the (ij)-th element of G is zero if the i-th and j-th feature belong to different subspaces. The subspace separation theorem is rigorously proved by Kanatani in [16]. Here we adopt the more intuitive argument in reference to the Tomasi-Kanade matrix factorization of Costeira and Kanade [4]. They prove the case for orthographic cameras but the argument is applicable to perspective cameras without requiring any alteration.

Since the set of values $\{G_{ij}\}$ does not change, let us compute block-diagonal

G^* from W^*. Substituting (6.3) into (6.6) we obtain,

$$\begin{aligned}
G^* &= V^* V^{*T} \\
&= S^{*T} H^{*T} \Sigma^* H^* S^* \\
&= S^{*T} \left(H^{*-1} \Sigma^{*-1} H^{*-T} \right)^{-1} S^* \\
&= S^{*T} \left[\left(H^{*-1} \Sigma^{*-\frac{1}{2}} V^{*T} \right) \left(V^* \Sigma^{*-\frac{1}{2}} H^{*-T} \right) \right]^{-1} S^* = S^{*T} \left(S^* S^{*T} \right)^{-1} S^* \\
&= \begin{bmatrix} S_1^T & \\ & S_2^T \end{bmatrix} \begin{bmatrix} \Lambda_1^{-1} & \\ & \Lambda_2^{-1} \end{bmatrix} \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix} \\
&= \begin{bmatrix} S_1^T \Lambda_1^{-1} S_1 & \\ & S_2^T \Lambda_2^{-1} S_2 \end{bmatrix}
\end{aligned}$$

where $\Lambda^* = S^* S^{*T}$ and $\Lambda_i = S_i S_i^T$. Each entry of G^* then has the value

$$G_{ij}^* = \begin{cases} S_1^T \Lambda_1^{-1} S_1 & \text{if feature trajectory } i \text{ and } j \text{ both belong to object 1} \\ S_2^T \Lambda_2^{-1} S_2 & \text{if feature trajectory } i \text{ and } j \text{ both belong to object 2} \\ 0 & \text{if feature trajectory } i \text{ and } j \text{ both belong to different objects.} \end{cases} \tag{6.9}$$

These values $\{G_{ij}^*\}$, which are equal to $\{G_{ij}\}$, have another crucial property in that they are invariant to the motion specifics. This is the case since only the structure matrix S and not the motion matrix M is included in equation (6.6). In other words, irrespective of the actual motion of the objects, for different motion subspaces, these objects will produce the same entries in G.

To summarize:

- Knowing only the number of motions and not actual segmentation of the motions, we can compute a matrix G whose elements G_{ij} can be interpreted as the interaction between feature i and j. If the value of G_{ij} is non-zero, features i and j belong to the same motion.

- Subspace separation is reduced to swopping pairs of rows and columns of G until it is in block diagonal form G^*. Once this is achieved, the corresponding permutations of the columns applied to W will place it in canonical form W^* with the subspaces separated. All the features are now assigned to their respective motion and reconstruction can commence.

6.5 Separation procedure

In the presence of noise all the elements of G, in general, are non-zero irrespective of subspaces. This ensures that the subspace separation procedure is not a trivial endeavour.

Two possible subspace separation procedures are discussed shortly, an elementary greedy algorithm and a method based on spectral clustering for better reliability and speed.

6.5.1 Greedy algorithm

The greedy algorithm is a straightforward separation method where trajectories $\mathbf{w}_i$ and $\mathbf{w}_j$ are grouped for a large $|G_{ij}|$. It is initially assumed that each trajectory belongs to its own subspace $\mathcal{L}_i$, $i = 1, \dots N$. The similarity measure $s_{\alpha\beta}$ between subspace $\mathcal{L}_\alpha$ and subspace $\mathcal{L}_\beta$ is defined as

$$s_{\alpha\beta} = \max_{\mathbf{w}_i \in \mathcal{L}_\alpha, \mathbf{w}_j \in \mathcal{L}_\beta} |G_{ij}| \,.$$

Subspaces are repeatedly merged for large $s_{\alpha\beta}$ until, typically, the number of subspaces corresponds to the observed number of motions. Progressively interchanging the corresponding rows and columns of G will eventually place it in an approximately block diagonal form. A variation of this method was adopted in [4].

6.5.2 Spectral clustering

Spectral clustering, according to [2], refers to a class of techniques that rely on the eigenstructure of a similarity matrix (shape interaction matrix G in our case) to partition features into disjoint clusters – where features in the same cluster have a high similarity and features in different clusters have a low similarity. This technique was first applied to motion segmentation by Yan and Pollefeys in [40]. Given that there are O observed motions and therefore O motion subspaces, the goal is to segment the features into O clusters.

For motion segmentation based on spectral clustering, Yan and Pollefeys recommend recursive 2-way clustering, aka Shi-Malik algorithm, detailed in [27]. The recursive 2-way clustering algorithm, applied to motion segmentation, will partition a set of features into two separate sets, according to the eigenvector that corresponds to the second-smallest eigenvalue of the Laplacian matrix

$$L = I - D^{-\frac{1}{2}} G D^{-\frac{1}{2}}$$

of G, where D is the diagonal matrix

$$D_{ii} = \sum_{j=1}^{N} G_{ij}.$$

A Laplacian matrix is a matrix representation of a graph, so that the motion

Algorithm 6.1 Recursive 2-way clustering algorithm applied to subspace separation.

Objective

Separate shape interaction matrix G into O independent motion subspaces by placing it in block diagonal form with the recursive 2-way clustering algorithm.

Algorithm

```
1: From the similarity matrix $G$, calculate the Laplacian matrix $L$.
2: Initially segment the features into two clusters $C_1$ and $C_2$.
3: while the number of clusters $M = |\{C_1, C_2, \ldots, C_M\}|$ is less than $O$ do
4:   for $i = 1, \ldots, M$ do
5:     compute the shape interaction matrix $G_i$ from only the features assigned to cluster $C_i$.
6:     Use $G_i$ to calculate Laplacian matrix $L_i$ and partition cluster $C_i$ further into two clusters, $C_i^1$ and $C_i^2$.
7:     Calculate a Cheeger constant for each of the partitionings $C_i^1$ and $C_i^2$, see [20].
8:   end for
9:   The cluster $C_j$ with the highest valued Cheeger constant for its partitioning into $C_j^1$ and $C_j^2$, is removed and replaced by $C_j^1$ and $C_j^2$ in the set of clusters. All other partitions $C_i^1$ and $C_i^2$, $i = 1, \ldots, M$ with $i \neq j$, are ignored.
10: end while
11: Once all $O$ clusters are found, $G$ can be placed in block diagonal form where each block is a cluster.
```

segmentation problem is interpreted as a graph cutting (or graph partitioning) problem for matrix L. A numerical measure for the quality of a graph cut is the Cheeger constant [20]. If the two graph partitions, after a cut was made, have few links between them then the Cheeger constant is large. Otherwise, if the two partitions have many links between them, the Cheeger constant is small (but positive).

An outline for the recursive 2-way clustering algorithm applied to subspace separation is given in Algorithm 6.1.

6.6 Algorithm outline

The motion segmentation algorithm is summarized in Algorithm 6.2. The complexity of this algorithm is dominated by the method employed to block-diagonalize G.

Once motion segmentation is completed it is possible that there are outlier trajectories assigned to a subspace. The matrix factorization algorithm assumes a maximum rank of 4 and auto-calibration assumes a single object, therefore

Algorithm 6.2 Motion Segmentation algorithm

Objective

Given a set of N features over F perspective images $\left\{\mathbf{x}_i^f\right\}$ belonging to O independent rigid motions, calculate which features belong to which motion.

Algorithm

```
1: Determine the scale factors $\lambda_i^f$ using equation (4.9) and build the rescaled measurement matrix $W$.
2: Determine the theoretical rank $r$ of $W$ from the number of observed motions.
3: Decompose $W = U\Sigma V^T$ using SVD truncated to the $r$ biggest singular values.
4: Compute shape interaction matrix $G$ using $V^T$.
5: Block-diagonalize $G$, so that each block represents a motion.
```

both processes are negatively effected with an outlier. The recommended course of action is to liberally remove possible outliers after segmentation. The outlier removal method implemented in this work is rudimentary: at each segmentation of features into two clusters (step 9 in Algorithm 6.2), all features are removed which have an affinity that deviates from the cluster's mean affinity more than a certain threshold than the cluster's affinity standard deviation.

By extending Algorithm 5.2 with Algorithm 6.2, we have reached the objective of this thesis – to develop a full SfM implementation that reconstructs, up to a similarity transformation, the structure and motion of multiple independently moving rigid bodies, assuming a perspective camera(s) which is not required to be calibrated. This SfM system is summarized in Algorithm 6.3.

Algorithm 6.3 Metric reconstruction of multiple rigid objects.

Objective

Given a set of N features over F perspective images $\left\{\mathbf{x}_i^f\right\}$ belonging to O independent rigid motions, recover the 3D structure and motion up to a similarity for each object associated with a motion.

Algorithm

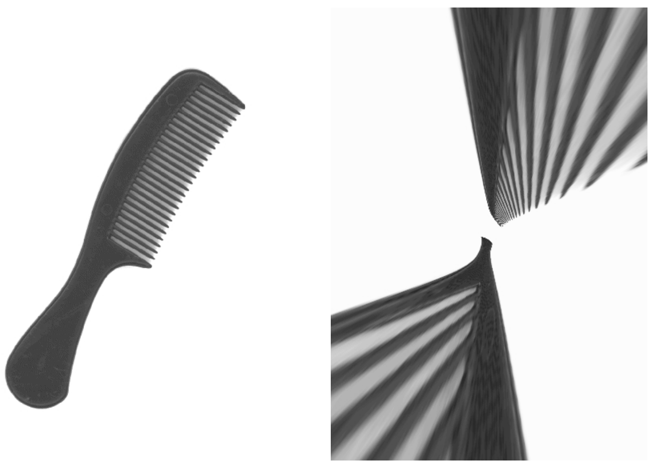
```
1: Segment the features into $O$ groups, where each group's features all undergo
   the same independent rigid motion (Algorithm 6.2).
2: for $j = 1, \ldots, O$ do
3:     Determine a projective reconstruction $\{P^f, \mathbf{X}_i\}$ for the image features of
       motion $j$ using the projective matrix reconstruction algorithm (Algorithm
       4.1).
4:     Rectify the projective reconstruction of motion $j$ to a metric reconstruc-
       tion with a transformation obtained through auto-calibration based on
       the absolute dual quadric $Q_\infty^*$ (Algorithm 5.1).
5: end for
```

Chapter 7

Implementation issue: quasi-affine reconstruction

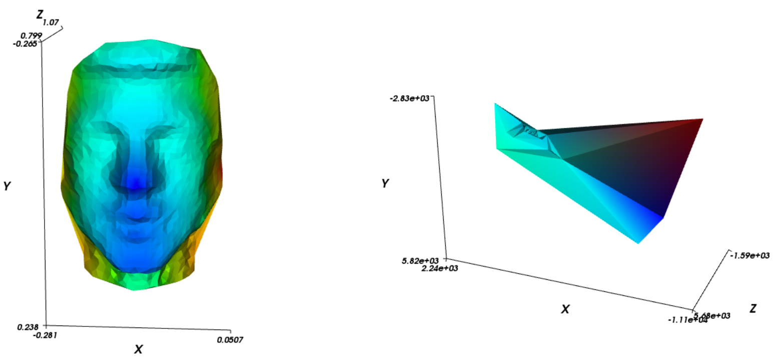
A 3D projective reconstruction is very general and allows for severe deformations of an object, to the extent that the object might not be recognizable. In fact, it can happen that the plane at infinity Π_∞ passes right through the object as illustrated in Figure 7.1 (taken from [12]) and Figure 7.2. The comb example from Figure 7.1 deals with a 2D feature set, but the principle is the same.

This can happen during the first projective reconstruction of the object and if this happens, the nonlinear optimization algorithm used to estimate the absolute dual quadric Q^*_∞, presented in Section 5.2.5, often fails to converge to the correct solution. Accordingly, it is necessary to move the plane at infinity Π_∞



(i) Comb (ii) Non-quasi-affine re-sampling of comb

Figure 7.1: Picture of a comb and a non-quasi-affine re-sampling of the comb.



(i) Face (ii) Non-quasi-affine resampling of face

Figure 7.2: Picture of a face and a non-quasi-affine re-sampling of the face.

through a projective transformation H (this causes no additional problems since the reconstruction is anyway only accurate up to a projective transformation), to infinity

$$H^T \mathbf{\Pi}_\infty = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

where $\mathbf{e_4} = [0,\, 0,\, 0,\, 1]$ is the canonical form of the plane at infinity. This ensures that it does not pass through the structure. To achieve this, H has the form

$$H = \begin{bmatrix} I & \mathbf{0} \\ -\frac{1}{\pi_4}\left[\pi_1,\, \pi_2,\, \pi_3\right] & \frac{1}{\pi_4} \end{bmatrix} \text{ where } \Pi_\infty = \left[\pi_1,\, \pi_2,\, \pi_3,\, \pi_4\right]^T \text{ and } \pi_4 \neq 0. \tag{7.1}$$

But Π_∞ is not known, and not trivial to find, as discussed in Chapter 6. Fortunately there is a relatively simple way of ensuring that Π_∞ does not pass unwittingly through the structure. We proceed to first define the problem and then to solve it.

7.1 Problem statement

A subset B of $\mathbb{R}^n$ is called *convex* if any line segment joining two points in B lies entirely in B. The *convex hull* of B is the smallest convex set containing B. We are concerned with 3D point sets, so $n = 3$. We view $\mathbb{R}^3$ as a subset of $\mathbb{P}^3$ consisting of all finite points. The infinite points constitute the plane at infinity Π_∞, i.e. $\mathbb{P}^3 = \mathbb{R}^3 \cup \Pi_\infty$. Thus a subset of $\mathbb{P}^3$ is called convex if and only if it is contained in $\mathbb{R}^3$ and convex in $\mathbb{R}^3$. This implies that a convex set does not contain any points at infinity. Now let B be a subset of

$\mathbb{R}^3$ and H a projective transformation of $\mathbb{P}^3$. The transformation H is said to be *quasi-affine* with respect to B if it preserves the convex hull of B, otherwise it is called non-quasi-affine. Points $\mathbf{X} = [X, Y, Z, W]^T \in \mathbb{P}^3$ with $W \neq 0$ and $\widetilde{\mathbf{X}} = [X/W, Y/W, Z/W]^T \in \mathbb{R}^3$ form a convex set, therefore a quasi-affine transformation ensures that the object remains on one side of the plane at infinity. Thus, by definition, a quasi-affine reconstruction (a reconstruction that differs from the full metric reconstruction by an quasi-affine transformation) lies at one side of the plane at infinity.

7.2 Transformation to quasi-affine

Suppose we have an Euclidean reconstruction with cameras P_E^j, $j = 1, \ldots, F$, observing the points $\mathbf{X}_{i\,E}$, $i = 1, \ldots, N$, in front of the cameras. Let $\mathbf{x}_i^j$ be the corresponding image features, i.e. $\mathbf{x}_i^j = P_E^j\, \mathbf{X}_{E\,i}$. Now suppose that we have a projective reconstruction $\left\{P^j, \mathbf{X}_i\right\}$ from the features $\mathbf{x}_i^j$, such that $P^j\mathbf{X}_i = \lambda_i^j\mathbf{x}_i$, then it can be shown (see [13]) that if $\mathbf{X}_i$ is finite for all i and the depths λ_i^j have the same sign for all j and i, then a quasi-affine transformation H exists taking each $\mathbf{X}_i$ to $\mathbf{X}_{E\,i}$. This implies that the reconstruction lies at one side of the plane at infinity. Therefore we are looking for a projective transformation so that all the reconstructed points are finite, and all the depths have the same sign. The first step is to adjust the given projective reconstruction, both the projective cameras P^j as well as the projective feature $\mathbf{X}_i$, by multiplication with -1 if necessary, so that λ_i^j is positive whenever $\mathbf{x}_i^j$ exists.

The next step is to find the transformation H that will transform the projective reconstruction to a quasi-affine reconstruction for which all the points lie in front of the camera. Writing the plane that is mapped by H to infinity, as $\mathbf{v}$, the condition that the points lie in front of the camera is given by

$$\operatorname{depth}(\mathbf{X}_i; P^j) \doteq (\mathbf{v}^T\mathbf{X}_i)(\mathbf{v}^T C^j)\delta > 0$$

where C^j is the camera center of camera P^j and $\delta = \operatorname{sign}(\det H)$. Since we are free to multiply the homogeneous quantity $\mathbf{v}$ by -1 if necessary, we may assume that $(\mathbf{v}^T C^1)\delta > 0$ for the first camera P^1. We therefore arrive at the cheiral inequalities,

$$\begin{aligned} \mathbf{v}^T\mathbf{X}_i &> 0, \text{ for all } i \\ \delta\mathbf{v}^T C^j &> 0, \text{ for all } j. \end{aligned} \tag{7.2}$$

Since the $\mathbf{X}_i$ and C^j are known we can solve this set of inequalities for $\mathbf{v}$. The only quantity that is unknown is δ. Since it can only be ± 1 we need to try both signs, if necessary. The cheiral inequalities are solved using linear programming.

More detail is given in a moment. Assume for now that $\mathbf{v}$ has been solved, the question then is to construct a suitable transformation H, given $\mathbf{v}$. Since H maps $\mathbf{v}$ to infinity, it follows that

$$H^{-T}\mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

or

$$\mathbf{v} = H^T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Thus H is any transformation such that its last row is $\mathbf{v}$ and $\det H \doteq \delta$. If the last component of $\mathbf{v}$ is non-zero then one can choose the first three rows of H to be of the form $\pm \begin{bmatrix} I & | & \mathbf{0} \end{bmatrix}$.

7.2.1 Solving the cheiral inequalities

The cheiral inequalities may be solved using linear programming (LP) techniques. As it stands, if vector $\mathbf{v}$ is a solution then so is $a\mathbf{v}$ for any positive factor a. We can restrict the solution domain further by normalizing each point $\mathbf{X}_i$ to size 1 for all i and each camera center $\mathbf{C}^j$ to size 1 for all j and then adding the further inequalities

$$-1 < v_i < 1 \text{ for } \mathbf{v} = [v_1, v_2, v_3, v_4]^T.$$

It is further possible to have one less variable to estimate by translating, using an affine transformation, the centroid of the 3D features $\mathbf{X}_i$ to the origin. Since the plane at infinity cannot cross the convex hull, it cannot cross the origin. This says that the last component v_4 of $\mathbf{v}$ is non-zero and that we can then write

$$\mathbf{v} = \begin{bmatrix} \widetilde{\mathbf{v}} \\ 1 \end{bmatrix}. \tag{7.3}$$

The quasi-affine transformation H then has the new form,

$$H = \begin{bmatrix} I & \mathbf{0} \\ \widetilde{\mathbf{v}}^T & 1 \end{bmatrix}.$$

To achieve a unique solution we also need to specify an objective function to

Algorithm 7.1 Computing a quasi-affine reconstruction.

Objective

Given a set of 3D features $\mathbf{X}_i$ and cameras matrices P^j constituting a projective reconstruction from a set of image features, i.e. $P^j\mathbf{X}_i = \lambda_i^j \mathbf{x}_i^j$, compute a projective transformation H transforming the projective to a quasi-affine reconstruction.

Algorithm

```
1: Replace some camera matrices $P^j$ with $-P^j$ and some features $\mathbf{X}_i$ with
   $-\mathbf{X}_i$ as required to ensure that each $\lambda_i^j > 0$.
2: Move the feature's centroid to the origin with affine transformation $H$ by
   replacing $\mathbf{X}_i$ with $H^{-1}\mathbf{X}_i$ and $P^j$ with $P^j H$.
3: Normalize each feature $\mathbf{X}_i$ to size 1 then calculate the camera centers $\mathbf{C}^j$
   according to (5.13) followed by a normalization to size 1.
4: Form the cheiral inequalities (7.4).
5: For each of the values $\delta = \pm 1$, calculate a solution to the cheiral inequalities
   (if it exists) . Let the solution be $\widetilde{\mathbf{v}}_\delta$.
6: The matrix $H_\delta$ is the required transformation and has the form
   $$H_\delta = \begin{bmatrix} I & \mathbf{0} \\ \widetilde{\mathbf{v}}_\delta^T & 1 \end{bmatrix}.$$
   If both $H_+$ and $H_-$ exist, then they lead to two oppositely orientated quasi-
   affine reconstructions.
```

be maximized. An appropriate strategy is to maximize the extent to which each of the cheiral inequalities is solved. To do this we introduce further variables d_1 and d_2. The inequalities of (7.2) combined with (7.3) then become

$$\begin{aligned} \mathbf{X}_i^T \begin{bmatrix} \widetilde{\mathbf{v}} \\ 1 \end{bmatrix} &> d_1 \quad \text{for all } i \\ \delta \mathbf{C}^{jT} \begin{bmatrix} \widetilde{\mathbf{v}} \\ 1 \end{bmatrix} &> d_2 \quad \text{for all } j. \end{aligned} \tag{7.4}$$

We seek to maximize d_1 and d_2 while satisfying all of the inequalities. We now have a standard LP problem that can be readily solved. A popular choice is the the simplex method. If a solution exists for which $d_1 > 0$ and $d_2 > 0$ then it is the desired solution.

7.3 Algorithm outline

The quasi-affine reconstruction algorithm is summarized in Algorithm 7.1 and should be applied after projective reconstruction and before metric rectification.

Algorithm 7.2 Revised metric reconstruction of multiple rigid objects.

Objective

Given a set of N features over F perspective images $\left\{\mathbf{x}_i^f\right\}$ belonging to O independent rigid motions, recover the 3D structure and motion up to a similarity for each object associated with a motion.

Algorithm

```
1: Segment the features into $O$ groups, where each group's features all undergo
   the same independent rigid motion (Algorithm 6.2).
2: for $j = 1, \ldots, O$ do
3:     Determine a projective reconstruction $\left\{P^f, \mathbf{X}_i\right\}$ for the image features of
       motion $j$ using the projective matrix reconstruction algorithm (Algorithm
       4.1).
4:     Prevent the plane at infinity $\Pi_\infty$ from passing through the projective
       reconstructed structure, and as a result algorithms possibly failing to
       converge, by transforming the projective reconstruction to a quasi-affine
       reconstruction (Algorithm 7.1).
5:     Rectify the projective reconstruction of motion $j$ to a metric reconstruc-
       tion with a transformation obtained through auto-calibration based on
       the absolute dual quadric $Q_\infty^*$ (Algorithm 5.1).
6: end for
```

The solving of this implementation issue leads to a revising of our SfM system, summarized in Algorithm 6.3, by extending it with Algorithm 7.1. The final version of the SfM system, which is the objective of this thesis, is now summarized in Algorithm 7.2.

Chapter 8

Experimental investigation

We now put the algorithms discussed in this study to the test. This chapter consists of two parts. The first part focuses on analyzing our full metric reconstruction system and the experiments are:

- Cube sequences
  - Pure synthetic sequence
  - Noisy synthetic sequence
  - Quasi-real sequence
- Quasi-real face sequence
- Real sequences
  - Lego sequence
  - Medusa sequence

The second part of the chapter focuses on analyzing our motion segmentation system and the experiments are:

- Quasi-real cube sequences
  - Background removal
  - Multiple motion segmentation
- Real sequence
  - Lego sequence

8.1 Cube sequences

This section covers three experiments that deal with the metric reconstruction of cubes. The first experiment uses a pure synthetic cube motion sequence created in Python. The second experiment is performed on the same sequence, but it is corrupted with controlled amounts of measurement noise. The third experiment makes use of a software rendered cube and a feature tracker to select and track cube features.

The cube, its motion and the camera calibration parameters are similar in all three of the experiments. The cube has an edge length of 2 and in the first frame its centroid is at a distance of 6 from the camera center. The cube translates and rotates at a constant speed in each of the x, y and z directions for a total of 50 frames. The projection of the cube onto its images is performed according to the perspective camera model with each image of size 640×480. The calibration parameters of the perspective camera is given by,

$$s = 0,\ r = 1.0,\ x_0 = 0.0,\ y_0 = 0.0 \text{ and } \alpha = 1.0$$

so that

$$K_{ref}^{f} = \begin{bmatrix} \alpha & s & x_0 \\ 0 & r\alpha & y_0 \\ 0 & 0 & 1 \end{bmatrix} = I,$$

fixed for all frames $f = 1, \ldots, F$, is the camera calibration matrix.

A reconstruction is done for each even-numbered frame starting from the 4th frame. Note that the matrix factorization and auto-calibration algorithms are batch-processing procedures, therefore a reconstruction at a specified frame number is a full reconstruction from the first frame until the specified frame.

To include in the experiments a comparison of the influence of the number of known calibration parameters on reconstruction accuracy, we do a series of reconstructions for each of the eight different calibration assumptions. The different assumptions are given in Table 8.1.

Since the ground truth is known for all of the experiments in this section, we can determine and compare the accuracy of the experimental results. To do this we first have to define the measures of accuracy for the experimental results. The accuracy criteria are first given then followed by the first cube experiment.

8.1.1 Accuracy criteria

We want to determine the accuracy in reconstructing structure, motion and unknown camera calibration parameters. For that we need an appropriate error measure. Since the reconstruction is only up to a similarity, for comparison

Calibration assumption	Known calibration parameters	Fixed calibration parameters
a		s, r, x_0, y_0, α
b	s	r, x_0, y_0, α
c	s, r	x_0, y_0, α
d	s, r, x_0, y_0	α
e	s	
f	s, r	
g	s, r, x_0, y_0	
h	s, r, x_0, y_0, α	

Table 8.1: Camera calibration assumptions.

purposes, we need to map reconstructions to ground truth using a similarity transformation.

Structure error

We define the L2-norm of the distance between the i-th estimated 3D feature $\mathbf{X}_i$ and its ground truth reference $\mathbf{X}_i^{ref}$ as

$$e_{s\,i} = \left\| \mathbf{X}_i^{ref} - H\mathbf{X}_i \right\|,$$

where

$$H = \begin{bmatrix} \sigma R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

is a similarity transformation with translation vector $\mathbf{t}$, rotation matrix R and scaling σ. The Euclidean 3D features are normalised so that the last element of their homogeneous vector representation is 1. The similarity transformation H attempts a transformation of the estimated features, reconstructed up to a similarity transformation in the metric strata, to their Euclidean equivalent with optimal orientation, position and scale so as to give a consistent Euclidean error estimate.

To estimate the similarity transformation H, the centroid of the features is first translated to the origin so that $\mathbf{t} = \mathbf{0}$. Now only four parameters have to be estimated:

- θ_x – x-axis rotation angle with rotation matrix R_x
- θ_y – y-axis rotation angle with rotation matrix R_y
- θ_z – z-axis rotation angle with rotation matrix R_z
- σ – scaling,

so that

$$H = \begin{bmatrix} \sigma R_x R_y R_z & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

The previously mentioned COBYLA non-linear constrained optimization algorithm is now used to minimize the error $e = \sum_{i=1}^{N} \left\| \mathbf{X}_i^{ref} - H\mathbf{X}_i \right\|$ to estimate these parameters. The parameters are initialised with

$$\theta_x = 0,\, \theta_y = 0,\, \theta_z = 0,\, \sigma = 1$$

and have constraints

$$0 \le \theta_x,\, \theta_y,\, \theta_z \le 2\pi.$$

We use the well-known Root-Mean-Square Error (RMSE) to calculate the reconstruction error

$$e_s = \sqrt{\frac{1}{N} \sum_{i=1}^{N} {e_s}_i^2},$$

where N is the number of features. The unbiased sample variance of the structure is defined as

$$\sigma_s^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left({e_s}_i - \bar{e}_{s\,i} \right)^2,$$

where the mean is defined as

$$\bar{e}_{s\,i} = \frac{1}{N} \sum_{i=1}^{N} {e_s}_i\,.$$

Motion error

The motion error consists of both a rotation error and a translation error. With the completion of a metric reconstruction, for each frame $f = 1 \ldots F$ we have an estimated projection matrix $P_f = K^f \left[R^f \,|\, \mathbf{t}^f \right]$, where the motion for each frame f relative to the first, $f = 1$, is represented by a rotation matrix R^f and translation vector $\mathbf{t}^f$. We first define the rotation error.

Any 3D rotation can be described by a (normalized) axis of rotation $\mathbf{a}$ and an angle of rotation ϕ. To keep the rotation accuracy simple, the error in the angle of rotation is ignored and the rotation error defined as the error in the rotation axis. For frame f the positive angle between the estimated rotation axis $\mathbf{a}^f$ and the ground truth reference rotation axis $\mathbf{a}_{ref}^f$ is given by

$$\theta^f = \arccos\left(\left| {\mathbf{a}^f}^T \mathbf{a}_{ref}^f \right| \right).$$

The difference between the estimated and reference rotation axis for frame

f is then defined as

$$e_r^f = \frac{\theta^f}{\pi/2},$$

so that $0 \leq e_r^f \leq 1$. The rotation error is then

$$e_r = \sqrt{\frac{1}{F-1}\sum_{f=2}^{F} {e_r^f}^2},$$

where F is the total number of frames. The unbiased sample variance of rotation is defined as

$$\sigma_r^2 = \frac{1}{F-2}\sum_{f=2}^{F}\left(e_r^f - \bar{e}_r^f\right)^2,$$

where the mean is defined as

$$\bar{e}_r^f = \frac{1}{F-1}\sum_{f=2}^{F} e_r^f.$$

Note that the first frame does not contain motion and is therefore not included in the error calculation.

We now define the translation error. We define the L2-norm of the difference between estimated and reference translation as

$$e_t^f = \left\| \mathbf{t}_{ref}^f - \mathbf{t}^f \right\|,$$

where the estimated and reference translation are both normalized to size 1. We again use the RMSE to now calculate the translation error,

$$e_t = \sqrt{\frac{1}{F-1}\sum_{f=2}^{F} {e_t^f}^2},$$

where F is the total number of frames. The unbiased sample variance of translation is defined as

$$\sigma_t^2 = \frac{1}{F-2}\sum_{f=2}^{F}\left(e_t^f - \bar{e}_t^f\right)^2,$$

where the mean is defined as

$$\bar{e}_t^f = \frac{1}{F-1}\sum_{f=2}^{F} e_t^f.$$

Again the first frame is not included in the calculation since there is no motion.

Calibration error

We define the Frobenius norm of the difference between the reference and estimated calibration parameters as

$$e_c^f = \left\| K_{ref}^f - K^f \right\|,$$

where K^f is the camera calibration matrix for frame f. The calibration matrices are normalised so that matrix element $K_{33}^f = 1$ and $K_{ref\ 33}^f = 1$. The calibration error is then

$$e_c = \sqrt{\frac{1}{F} \sum_{f=1}^{F} {e_c^f}^2},$$

where F is the number of frames. The unbiased sample variance of the calibration is then defined as

$$\sigma_c^2 = \frac{1}{F-1} \sum_{f=1}^{F} \left(e_c^f - \overline{e}_c^f \right)^2,$$

where the mean is defined as

$$\overline{e}_c^f = \frac{1}{F} \sum_{f=1}^{F} e_c^f.$$

Note that for a camera with fixed calibration, e_c^f is the same for each frame.

8.1.2 Pure synthetic cube

This Python generated sequence serves as a sanity check, to check the validity of our reconstruction algorithms and the capability of their implementation under ideal circumstances. Since the sequence is synthetic, no feature tracking is performed and therefore there are no measurement noise or potential for erroneous tracking. The cube is also assumed to be be transparent. The 2D features of the cube projections that we attempt to reconstruct, include each of the eight cube corners and the centers of each of the twelve cube edges, for a total of 20 features.

The structure error values for all the assumptions over the 50 frames are given in Figure 8.1, rotation errors in Figure 8.2, translation errors in Figure 8.3 and calibration errors in 8.4. The dark bar in the histograms, Figure 8.5, is the average error over the 50 frames, and the light bar is the error value associated with the frame that returned the most accurate structure.

Note that there is a clear correlation when comparing the error results for the structure, rotation, translation and calibration. An assumption performing

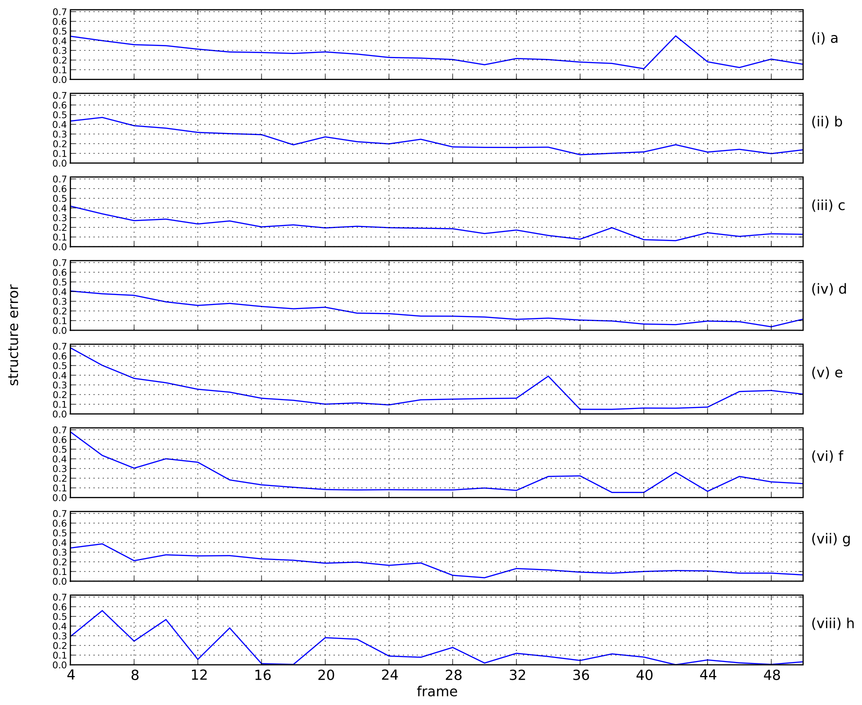


Figure 8.1: Structure accuracy of the pure synthetic sequence.

well in one of the criteria is also performing well in the rest. This is especially clear for assumptions **e** through **h**.

Figure 8.1 illustrates the reconstruction error over 50 frames for all the calibration assumptions, and Figure 8.5 (i) the average and best structure error. We observe for all of the assumptions that the structure accuracy improves with a frame increase up to about frame 30, ignoring a spike here and there, where they have all converged to an error value less than $e_s = 0.2$. From there the structure accuracy stays more or less constant. Inspecting Figure 8.5 (i), assumption **h**, with all calibration parameters known, clearly gives the best average and single structure error, as expected. Its best accuracy with an error value of $e_s = 0.0008$, and variance of $\sigma_s^2 = 8 \times 10^{-8}$, for frame 42 is near perfect. Assumptions **g** and **d**, with only the focal length unknown, both return a very good average and best error. Assumption **a** with no parameters known, but all fixed, returns the weakest average- and best error. The average structure error as well as the best structure error visibly improves with an increase in the number of known calibration parameters. Again this is no surprise.

Assumption **e**, with only the skew known, performs surprisingly well, better

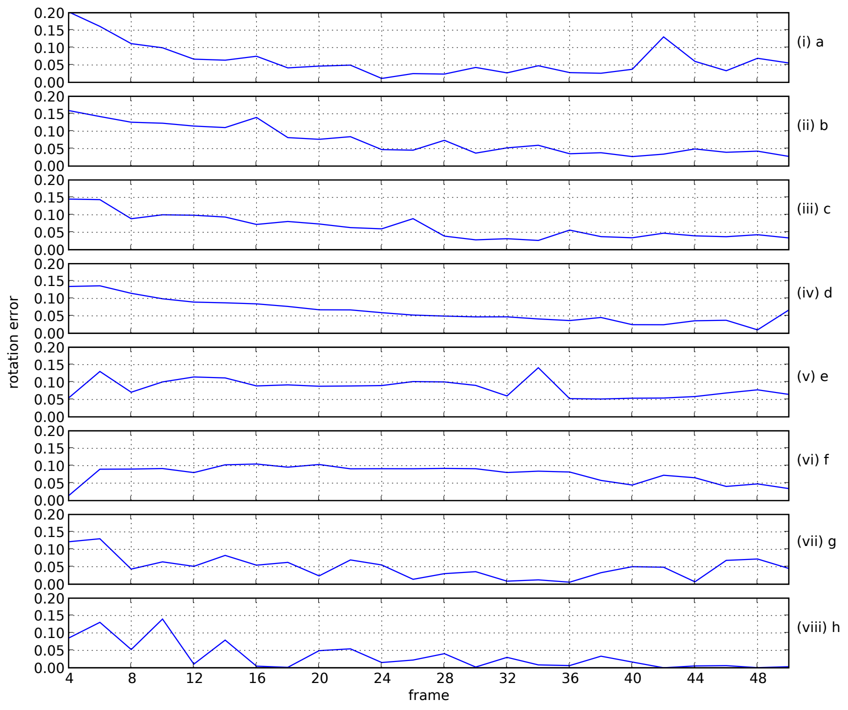


Figure 8.2: Rotation accuracy of the pure synthetic sequence.

than calibration assumptions **a** and **b**, even though **a** and **b** both have more known parameters. This performance could be attributed to the adaptability of **e**, with many unknowns that aren't fixed, and maybe to a bit of luck. It should also be noted that assumption **e** is theoretically only able to give a solution after 8 frames, according to equation (5.18).

The rotation, translation and calibration error results are similar to the structure results discussed above, except for a slightly better rotation performance for assumption **a** in Figure 8.5 (iii) and a dramatically more erroneous translation performance for assumption **a** in Figure 8.5 (iv). It seems that the translation accuracy is more sensitive to the number of known calibration parameters than the rest. Note that the constant non-zero calibration error for assumption **h** in Figure 8.4 is *not* a mistake. We would expect an error of zero since all the calibration parameters are known. The reason it is not zero is that before and during reconstruction, scaling of the image features typically occur. This can be understood in the following way.

A 2D feature scaling is the result of a left multiplication of a measurement

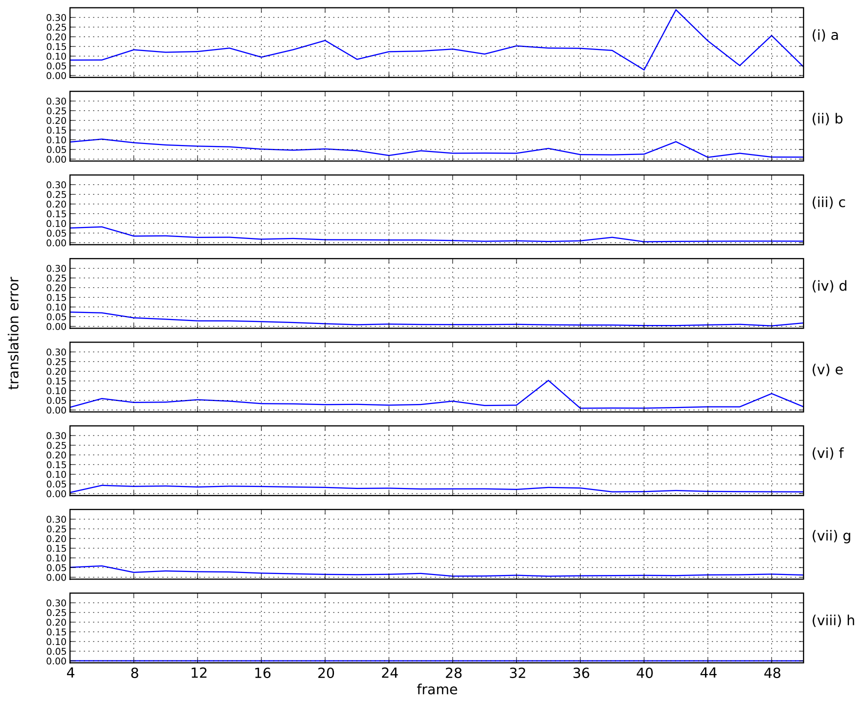


Figure 8.3: Translation accuracy of the pure synthetic sequence

$\mathbf{x}_i^f$ with a matrix H, where

$$H = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The effect it has on the calibration matrix K^f is given by

$$\begin{aligned} H\mathbf{x}_i^f &= HP^f\mathbf{X}_i \\ &= H\left(K^f\left[R^f \mid \mathbf{t}^f\right]\right)\mathbf{X}_i \\ &= K'^f\left[R^f \mid \mathbf{t}^f\right]\mathbf{X}_i \end{aligned}$$

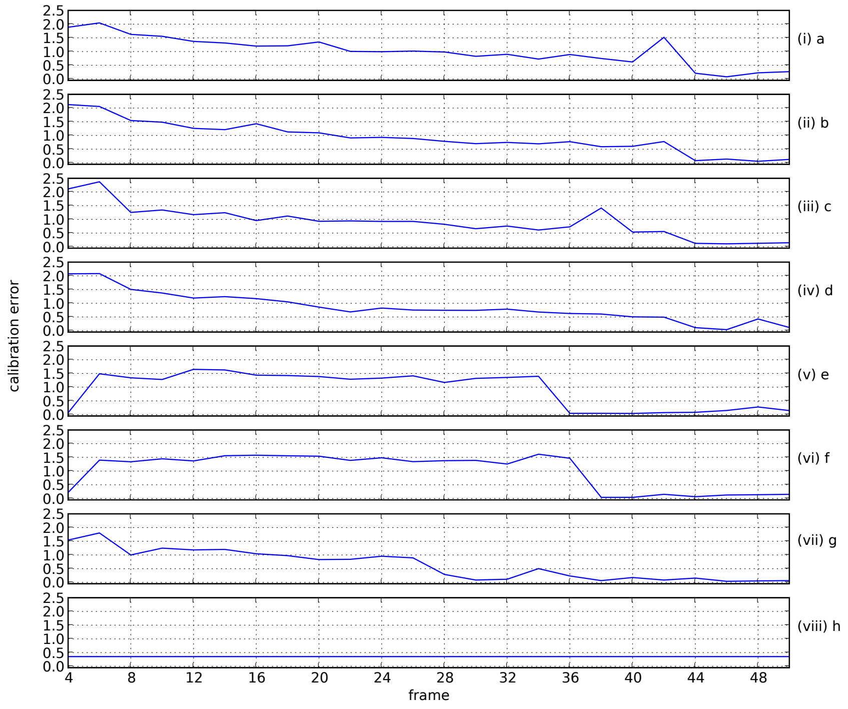


Figure 8.4: Calibration parameter accuracy of the pure synthetic sequence.

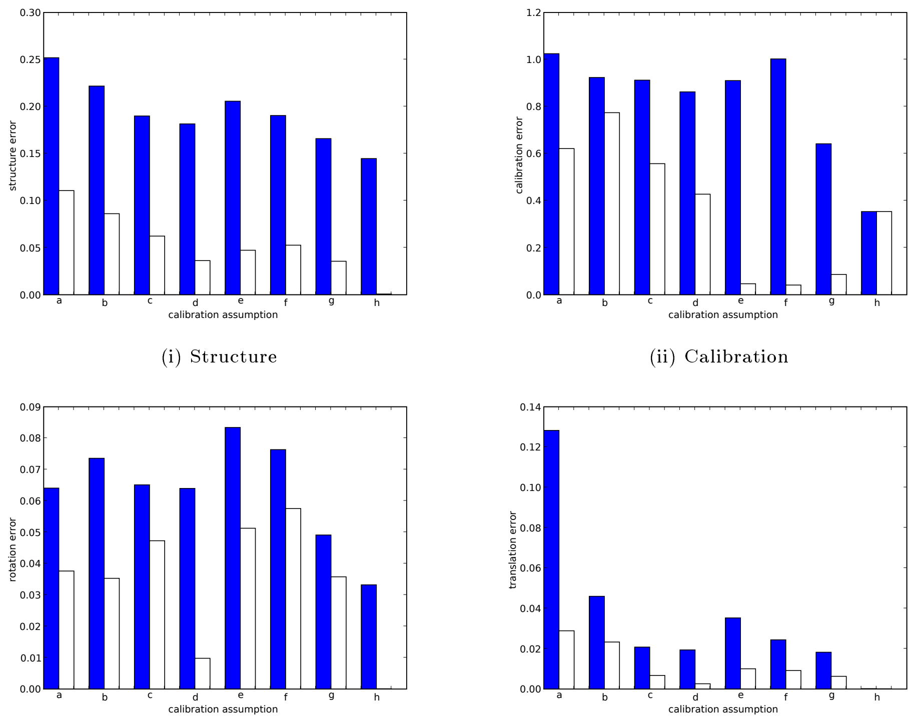
where

$$\begin{aligned} K'^f &= HK^f \\ &= \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha & s & x_0 \\ 0 & r\alpha & y_0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \sigma\alpha & \sigma s & \sigma x_0 \\ 0 & \sigma r\alpha & \sigma y_0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

In our experiment, with $K^f = I$, we have

$$K'^f = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

so that the reconstructed focal length is no longer (in our experiment) $\alpha = 1$. This is also the reason why if not all the rescaling and the scale factors are known, the focal length should not be assumed known but best left to auto-



(i) Structure (ii) Calibration

(iii) Rotation (iv) Translation

Figure 8.5: The average error (dark bar) and the error for the frame with the lowest structure error (light bar) for the pure synthetic sequence.

calibration.

The structure error is arguably the most important reconstruction criteria. We proceed to display the structure, rotation, translation and calibration values for a specific frame for one of the worst performing assumptions, assumption **a**, and one of the best performing assumptions, assumption **g**. This based on the structure error. We do this for the frame in which the assumption returned its best structure error (the smallest). In Figure 8.7 we display the values generated at frame 40 for assumption **a**, with all of its parameters fixed and none known, where it returned a structure error of $e_s = 0.1107$ with a variance of $\sigma_s^2 = 0.0007$, indicating the effect of the number of frames. In Figure 8.8 we display the values generated at frame 30 for assumption **g**, with all of its parameters known except for focal length, where it returned a structure error of $e_s = 0.0356$ with a variance of $\sigma_s^2 = 9 \times 10^{-5}$.

We see almost perfect structure for **g** while the structure of **a** visibly deviates from the ground truth. An ellipse, as given in Figure 8.6, succinctly displays all of the calibration parameters in one figure. The tilt of the ellipse is given by the skew, its center is the principal point, its x-radius equals α and the y-radius

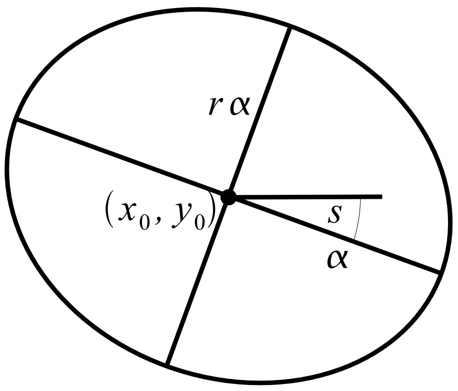
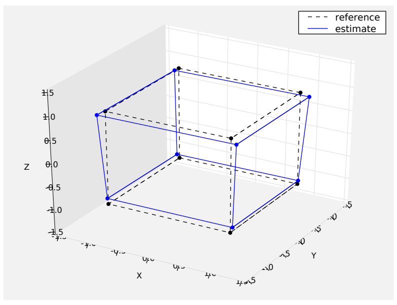


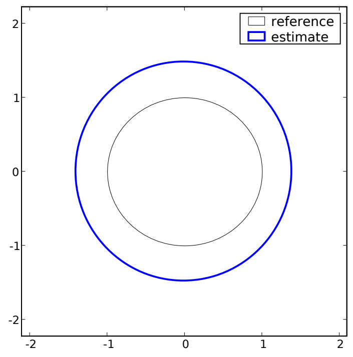
Figure 8.6: Ellipse representation of camera calibration.

equals $r\alpha$. Studying the calibration error of Figure 8.7 (ii) and Figure 8.8 (ii), for assumption **a** the sphere is slightly off-center and narrower at the sides while for **g** it is a perfect circle with a slight radius difference from the ground truth. The performance of both the rotation and translation graphs are very similar, with the translation of **g** being quite impressive. It seems the z-component of the rotation axis for both assumptions deviates slightly more than the other components, from the reference values.

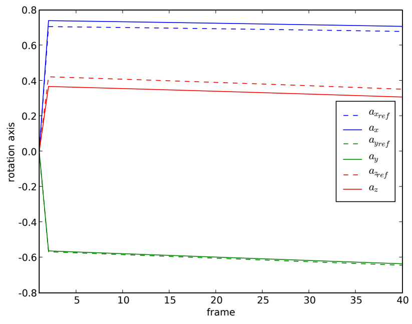
The synthetic cube experiment has made it clear that the more calibration parameters you know the better your chance of a good reconstruction. In the case of not knowing a significant number of calibration parameters, you are at least generally guaranteed a good rotation and translation reconstruction. It is further recommended to auto-calibrate the focal length. If the estimated focal length differs from the ground truth it is of no real concern since image features typically are scaled, so long as the calibration ratio r is the same.



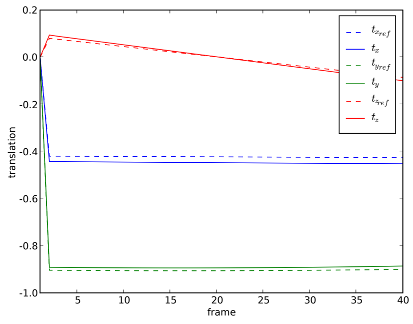
(i) Structure



(ii) Calibration

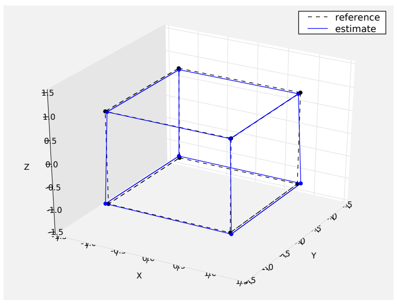


(iii) Rotation

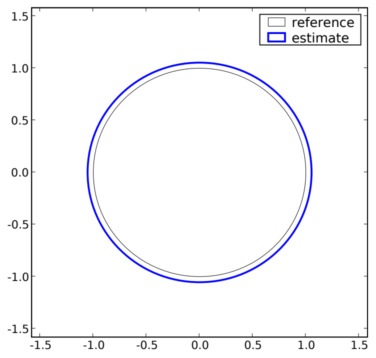


(iv) Translation

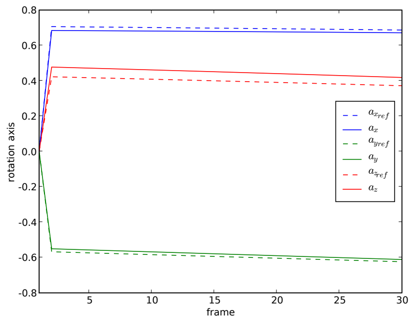
Figure 8.7: The estimated structure, rotation, translation and calibration parameters for assuming calibration **a**, given for frame 40.



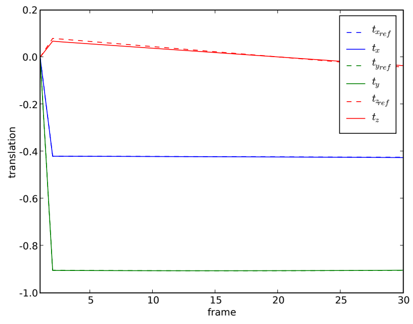
(i) Structure



(ii) Calibration



(iii) Rotation



(iv) Translation

Figure 8.8: The estimated structure, rotation, translation and calibration parameters for assuming calibration **g**, given for frame 30.

8.1.3 Synthetic cube with measurement noise

We now corrupt the synthetic motion sequence in Section 8.1.2 with measurement noise. Four different noise models are investigated, all zero-mean and Gaussian distributed with variances of 0.0001, 0.001, 0.01 and 0.1 respectively. This sub-pixel noise is added, for each frame, to the features located in the synthetic images. Note that the images are scaled to size 640×480. We perform this experiment on two calibration assumptions: the first with none of the calibration parameters known but all of them fixed (assumption **a**) and secondly all the calibration parameters known except the focal length that is assumed not to be fixed (assumption **g**).

We start with comparing the noise models in terms of structure, rotation, translation and calibration error for assumption **a**. The structure error comparison is given in Figure 8.9, the rotation error comparison in Figure 8.10, the translation error comparison in Figure 8.11 and the calibration error comparison in Figure 8.12. Again the average error over all the frames, and the error that coincides with the frame that returned the best structure error, is displayed in Figure 8.13.

The structure, rotation and calibration error against the number of frames look very similar for all the error models. The translation error against the number of frames, Figure 8.11, on the other hand show a visible increase in the error with an increase in noise, which is more prominent with the lower frames. The histograms for the structure and calibration error again don't indicate any clear deterioration with noise increase while the rotation, Figure 8.13 (iii), shows a slight error increase starting from noise model 0.001. With Figure 8.13 (iv) we can clearly see that the translation is more sensitive to noise and also has an increasing error, much more prominent than rotation, starting from noise model 0.001. It is clear the reconstruction algorithm is robust against noise, especially for structure and calibration, and only really shows an error increase with translation values.

We now proceed to display the structure, rotation, translation and calibration values for assumption **a** for the measurement noise model with variance $\sigma^2 = 0.1$ for the frame that returned best structure error. The best structure error was returned for frame 50 with $e_s = 0.0906$ and variance of $\sigma_s^2 = 0.0005$, both actually lower than the best structure error and variance for the pure synthetic case.

The cube now differs from the ground truth in that it is narrower at the top instead of wider, as in the case of zero noise. This time there is an unexpected improvement in the calibration parameter accuracy, with a slightly more accurate ratio and principal point and a much improved focal length estimate.

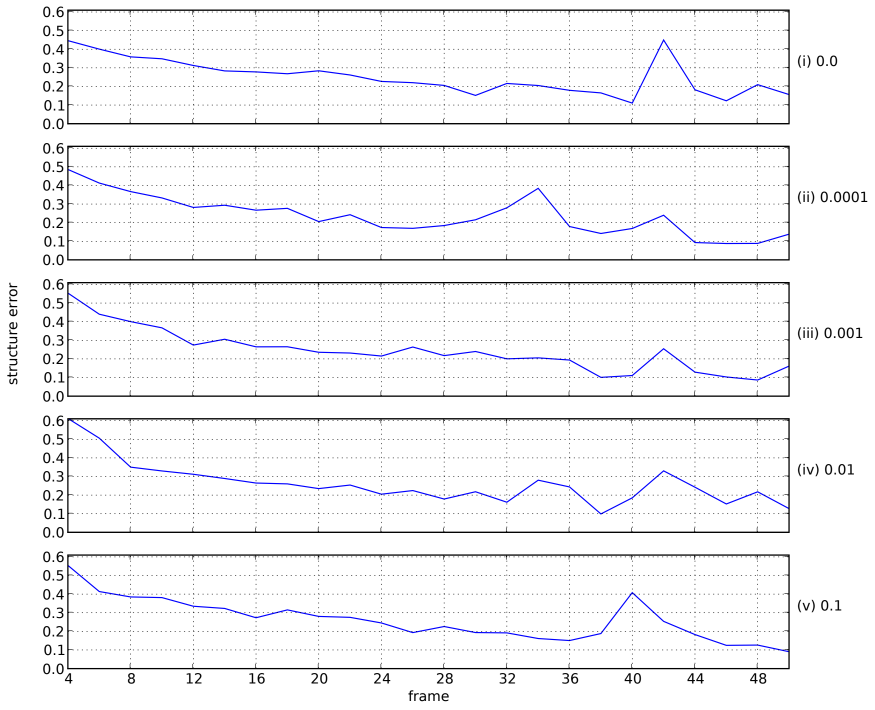


Figure 8.9: Structure accuracy of the noisy synthetic sequences assuming calibration **a**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

The noise influence is apparent in the rotation and translation in Figure 8.14 (iii) and (iv) with the zig-zag lines. The translation deviation from the ground truth is most prominent as we expected with the z-component most sensitive. The translation error variance is $\sigma_t^2 = 0.0047$ which is relatively high for the translation error of $e_t = 0.0786$.

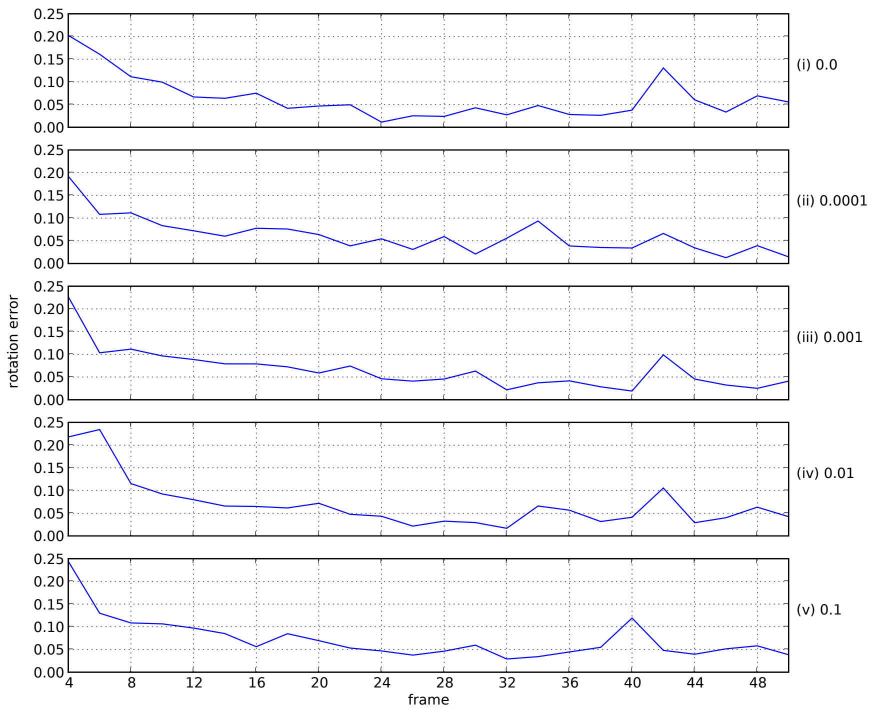


Figure 8.10: Rotation accuracy of the noisy synthetic sequences assuming calibration **a**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

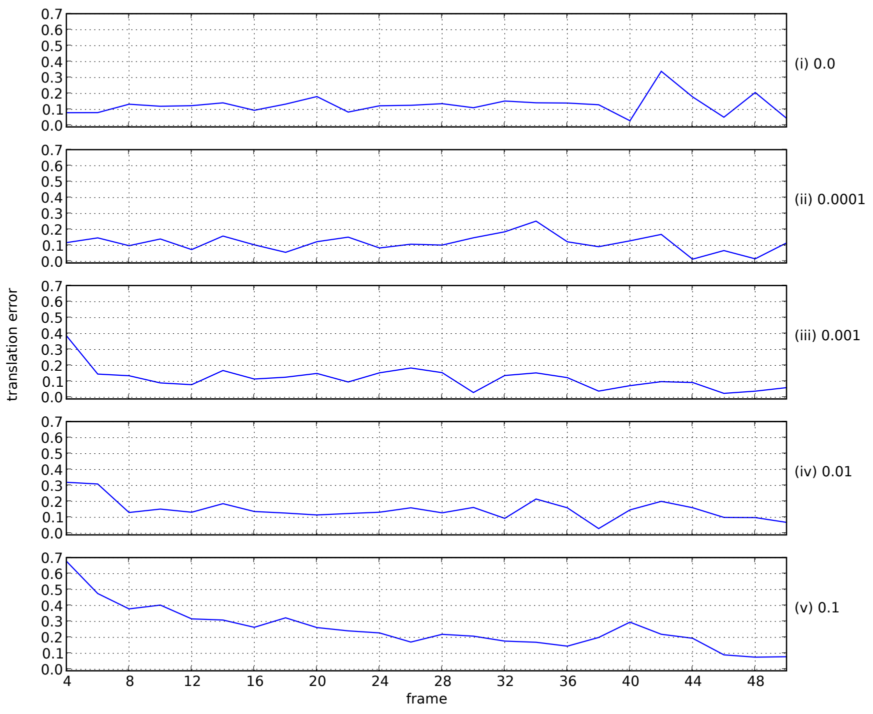


Figure 8.11: Translation accuracy of the noisy synthetic sequences assuming calibration **a**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

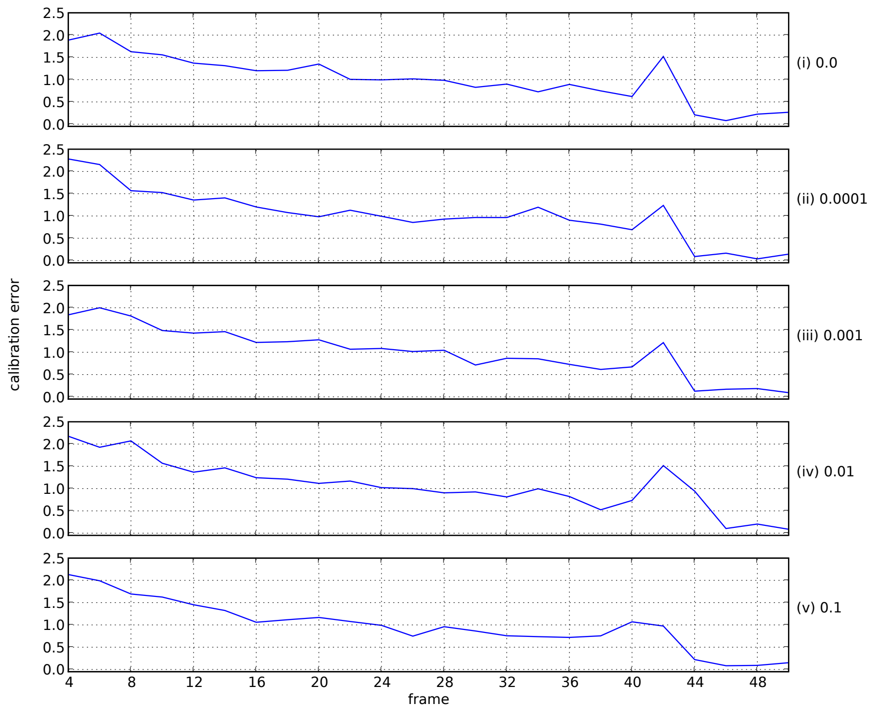
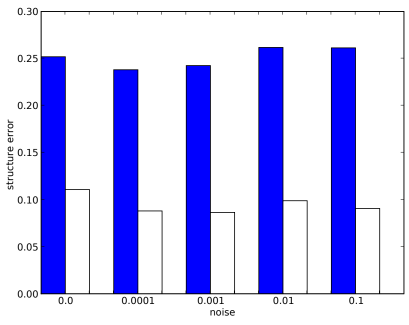
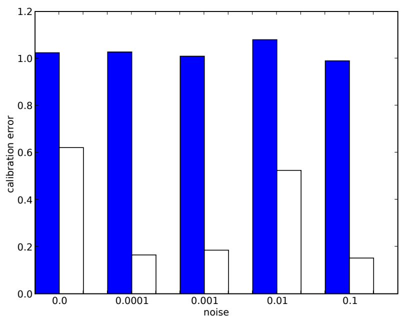


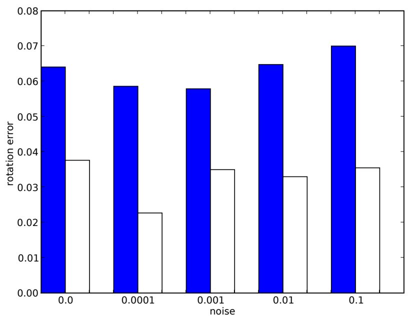
Figure 8.12: Calibration parameter accuracy of the noisy synthetic sequences assuming calibration **a**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).



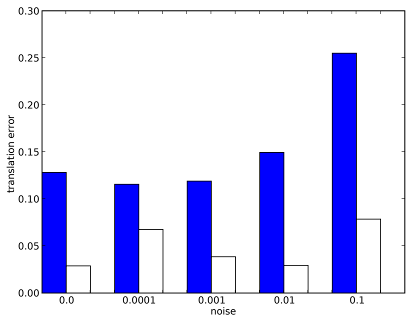
(i) Structure



(ii) Calibration

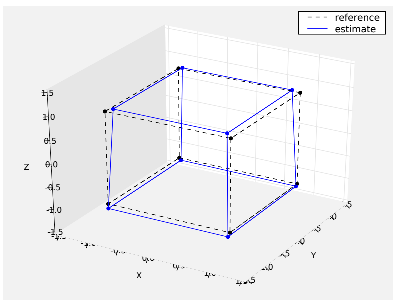


(iii) Rotation

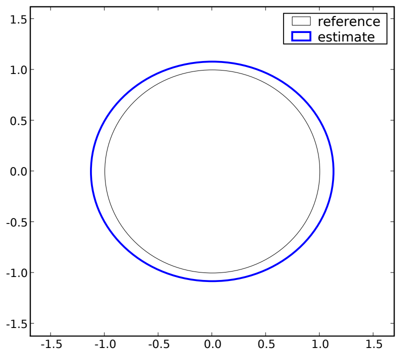


(iv) Translation

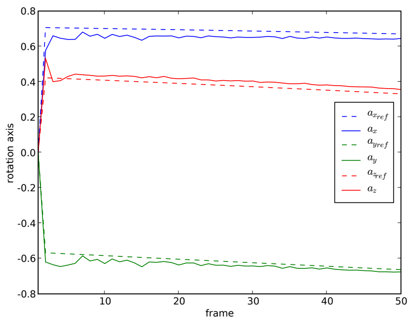
Figure 8.13: The average error (dark bar) and the error for the frame with the lowest structure error (light bar) for the noisy synthetic sequences assuming calibration **a**.



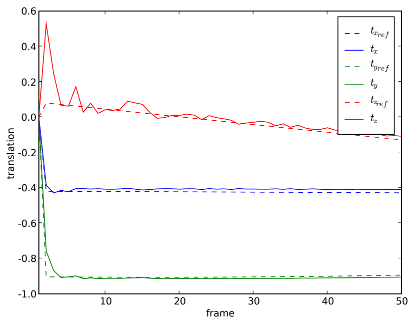
(i) Structure



(ii) Calibration



(iii) Rotation



(iv) Translation

Figure 8.14: The estimated parameters for the noisy synthetic sequence with variance $\sigma^2 = 0.1$ noise assuming calibration **a**, given for frame 50.

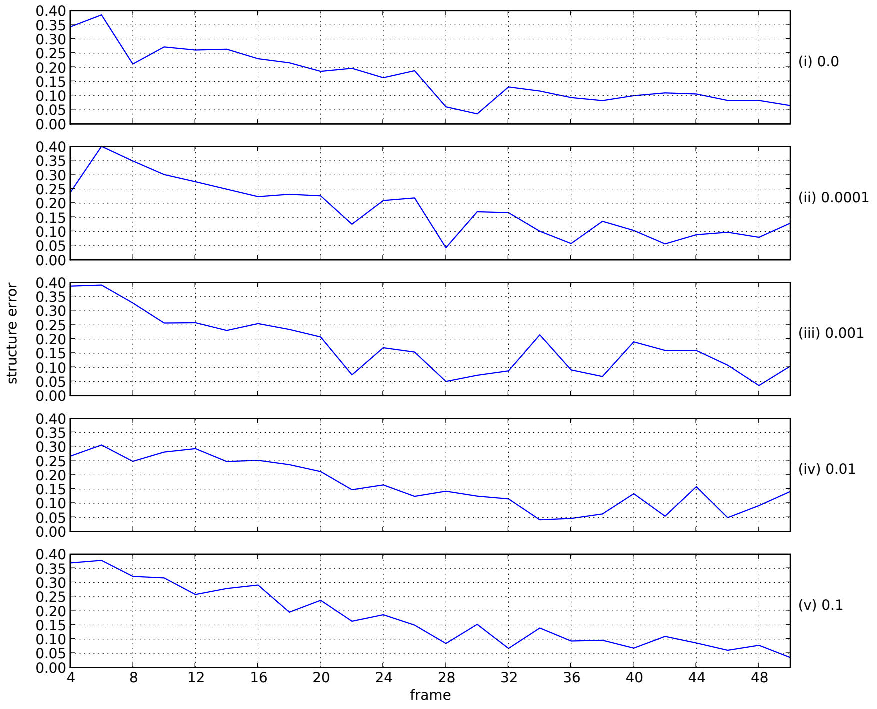


Figure 8.15: Structure accuracy of the noisy synthetic sequences assuming calibration **g**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

We now compare the noise models in terms of structure, rotation, translation and calibration error for assumption **g**. The structure error comparison is given in Figure 8.15, the rotation error comparison in Figure 8.16, the translation error comparison in Figure 8.17 and the calibration error comparison in Figure 8.18.

The results are very similar to those of assumption **a**, so that with an increase in noise there is very little difference in structure and calibration parameter accuracy, while a slight increase in rotation and a dramatic increase in translation occurs. The error increase for rotation and translation is only more prominent in assumption **g** and now especially dramatic for translation.

We again display the structure, rotation, translation and calibration values for the measurement-noise model with variance $\sigma^2 = 0.1$ for frame 50 which again has the lowest structure error. The structure error for frame 50 is $e_s = 0.0351$ which is slightly lower than the pure synthetic case and has a variance of $\sigma_s^2 = 0.0002$ which is much greater than for the pure synthetic case.

The cube in Figure 8.20 (i) looks no different than in the case for no noise

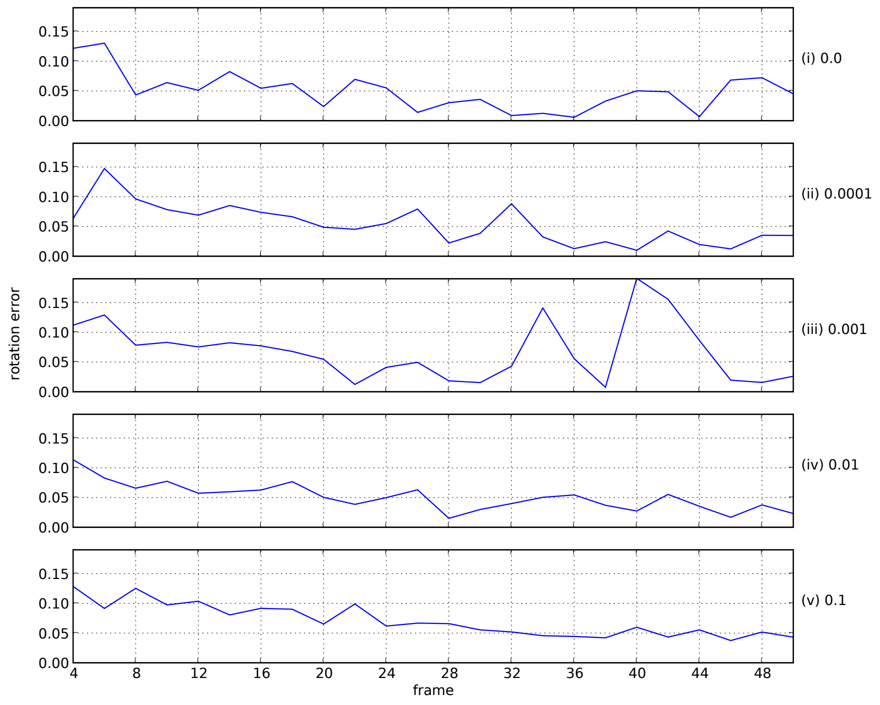


Figure 8.16: Rotation accuracy of the noisy synthetic sequences assuming calibration **g**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

and again is very similar to the ground truth. The focal length estimate is now much lower than the estimate for zero noise with the ellipse having a smaller radius. Again the noise influence is visible in the rotation and translation plots with the zig-zag lines. As with assumption **a**, the translation deviation from the ground truth is most prominent and even more so now. The sensitivity of the z-component of translation to noise is also visible here. The translation's error variance is now $\sigma_t^2 = 0.0069$ which is large compared to the translation error of $e_t = 0.0897$.

It can be concluded from this experiment, which includes controlled amounts of measurement noise in the synthetic motion sequence, that the reconstruction algorithm and its implementation is very robust in respect to measurement-noise and survives all of the noise models exceptionally well. The increase in noise is only really visible in the motion values with no clear evidence in either structure and calibration. The robustness to measurement noise would be the result of the batch-processing nature of the reconstruction algorithm. The batch-processing nature of the reconstruction algorithm leads to a very large number of values

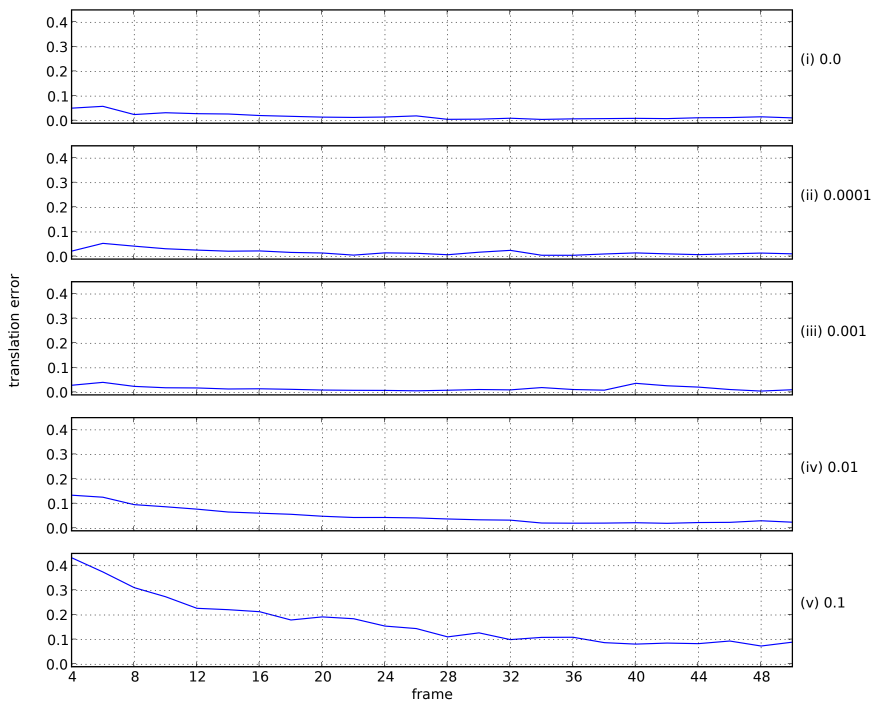


Figure 8.17: Translation accuracy of the noisy synthetic sequences assuming calibration **g**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).

involved in a single reconstruction process and the noise therefore averages out. The noise averaging out for a large number of values is especially true for the structure which is a single vector for each feature that is calculated over all of the F frames and is not value calculated for each frame such as the motion. The structure is therefore more resistant to the noise than the motion, as was observed.

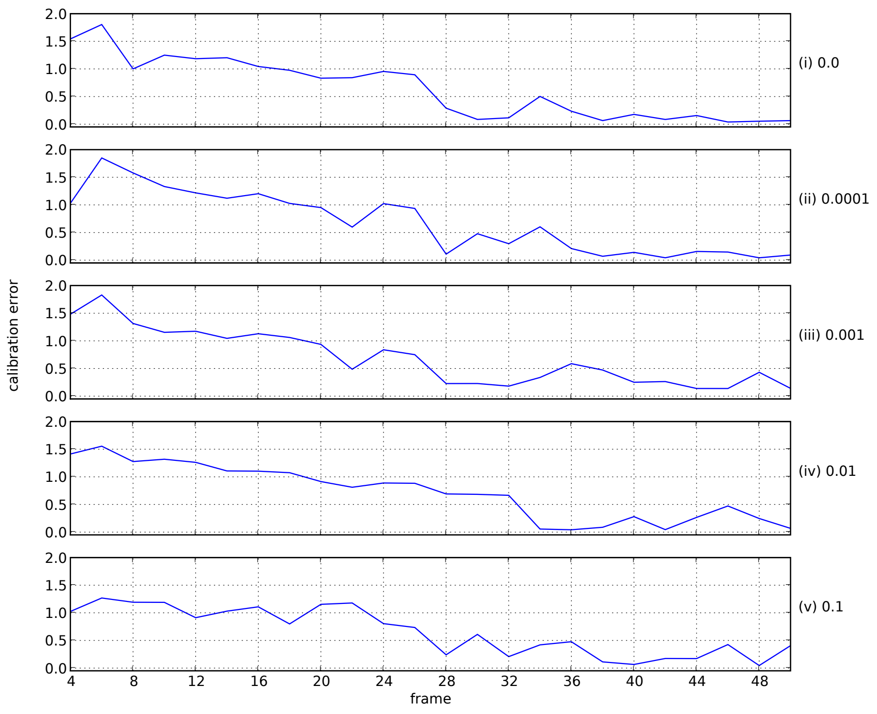
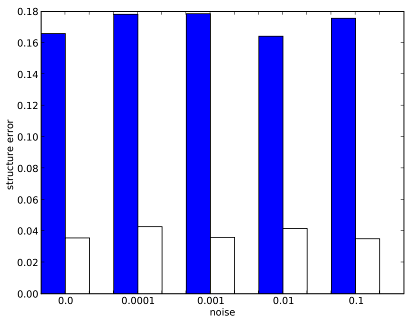
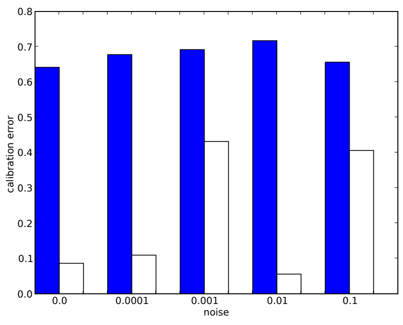


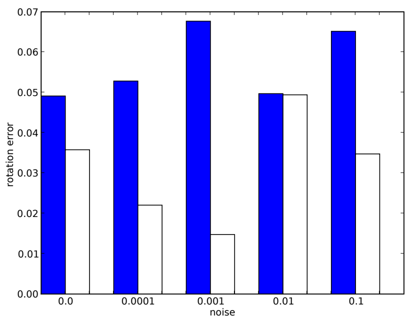
Figure 8.18: Calibration parameter accuracy of the noisy synthetic sequences assuming calibration **g**. The noise increases from noiseless in (i) to noise with a variance of $\sigma^2 = 0.1$ in (v).



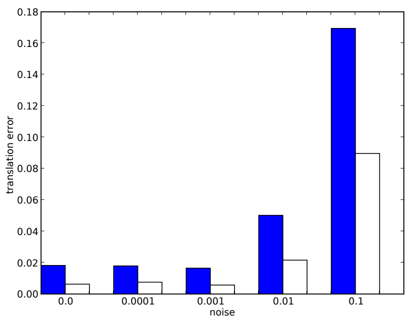
(i) Structure



(ii) Calibration

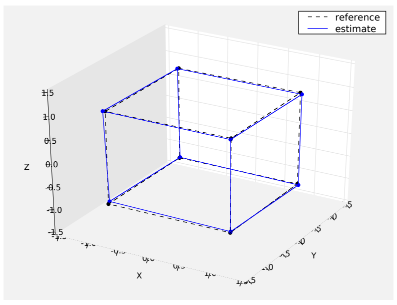


(iii) Rotation

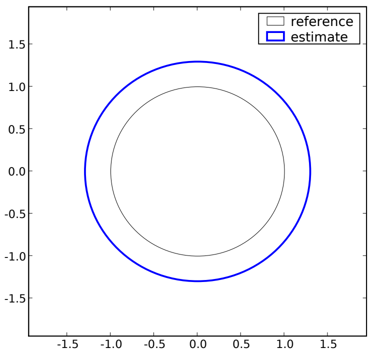


(iv) Translation

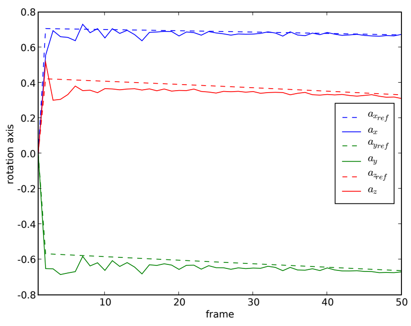
Figure 8.19: The average error (dark bar) and the error for the frame with the lowest structure error (light bar) for the noisy synthetic sequences assuming calibration **g**.



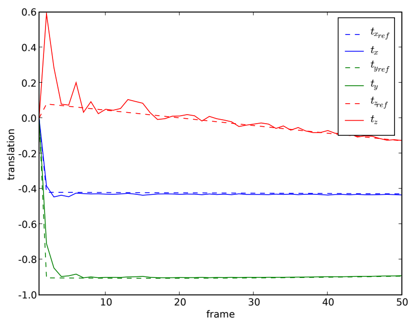
(i) Structure



(ii) Calibration



(iii) Rotation



(iv) Translation

Figure 8.20: The estimated parameters for the noisy synthetic sequence with variance $\sigma^2 = 0.1$ noise assuming calibration $\mathbf{g}$, given for frame 50.

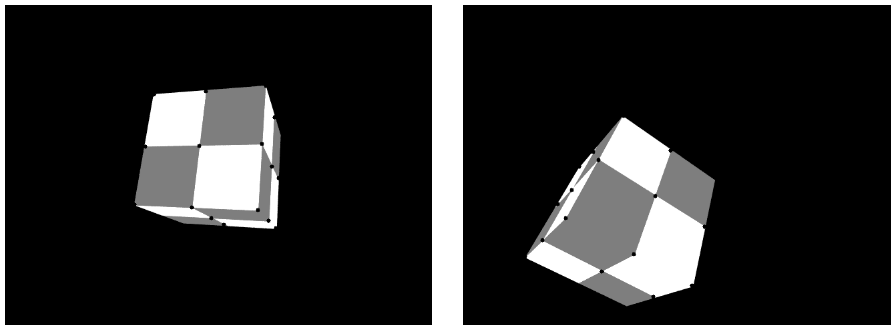


Figure 8.21: Features detected and tracked on the quasi-real cube.

8.1.4 Quasi-real cube sequence

In this experiment the cube and its motion is generated with POV-Ray and the perspective projected images all generated with POV-Ray's perspective camera model. POV-Ray is a ray tracing program available for a variety of computer platforms. The cube's structure, motion and the camera's calibration are all the same as in the synthetic experiment (and the "noise" experiment). The only difference in the motion sequence is that the cube has been rotated in the first frame to reduce feature occlusion through the sequence and to therefore provide more features to the tracker. The Lucas and Kanade (LK) tracker [18] implemented in OpenCV is used. OpenCV is a cross-platform open source computer vision library. The purpose of the LK feature tracker is to detect and track pointwise features in each frame. The cube is textured with a chess-board pattern to simplify the tracking and error calculation process by providing predictable easy-to-track features. Since this experiment makes use of non-synthetic images and a tracker, it therefore has an unknown amount of measurement-noise and the possibility of erroneous tracking and feature occlusion is a reality. The LK tracker successfully tracked 16 features, with their position in the first and last frame illustrated in Figure 8.21.

The quasi-real cube sequence is now reconstructed over the 50 frames for assumptions **a** through **g**, from Table 8.1, of the camera's calibration. Assumption **h** is not used since the scaling of the projected features in the generated images are not known. For a reconstruction at a certain frame the structure error is given in Figure 8.22, the rotation error in Figure 8.23, the translation error in Figure 8.24 and the calibration error in Figure 8.25. A set of histograms is again provided in Figure 8.26 with the dark bar the average error over all of the frames and the light bar the error that coincides with the frame that returned the lowest structure error.

The results from this experiment differ significantly from the previous two.

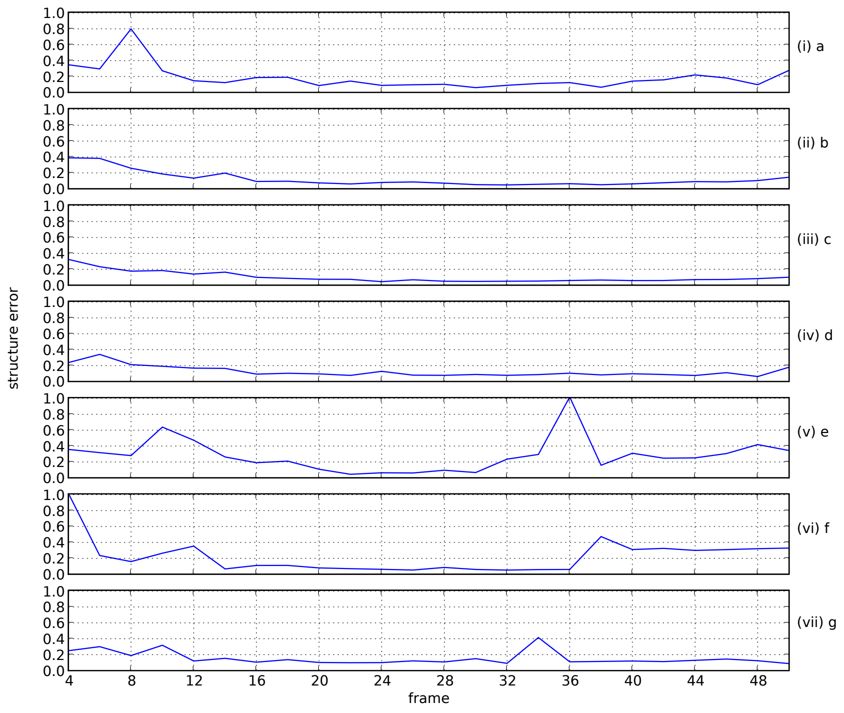


Figure 8.22: Structure accuracy of the quasi-real sequence.

For the first time the fixed calibration assumptions outperform the rest in terms of structure accuracy. With no calibration parameters explicitly known, assumption **a** has greater average structure accuracy than assumption **f** which knows both the skew and the ratio and is almost as accurate as the average of assumption **g**. Assumption **a**'s best structure error is actually lower than that of assumption **g**. We would expect that the extra parameters that are known with a fixed calibration would improve accuracy but it never had a visible influence with the synthetic tests.

Another difference from the synthetic tests is that the assumptions where the principal point is known, **d** and **g**, does not perform as well. They performed 2nd and 3rd best after assumption **h**, with all parameters known, in the pure synthetic test and now assumption **d** has a higher average error than assumption **b** which is fixed and only knows the skew. A larger error with a known principal point would be the result of the actual principal point not being where we expect it to be. We assume that measurement noise or erroneous tracking has caused the principal point to shift from where it theoretically should be, so that it is not at the origin, i.e. the precise center of the images, anymore.

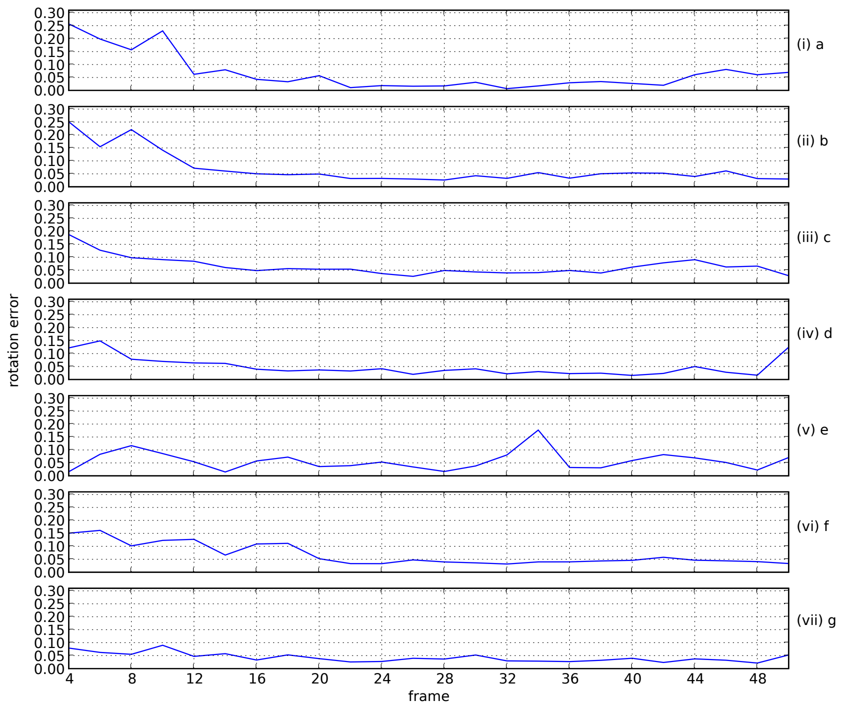


Figure 8.23: Rotation accuracy of the quasi real sequence.

The structure error still converges to a value close to or below $e_s = 0.2$ at frame 30, so the structure accuracy has not degraded significantly from a pure synthetic sequence to a quasi-real sequence. The average rotation errors has also stayed more or less the same when consulting the average rotation error in Figure 8.5 (iii) and Figure 8.26 (iii). But there is an unmistakable increase in the translation error. Where most of the average translation errors were below $e_t = 0.1$ for the pure synthetic experiment, they have all basically doubled to above $e_t = 1.5$ for the quasi-real experiment. To investigate further we again proceed to display the structure, rotation, translation and calibration values for assumption **a** and **g** for the frame where they returned the lowest structure error.

Assumption **a** returned its lowest structure error of $e_s = 0.0606$ with a variance of $\sigma_s^2 = 0.0005$ at frame 30. This is more accurate than for the best pure synthetic accuracy. The reconstructed features with a fitted flat-shaded surface is displayed in Figure 8.27 and the estimated parameters in Figure 8.28.

We observe in Figure 8.28 (i) that the structure and rotation is very close to the ground truth as the error values suggest. The estimated calibration ellipse is

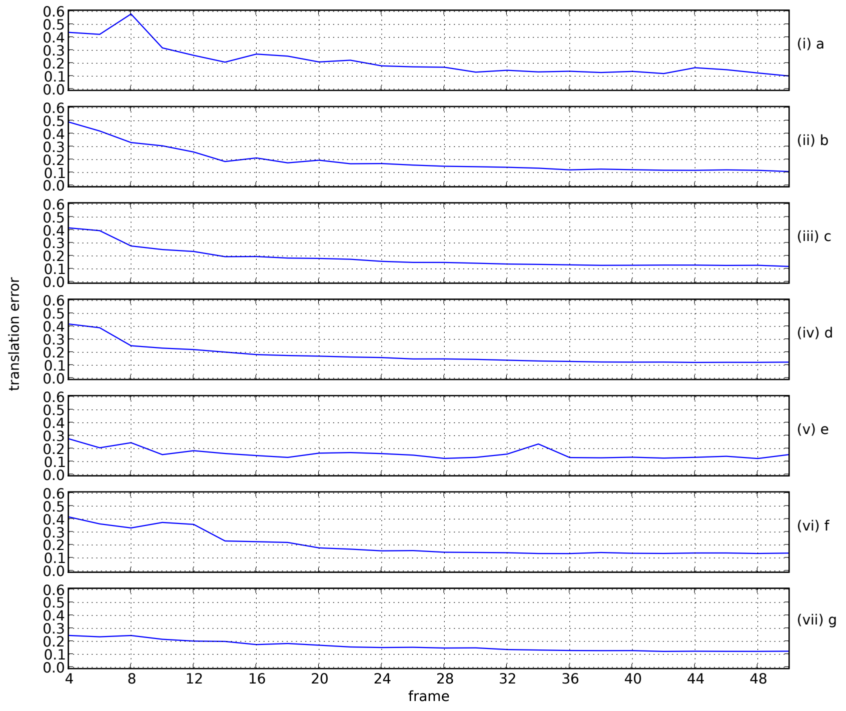


Figure 8.24: Translation accuracy of the quasi real sequence.

off-center which reinforces our suspicion that the principal point is not exactly at the center anymore. The translation figure nicely displays the large translation error values that were reported. The translation estimate is again visibly the most sensitive to noise.

We now consider assumption **g**. Assumption **g** returned its lowest structure error of $e_s = 0.0878$ with a variance of $\sigma_s^2 = 0.0007$ at frame 50. These values are much higher than those returned with the pure synthetic experiment. The reconstructed features with a fitted surface is displayed in Figure 8.29 and the estimated parameters in Figure 8.30.

The structure values deviate slightly more from the ground truth than those of assumption **a**, while the rotation value's difference from the ground truth is much more visible. The translation is again dramatically more erroneous and sensitive. The negative results from assuming to know the wrong principal point is evident in Figure 8.30 (iv).

We conclude from this experiment that the reconstruction algorithm and its implementation returns reliable results even when the feature tracking is not synthetic. When it comes to noise, the translation estimates are the most

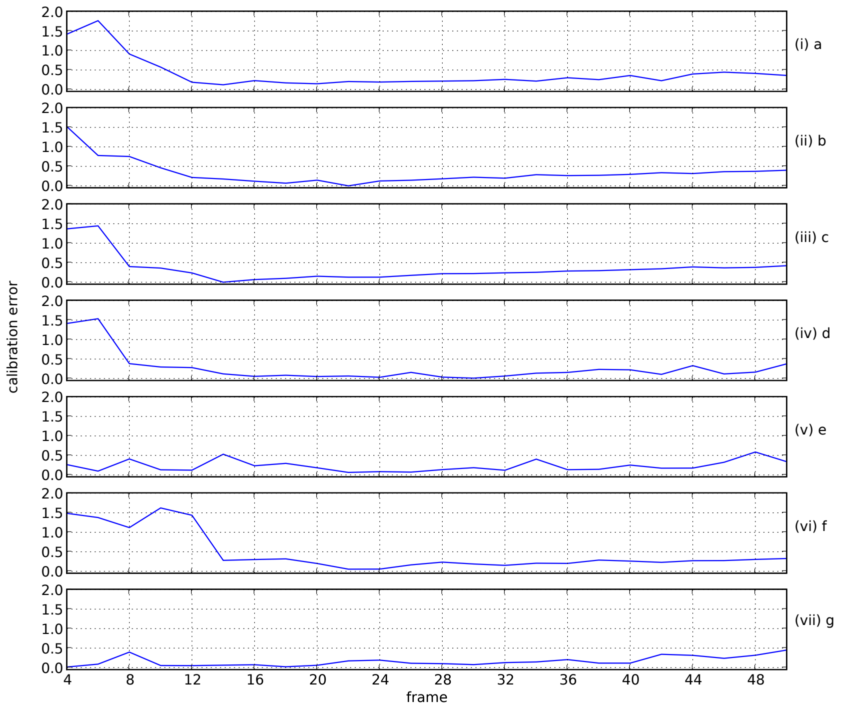
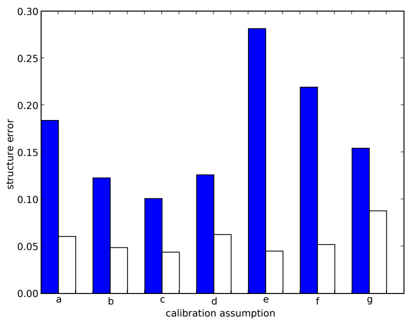
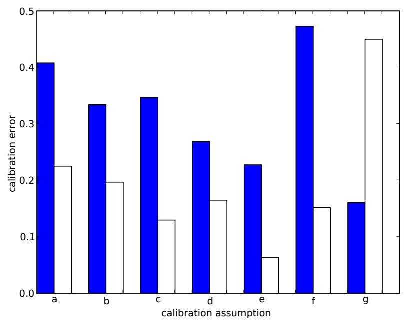


Figure 8.25: Calibration parameter accuracy of the quasi real sequence.

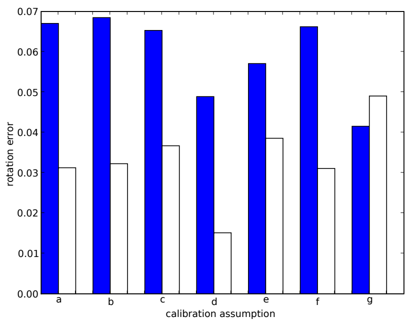
sensitive and absorb most of the error. It is further illustrated that assuming a fixed camera, when it is relevant, can improve accuracy and that auto-calibrating the principal point leads to more accurate results than just assuming to know a principal point that is not exact.



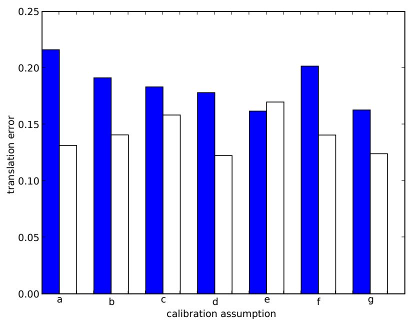
(i) Structure



(ii) Calibration



(iii) Rotation



(iv) Translation

Figure 8.26: The average error (dark bar) and the error for the frame with the lowest structure error (light bar) for the quasi-real sequence.

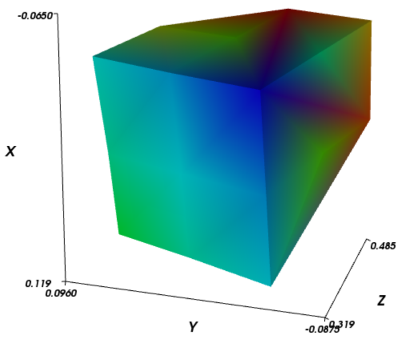
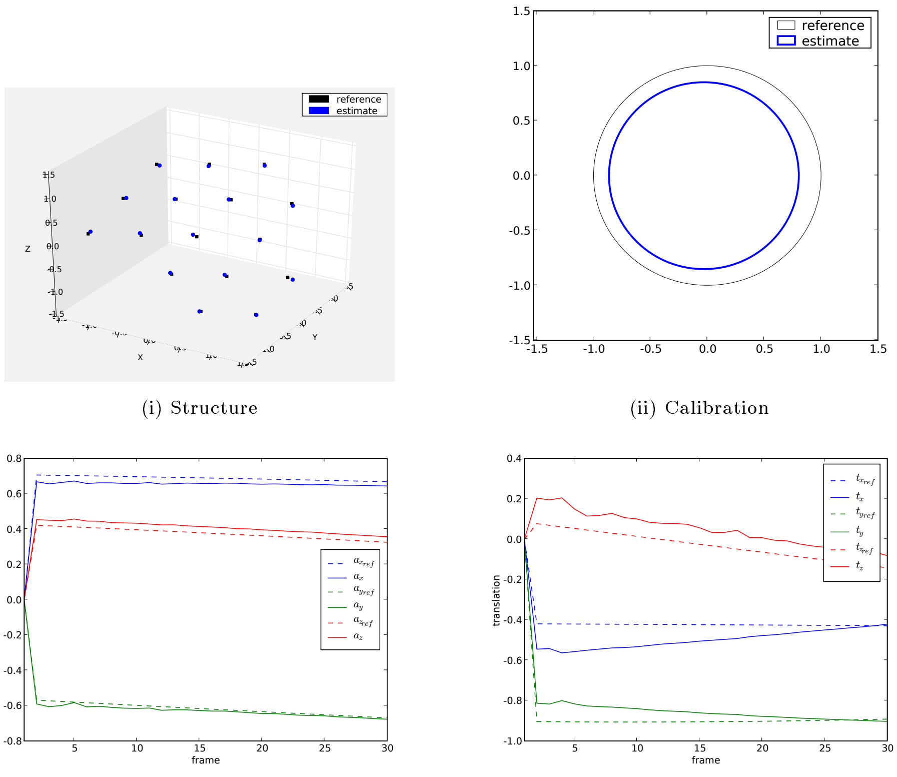


Figure 8.27: Reconstructed quasi-real cube for frame 30 assuming calibration **a**.



(i) Structure (ii) Calibration

(iii) Rotation (iv) Translation

Figure 8.28: The estimated parameters for the quasi-real sequence assuming calibration **a**, given for frame 30.

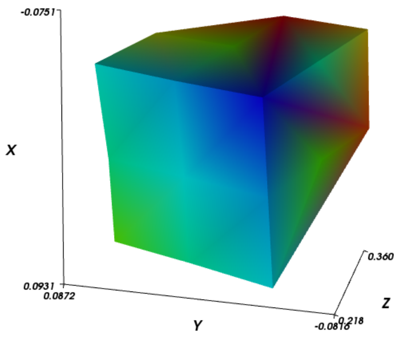
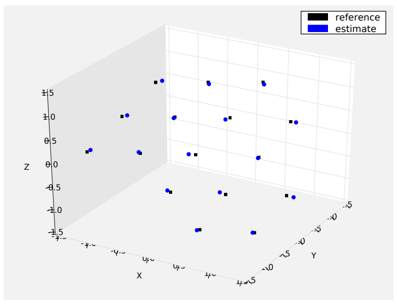
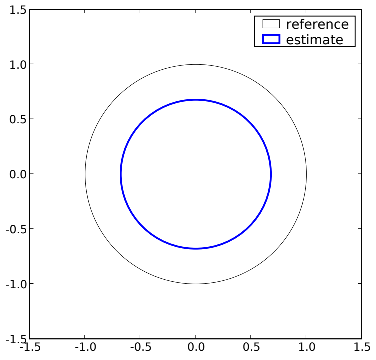


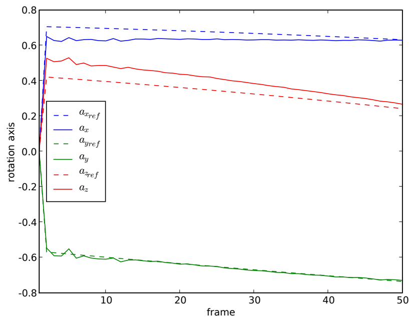
Figure 8.29: Reconstructed quasi-real cube for frame 50 assuming calibration **g**.



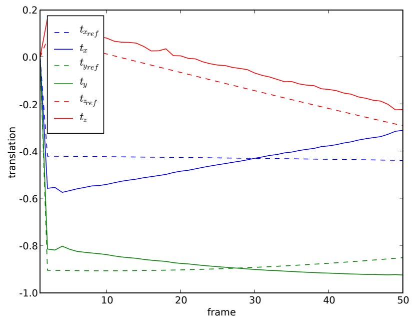
(i) Structure



(ii) Calibration



(iii) Rotation



(iv) Translation

Figure 8.30: The estimated parameters for the quasi-real sequence assuming calibration **g**, given for frame 50.

	Pure synthetic	Noisy 0.0001	Noisy 0.001	Noisy 0.01	Noisy 0.1	Quasi-real
e_s	0.1107	0.0880	0.0865	0.0989	0.0906	0.0606
σ_s^2	0.0007	0.0009	0.0008	0.0007	0.0005	0.0005
e_r	0.0377	0.0129	0.0251	0.0318	0.0384	0.0316
σ_r^2	5×10^{-6}	5×10^{-6}	4×10^{-6}	2×10^{-5}	0.0002	2×10^{-5}
e_t	0.0289	0.0677	0.0386	0.0295	0.0786	0.1314
σ_t^2	2×10^{-6}	3×10^{-6}	3×10^{-6}	0.0006	0.0047	0.0023
e_c	0.6213	0.1651	0.1852	0.5245	0.1518	0.2251
σ_c^2	0.0	0.0	0.0	0.0	0.0	0.0
Frame	40	46	48	38	50	30

Table 8.2: Summary of results for calibration assumption **a**.

	Pure synthetic	Noisy 0.0001	Noisy 0.001	Noisy 0.01	Noisy 0.1	Quasi-real
e_s	0.0356	0.0428	0.0360	0.0416	0.0351	0.0878
σ_s^2	9×10^{-5}	0.0002	0.0002	0.0001	0.0002	0.0007
e_r	0.0358	0.0222	0.0157	0.0504	0.0431	0.0526
σ_r^2	4×10^{-6}	7×10^{-6}	3×10^{-5}	1×10^{-4}	0.0007	0.0004
e_t	0.0063	0.0076	0.0057	0.0216	0.0897	0.1240
σ_t^2	1×10^{-5}	1×10^{-5}	1×10^{-5}	0.0003	0.0069	0.0013
e_c	0.0863	0.1094	0.4316	0.0557	0.4061	0.4499
σ_c^2	7×10^{-5}	2×10^{-5}	1×10^{-5}	6×10^{-6}	6×10^{-5}	5×10^{-5}
Frame	30	28	48	34	50	50

Table 8.3: Summary of results for calibration assumption **g**.

8.1.5 Comparison of calibration assumptions a and g

The results for calibration assumptions **a** and **g**, are summarized in Table 8.2 and 8.3 for easy comparison. We include the error and variance values for all the experimental sequences for the frame in which the reconstruction returned the lowest structure error. Note again that the reconstruction accuracies are better for assumption **g** with more constraints known, the reconstruction error and variance increases as the experiments become less ideal and that the structure accuracy is more stable than that of the motion.

8.2 Quasi-real face sequence

We now attempt the reconstruction of a more complex scene. Toulouse de Margerie created a 3D model of his head in POV-Ray [5] and added some motion to display the 3D structure of his model. The objective of this experiment is

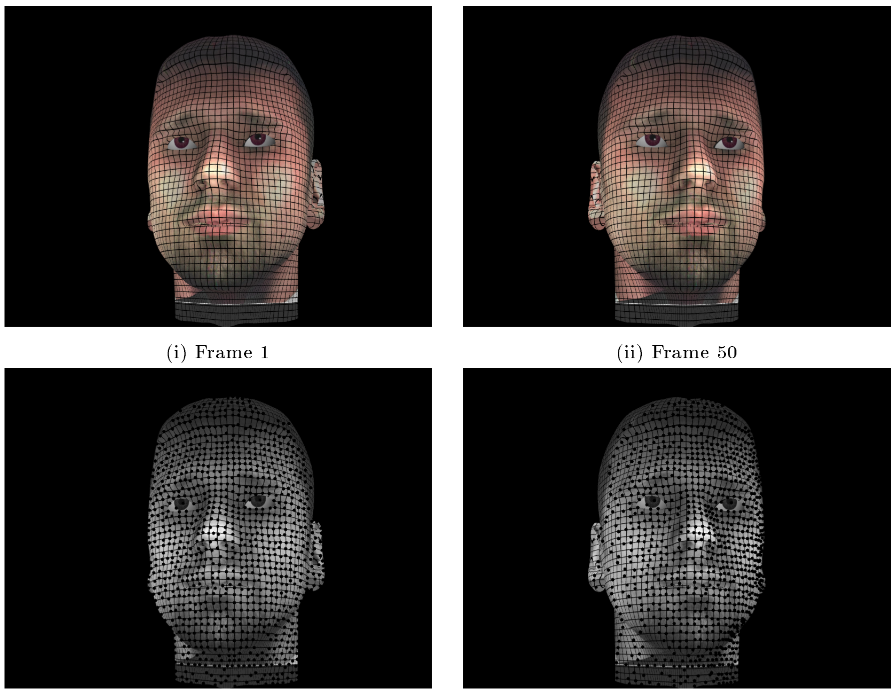
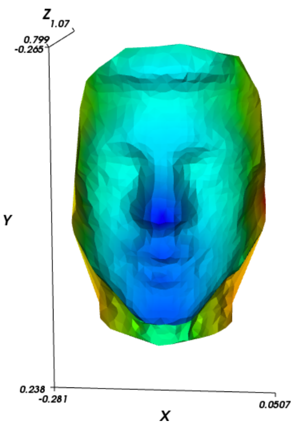


Figure 8.31: The (i) first and (ii) last frame of the head motion sequence with the tracked features.

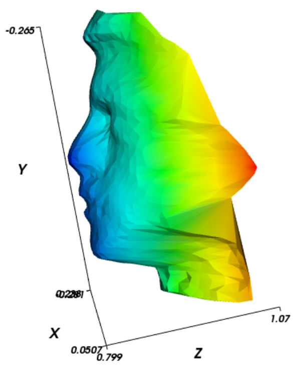
to re-create a 3D model of his face from images of a small sub-sequence of his POV-Ray motion sequence. To be able to recognize the resulting face model, a relatively large number of features need to be tracked and they have to be reasonably uniformly arranged. We therefore place a grid over the face texture.

From a 50-frame sub-sequence with a continuous clear view of the face, 1098 features were successfully detected and tracked. The first and last frame along with the tracked features are given in Figure 8.31. Neither the camera calibration and motion during the sub-sequence is known, nor the 3D position of the tracked 2D features, therefore an error comparison is not possible. Observing resemblance to the source motion sequence will have to do for an accuracy test.

The best reconstruction was obtained by assuming a camera with fixed calibration and with skew $s = 0$ and ratio $r = 1$. Auto-calibration returned a focal length of $\alpha = 0.9121$ and a principal point of $x_0 = 0.1096$ and $y_0 = 0.02778$ relative to the center of the image plane after rescaling. The re-created 3D face is shown in Figure 8.32. The model was obtained by simply fitting a flat shaded surface on the reconstructed features. Despite an arguable resemblance to Toulouse without the correct face texture applied, the 3D model is definitely a face and has the right dimensions. We can clearly make out complex facial features like eyes, nose and mouth. Applying the reconstruction algorithm to

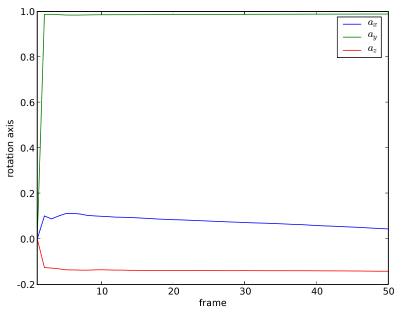


(i) Front view

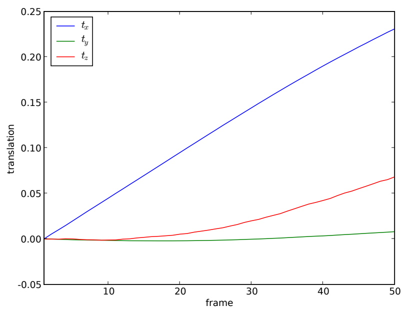


(ii) Side view

Figure 8.32: Reconstructed head structure assuming a fixed camera calibration with skew $s = 0$ and ratio $r = 1$.



(i) Rotation axes



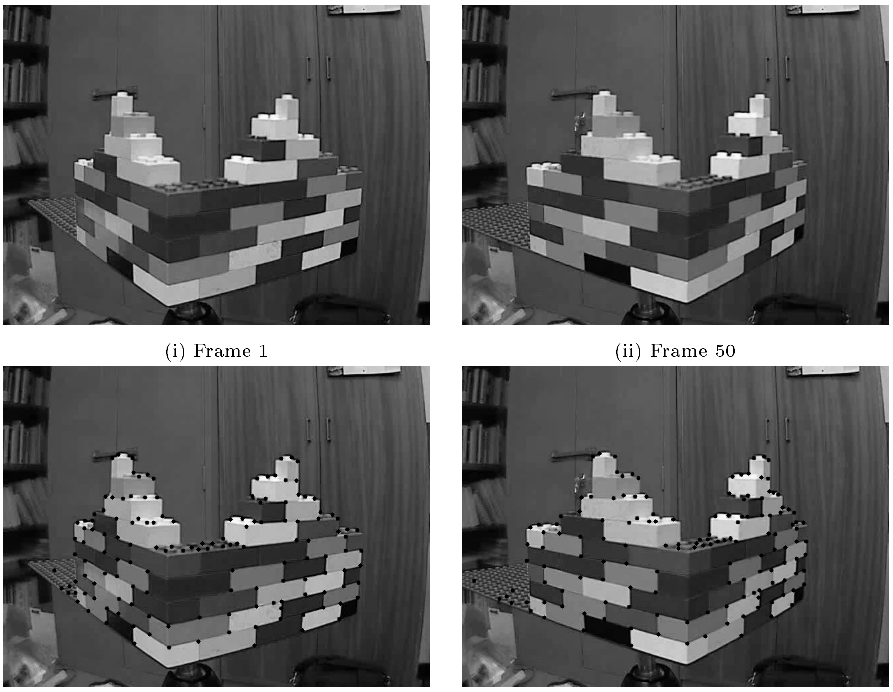
(ii) Translation

Figure 8.33: Reconstructed head motion assuming a fixed camera calibration with skew $s = 0$ and ratio $r = 1$.

a denser arrangement of more features and applying the face texture, the 3D model is sure to have undeniable resemblance.

With the ground truth of the motion not known, the calculated rotation axes and translation are plotted for each frame in Figure 8.33. Note that the translation values are not normalized since nothing is being compared, unlike with the previous translation plots. The lines are smooth in the motion figures and are representative of the smooth motion of the head. The motion in the sub-sequence resembles orbital motion around the y-axis which is visible with the significantly higher a_y values of the rotation axes. The reconstruction further interpreted a slight translation in the x direction.

This experiment demonstrates that the reconstruction algorithm manages more complex structures and can be applied with confidence.



(i) Frame 1 (ii) Frame 50

Figure 8.34: The (i) first and (ii) last frame of the Lego motion sequence with the tracked features.

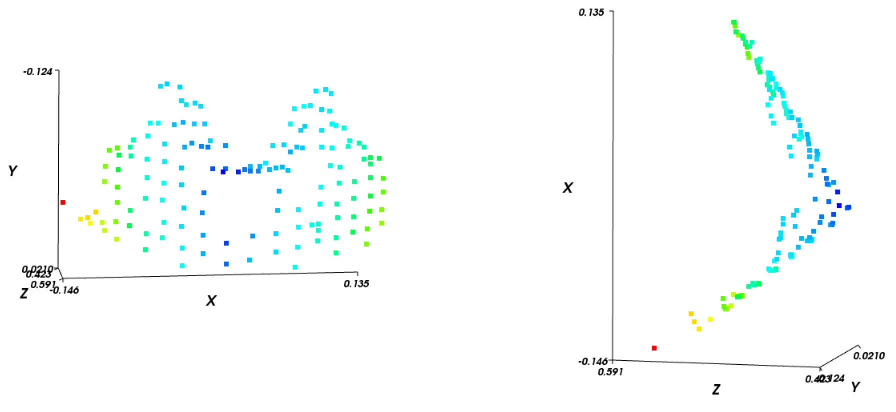
8.3 Real sequences

This section covers two more reconstruction experiments on real sequences that thoroughly test the capability of the SfM algorithm. The first experiment is performed on a sequence of a structure built from Lego. The final experiment is from an archaeological excavation recorded on a hand-held camera.

8.3.1 Lego sequence

This experiment examines the reconstruction of a Lego structure rotating on a turntable, i.e. orbital motion, for 50 frames. The Lego structure along with the tracked features can be seen in Figure 8.34. The LK feature tracker successfully tracked 145 robust features.

Inspection of Figure 8.34 shows that the straight real-world vertical lines are curved, i.e. barrel distorted, as if mapped around a sphere. This effect is especially visible with the right vertical edge of the right-hand cabinet. This distortion is the result of *lens distortion* — a common occurrence when working with real images and is not present in synthetic or quasi-real motion sequences (unless simulated on purpose). Lens distortion is not included in our camera model, nor is it rectified with the implementation of the reconstruction algorithm



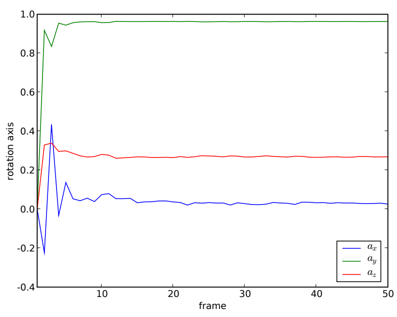
(i) Front view (ii) Top view

Figure 8.35: Reconstructed Lego structure assuming a fixed camera calibration with skew $s = 0$ and ratio $r = 1$.

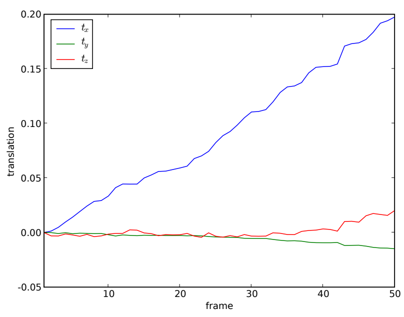
itself. Since we use a linear camera model (no radial distortion) it is of interest to learn how much radial distortion affects the reconstruction. Of course it is possible to remove radial distortion through a calibration process — the main idea behind this study is to investigate the use of uncalibrated cameras.

With no ground truth available for this experiment, the reconstructed structure is compared with the images to get a sense of accuracy. Based on this criteria, the most accurate reconstruction was the one obtained by assuming that all the cameras have a fixed calibration with skew $s = 0$ and ratio $r = 1$. The camera calibration was further estimated by auto-calibration to have a principal point of $x_0 = -0.0029$ and $y_0 = 0.1087$ relative to the center of the image plane after rescaling and focal length, in terms of the pixel dimension in the x-direction, of $\alpha = 0.9574$. The reconstructed structure is displayed in Figure 8.35. The reconstructed structure looks very convincing with no obvious deviation from the images or outlier features. Figure 8.35 (ii), the top view, is especially impressive — illustrating the large depth difference that was reconstructed and the right angle between the two sides of the Lego structure.

The reconstructed motion is illustrated in Figure 8.36 by plotting the rotation axes and translation. Since there is no ground truth available the validity of the estimates motion cannot be proved. Again, with no comparison taking place, the translation values are not normalized. The Lego motion sequence consists of orbital motion about the y-axis which is visible in the rotation plot with its large values for the y-component of the rotation axes. There is also a slight z-component to the rotation axes that could mean that the rotation platform is slightly tilted. The rotation is also clearly constant according to the plot. The translation reconstruction shows a slight translation in the x-direction



(i) Rotation axes



(ii) Translation

Figure 8.36: Reconstructed Lego motion assuming a fixed camera calibration with skew $s = 0$ and ratio $r = 1$.

which may, or may not, be correct for the motion sequence with orbital rotation about the y-axis. The turntable was hand-cranked and this can be seen with the slightly jerky lines in the motion figure.

The reconstruction of the structure and motion of the real Lego motion sequence looks to be a success when using the source images as comparison, with no ground truth available. This in spite of the presence of lens distortion.

8.3.2 Medusa sequence

The second real test sequence was obtained from an archaeological excavation in the ancient city of Sagalassos in Turkey. This experiment uses 100 frames of a video of the head of Medusa decorating a fountain [24]. This video was recorded by a person walking with a hand-held camera and the reconstruction will surely test the versatility and robustness of our algorithm. The LK feature tracker successfully detected and tracked 728 features over the 100 frames. The Medusa head, along with the tracked features, can be seen in the first and last frame of the sequence in Figure 8.37. Note that since this is a real test sequence with no artificial markers, the reconstruction does not include many visible facial features.

Again we have no ground truth data and compare the reconstruction with the source video for a measure of accuracy. The reconstruction most resembling the source video was obtained by assuming a camera with skew $s = 0$, ratio $r = 1$ and the rest of the calibration parameters not fixed. The choice of skew $s = 0$ and ratio $r = 1$ seems to be a reliable assumption for unknown calibration. Auto-calibration calculated a focal length, in terms of the pixel dimension in the x-direction, of $\alpha = 0.861$ and the principal point of $x_0 = 0.0056$ and $y_0 = 0.0369$, relative to the center of the image plane after rescaling, for the the reference frame. A front and side view of the reconstruction is given in Figure 8.38. The

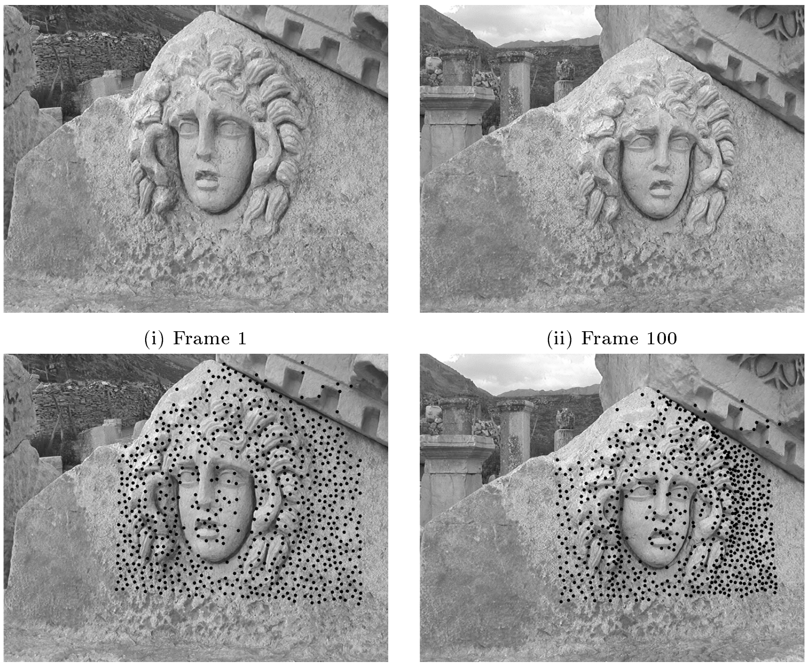
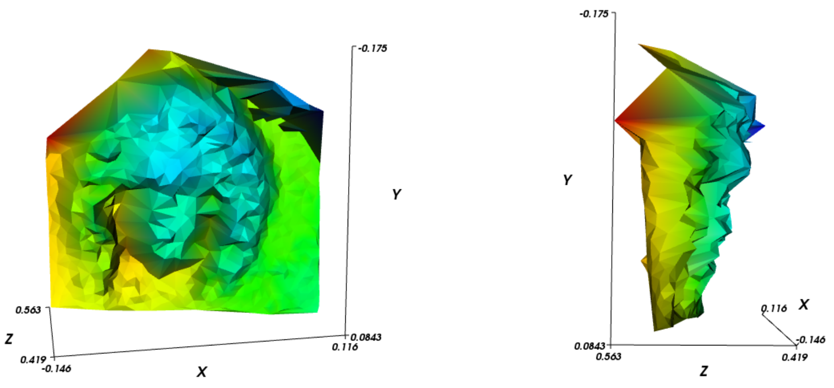


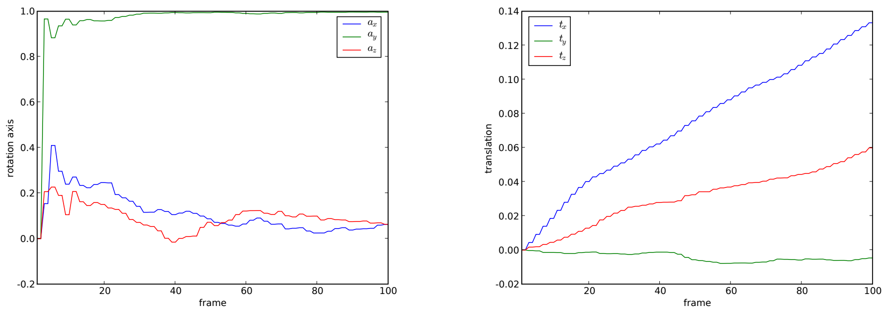
Figure 8.37: The (i) first and (i) last frame of the Medusa motion sequence with the tracked features.



(i) Front view (ii) Side view

Figure 8.38: Reconstructed Medusa structure assuming a camera calibration with skew $s = 0$ and ratio $r = 1$.

reconstructed features are fitted with a flat-shaded surface to make the scene more visible. The Medusa head is clearly visible on the reconstructed wall, in Figure 8.38 (i), with the hair nicely textured. The right half of the face can also be seen in the reconstruction with a hint of the eye and mouth. The side view in Figure 8.38 (ii) gives a nice idea of the depth of the face.



(i) Rotation axes (ii) Translation

Figure 8.39: Reconstructed Medusa motion assuming a camera calibration with skew $s = 0$ and ratio $r = 1$.

The reconstructed motion for each frame is given in Figure 8.39 with the rotation axes in Figure 8.39 (i) and the translation in Figure 8.39 (ii). The lines in the figure are very jerky which accurately represents the shakiness of the video filmed by a walking person. The source video displayed orbital motion about the y-axis and translation in the x-direction, with the cameraman focusing on the Medusa while walking around it, and this is clearly visible in the motion plots with the large y-component of the rotation axes and the slight translation in the x-direction.

This experiment confirms the robustness and the versatility of our algorithm since a real motion sequence from a hand-held camera, of unknown calibration, in motion can be accurately reconstructed.

8.4 Summary of the metric reconstruction results

The metric reconstruction system was tested on various data sets, varying from the ideal to the unfavorable and the ill-defined in terms of measurements, feature tracking and calibration parameters. From these metric reconstruction experiments we observe that the reconstruction algorithm and its implementation performs well for synthetic, noisy, quasi-real and real data.

We conclude from the series of cube experiments that the more calibration parameters you know, the greater the reconstruction accuracy – but, if the principal point (x_0 and y_0) and the focal length in terms of the pixel dimension in the x-direction (α) are not exactly known, better results are obtained by auto-calibrating them instead of assuming a wrong value. For all of the sequences where the camera calibration was unknown, the best results were obtained by assuming a calibration of skew $s = 0$ and ratio $r = 1$. This assumption is therefore recommended for reconstructions where the camera calibration is unknown. We further observe that the structure accuracy stays more or less the same for less ideal experiments while the motion error and error variance increase as the experiments become less ideal. The quasi-real, face sequence results showed that the system is capable of reconstructing complex structures such as facial features. Finally the real-world sequences show that the system is robust and versatile enough to reconstruct real scenes under real conditions with unknown calibration.

This completes our metric reconstruction experiments; next we focus on motion segmentation.

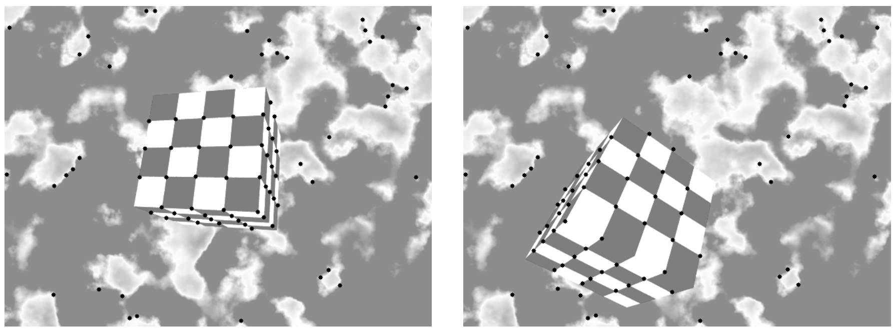

(i) First frame (ii) Last frame

Figure 8.40: Features detected and tracked on the quasi-real cube.

8.5 Cube Sequence for motion segmentation

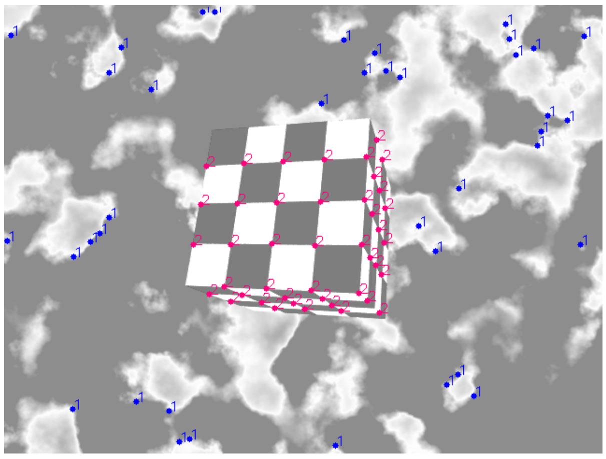
In this section we again use quasi-real cubes, except here the motion segmentation algorithm and its implementation will be put to the test. This section consists of two tests. In the first we attempt to separate the features on a cube in motion from those of the stationary background and in the second we attempt to segment the features from four cubes with different motion.

8.5.1 Background removal

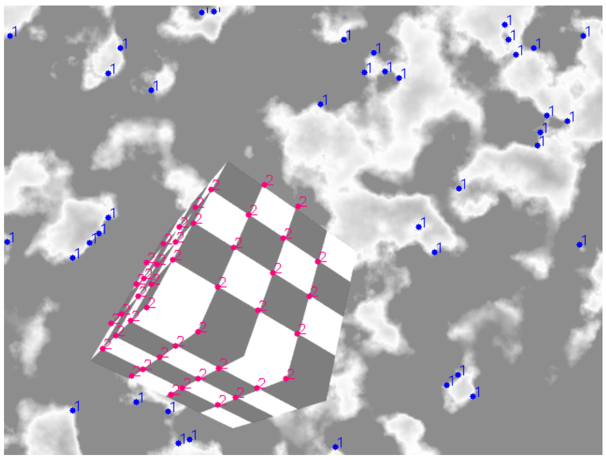
The motion sequence used in this experiment is the same one from Section 8.1.4, the only difference being that the motion now takes place in front of a stationary background (added with POV-ray). The LK feature tracker successfully tracked 84 features. These features belong to both the stationary background and the cube in motion and have to be segmented before 3D reconstruction is possible. The segmentation of foreground features (cube features) from those of a background is also called *background removal.* The first and last frames of the scene, along with the tracked features, are shown in Figure 8.40. We now apply motion segmentation.

The result of motion segmentation, in the first and last frame, is visible in Figure 8.41. The features were successfully segmented into 39 blue features belonging to the background (referred to as motion 1 in the figure) and 43 red features belonging to the cube (referred to as motion 2 in the figure). Outlier removal did not remove any features and neither were there any misclassified features.

The perfect result of this experiment clearly shows that the motion segmentation is more than capable of segmenting motion from a stationary background under near-ideal circumstances.



(i) First frame



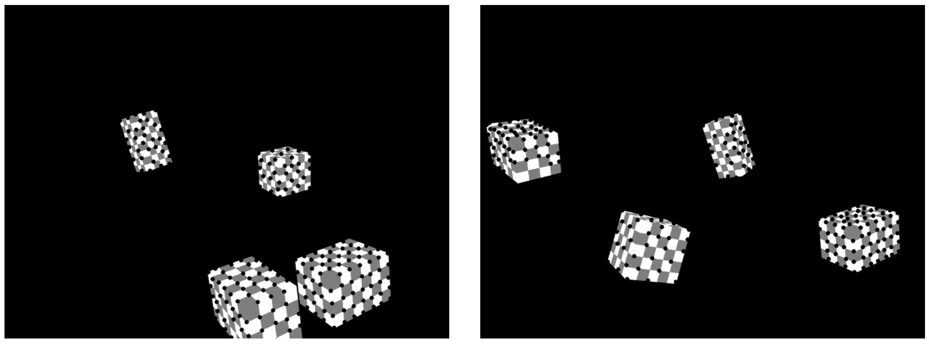
(ii) Last frame

Figure 8.41: Motion segmentation of quasi-real cube.

8.5.2 Multiple motion segmentation

We now face a more complex problem. This experiment attempts to segment the features of four cubes with different motions. This motion sequence consists of 50 frames of 640 $\times$ 480 images created with POV-Ray. The LK tracker successfully detected and tracked 191 features that belong to all four of the cubes. The first and last frame of the scene along with the tracked features can be seen in Figure 8.42. We now apply motion segmentation.

The motion segmentation result for the first and last frame is visible in Figure 8.43. The algorithm segmented 72 features to cube 1, 29 features to 2, 48 features to cube 3 and 29 features to cube 4. This leaves 13 features that are not accounted for and therefore were thrown out in the outlier removal process. On closer inspection of the outliers, 12 were correctly assigned and unnecessarily



(i) First frame (ii) Last frame

Figure 8.42: Features detected and tracked on 4 quasi-real cube.

removed. The last remaining feature was assigned to cube 2 while located on cube 3 and was successfully picked up by outlier removal.

The motion segmentation of quasi-real motions, based on this experiment, can be considered reliable since after outlier removal all of the features were assigned to their correct motion.

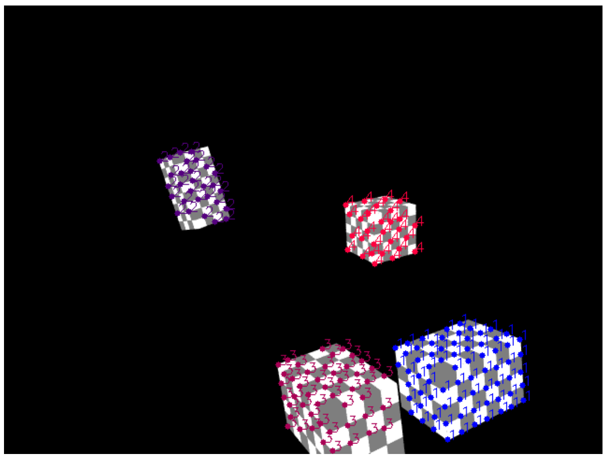
8.6 Real sequence for motion segmentation

We now attempt motion segmentation for real motion sequences. The objective of this experiment is to segment the rotating Lego structure of Section 8.3.1 from the background. The LK feature tracker tracked 230 features, a mix of Lego and background, that can be observed for the first and last frame in Figure 8.44.

The motion segmentation results presented for the first and last frame can be seen in Figure 8.45. The algorithm assigned 86 blue features to the background (referred to as motion 1 in the figure) and 143 red features to the Lego (referred to as motion 2 in the figure). This leaves one feature thrown out with the outlier removal procedure. At closer inspection this feature was correctly assigned and unnecessarily removed.

Examining Figure 8.45 we observe that 2 features on the upper left corner of the Lego are erroneously assigned to the background. Fortunately none of the background's features were assigned to the Lego and it (the Lego structure) can still be fully reconstructed without a problem. The background, on the other hand, undergoes no motion and therefore a metric reconstruction is anyway not possible.

The motion segmentation performed reasonably well for this real sequence with metric reconstruction still possible for the result. Still, an inspection of the motion segmentation results for real sequences is recommended, since it is


(i) First frame

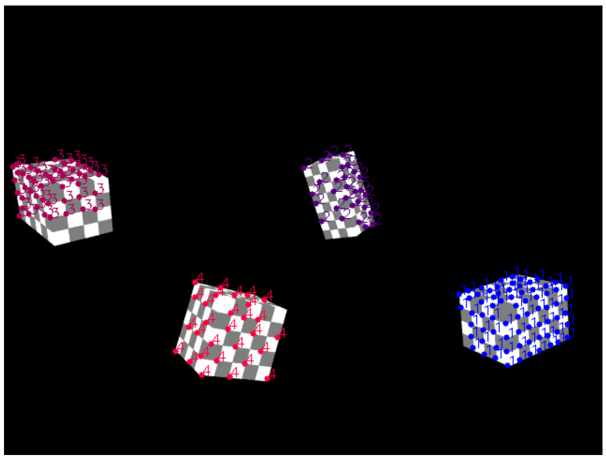
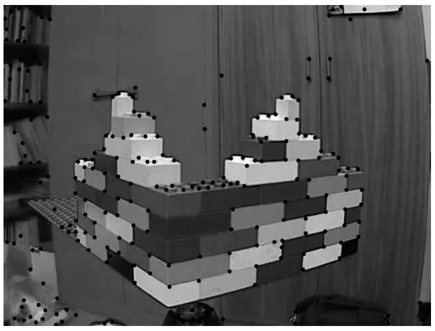

(ii) Last frame

Figure 8.43: Motion segmentation of 4 quasi-real cube.


(i) First frame

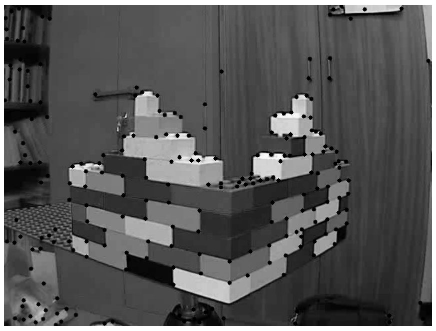
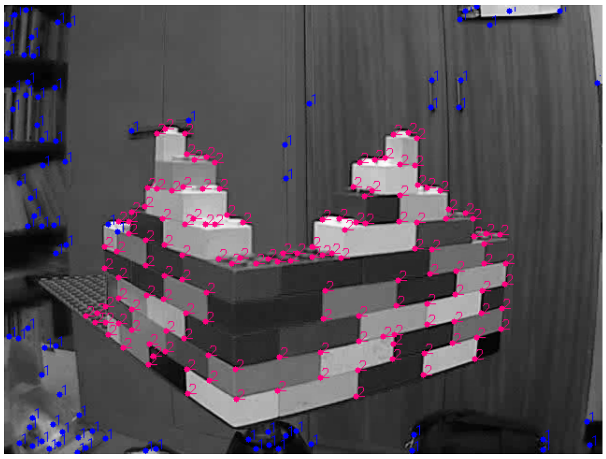

(ii) Last frame

Figure 8.44: Features detected and tracked on Lego.


(i) First frame

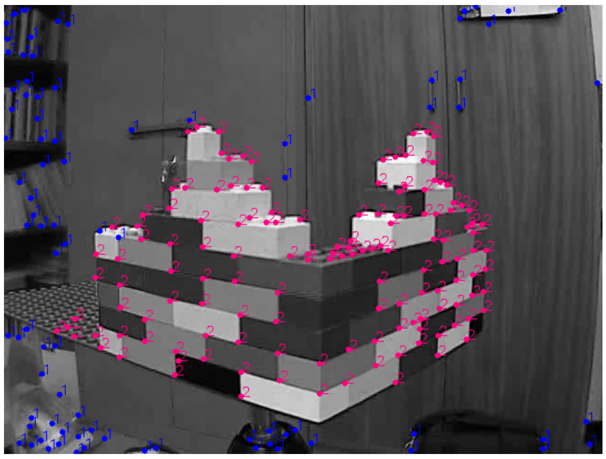

(ii) Last frame

Figure 8.45: Motion segmentation of Lego.

possible for the outlier removal process to miss misclassified features.

8.7 Summary of the motion segmentation results

The motion segmentation algorithm and its implementation, just as the metric reconstruction system, were exposed to various types of data. The experimental results are encouraging, with good performance in the quasi-real experiments and slightly worse performance for the real experiment. The motion segmentation system is reliable for background removal and multiple motion segmentation for quasi-real data, but have to be double-checked for real data. The removal of potential outliers after motion segmentation was shown to be of great importance.

This concludes the motion segmentation experiments and the chapter on experimental investigation.

Chapter 9

Conclusion

This thesis set out to recover the motion and 3D structure of multiple rigid objects, each moving independently, up to a similarity transformation. These objects are observed as collections of features tracked in a single-camera video sequence, or images from multiple cameras. We assume a perspective camera model but do not require the cameras to be calibrated.

Two popular SfM approaches exist—the Kalman filter approach introduced by Azarbayejani and Pentland [1] and the matrix factorization approach introduced by Tomasi and Kanade [34]. The Kalman filter approach is a sequential algorithm that does not allow for motion segmentation. It also assumes a small change in feature positions between frames, takes a while to converge and assumes a camera (or cameras) with fixed calibration. The matrix factorization algorithm, on the other hand, is typically a batch processing algorithm. It allows for motion segmentation and readily accepts images from very different viewpoints. It requires few frames for a solution, and supports cameras with different calibration parameters. Since a solution can be obtained with just a few frames, the matrix factorization solution can be used to initialize a sequential method such as the Kalman filter.

Once the set of projective depths has been calculated, a matrix factorization implementation gives a reconstruction up to a projectivity. The main difficulty is the upgrading from a projective to a metric reconstruction. The first difficulty is to ensure that the reconstructed object and camera centers are on the same side of the plane at infinity in the projective reconstruction. This is enforced by means of the cheiral inequalities.

For the upgrading (or metric rectification) process, ideas from automated camera calibration were used, based on the absolute dual quadric. These ideas were first introduced by Triggs [37] and modified with an additional performance enhancing constraint, again using the cheiral inequalities mentioned above. The basic idea is that the absolute dual quadric is invariant to similarity transformations and contains the absolute conic and the plane at infinity which are both required for a metric rectification.

The fact that the camera calibration parameters are assumed unknown is of importance as it makes for a versatile system. Our system allows the user to provide any known parameters, but does not demand such information. The worst case is if a motion sequence is obtained from multiple cameras, each with a different calibration and all calibration parameters unknown. In this situation auto-calibration cannot be performed and the best possible solution is a projective reconstruction. The situation becomes easier when a single camera is used or multiple cameras with a fixed calibration (the same in all frames) are used, but all the calibration parameters still unknown. In this situation a metric reconstruction is possible using a minimum of 3 frames. Our experiments show that it typically achieves the weakest, but probably still acceptable, reconstruction.

We also experimented with situations where different combinations of parameters are assumed to be known. It is found that the more parameters that are known, the better the reconstruction accuracy. It is further found that just assuming the widely applicable zero skew $s = 0$ and unit pixel aspect ratio $r = 1$ yields reliable and accurate reconstructions and is recommended if no calibration parameters are known. In addition, the fewer parameters that are known, the more frames that are needed to estimate the metric reconstruction. This ranges from a minimum of two frames, with all parameters known, to up to to a minimum of eight frames, with parameters not fixed and only one known.

Note that under certain conditions, depending on the motion in a video sequence, a metric reconstruction is not possible. These motions are referred to as critical motions. In these conditions the final reconstruction is somewhere between projective and metric.

It was further demonstrated that the reconstruction is robust to random noise. Only when the variance exceeds $\sigma^2 = 0.001$ do we notice a significant deterioration in the accuracy. The translation values tend to be the most sensitive.

For motion segmentation, we adopted ideas from Costaira and Kanade [4] that assumed an orthographic camera model and modified them for the perspective camera model. Our experiments show that the motion segmentation process, including outlier removal, is robust for objects with independent motion.

One practical application of the batch process described here, is it to use it with a sequential method such as the Kalman filter approach to SfM, to speed up convergence. The system can also be used purely to auto-calibrate cameras or segment features into different objects each to be reconstructed separately by any other SfM system. Another important application is the reconstruction of archaeological and historical sites from photographic evidence. We provide experimental evidence for the efficacy of this technique.

9.1 Future work

This work leaves considerable room for future research:

Dense feature reconstruction

The SfM system developed in this work makes use of the LK feature tracker which is a sparse optical flow tracker. The reconstruction process could just as well be applied to a dense optical flow tracker that reconstructs all of the pixels for each frame in a sequence. This would result in greatly increased detail for successful reconstructions. Note that the uncertainty in the input data would increase with a dense feature reconstruction, since feature correspondences would become more unreliable.

Segmenting dependent motion

The current motion segmentation assumes independent motion; objects with dependent motion or overlapping motion subspaces will only confuse it. A modification to allow the accurate segmentation of dependent motion, and a method of knowing their dependencies, would allow for the reconstruction and assembling of articulated objects. This would enable the reconstruction of more complex objects by approximating them as a single articulated object. A possible example would be the recognition of sign language performed by a human, represented as an articulated 3D model.

Reconstruct more complex motions

This work is limited to rigid objects. An extension that would allow the 3D reconstruction of more complex objects and motions, e.g. elastic objects undergoing deformation, could also be considered. This is undoubtedly a challenging problem but would have many important applications. An example is the accurate 3D reconstructing of human organs for medical examination.

Sequential algorithm

In [19] Morita and Kanade focus on developing the matrix factorization batch programming method into a sequential algorithm. The original Tomasi and

Kanade algorithm is robust and accurate but is difficult to apply to real-time applications and is thus limited to off-line computations. The reason for this limitation is that it is based on a batch-type operation with the computational effort further dominated by the large cost of the SVD. The sequential method has results very close to the accuracy and robustness of the original method. The key to developing such a sequential method is to observe that the shape of a rigid object does not change over time. The shape space is stationary and it should therefore be possible to derive the structure for frame f from the structure of the previous frame without performing expensive computations.

In the sequential algorithm the original SVD is replaced with an updating computation for only the four dominant eigenvectors which can be performed in $\mathcal{O}\left(N^2\right)$ operations compared with the full SVD requiring $\mathcal{O}\left(FN^2\right)$ operations, where N is the number of features and F the total number of frames. This method produces estimates of shape and motion at each input frame. A covariance-like matrix is stored instead of feature positions (measurement matrix W), and so its size remains constant as the number of frames increase. This also allows the algorithm to handle infinitely large sequences.

Note that the Morita-Kanade sequential algorithm is based on an orthographic camera, which is not as complex as the perspective camera, and excludes the complex metric rectification of a projective reconstruction. This sequential algorithm, extended successfully for perspective cameras, should increase the application potential of the algorithm presented in this work.

Bibliography

[1] A. J. Azarbayejani, T. Galyean, B. Horowitz, and A. Pentland. Recursive estimation of CAD model recovery. In *the 2nd CAD-Based Vision Workshop*, pages 90–97, 1994. 2, 4, 136

[2] F. R. Bach and M. I. Jordan. Learning spectral clustering. Technical Report UCB/CSD-03-1249, UC Berkeley, 2003. 76

[3] D. Cardon, W. Fife, J. Archibald, and D. Lee. Fast 3D reconstruction for small autonomous robots. In *Industrial Electronics Society IECON 2005*, pages 373–378, 2005. 2

[4] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *International Conference on Computer Vision*, pages 1071–1076, 1995. 4, 5, 69, 72, 74, 76, 137

[5] T. de Margerie. Povray model of my head, 2008. (27 January 2008). 121

[6] O. D. Faugeras, Q. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pages 321–334, 1992. 4, 7, 46

[7] G. W. Gear. Multibody grouping from motion images. *International Journal on Computer Vision*, 29(2):133–150, 1998. 72

[8] A. Grün, F. Remondino, and L. Zhang. Photogrammetric reconstruction of the great Buddha of Bamiyan, Afghanistan. *The Photogrammetric Record*, 19(107):177–199, 2004. 2

[9] M. Han and T. Kanade. Perspective factorization methods for Euclidean reconstruction. Technical Report CMU-RI-TR-99-22, Carnegie Mellon University, 1999. 4

[10] R. I. Hartley. Cheirality invariants. In *DARPA Image Understanding Workshop*, pages 745–753, 1993. 5, 57

[11] R. I. Hartley. Euclidean reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision*, pages 237–256, 1993. 5, 57

[12] R. I. Hartley. Cheirality. *International Journal of Computer Vision*, 26(1):41–61, 1998. 80

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry*, volume 1. Cambridge University Press, College Station, Texas, second edition, 2003. 7, 24, 46, 82

[14] A. Heyden and K. Åström. Euclidean reconstruction from constant intrinsic parameters. In *International Conference on Pattern Recognition*, pages 339–343, 1996. 49

[15] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 2

[16] K. Kanatani. Motion segmentation by subspace separation and model selection. In *International Conference on Computer Vision*, pages 586–591, 2001. 74

[17] D. Kehr. Duplicate motion, then capture emotion. *New York Times*, November 2007. 2

[18] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence*, pages 674–679, 1981. 5, 113

[19] T. Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. In *ARPA Image Understanding Workshop*, pages 1177–1188, 1994. 138

[20] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001. 77

[21] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997. 4

[22] M. Pollefeys. *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences.* PhD thesis, Katholieke Universiteit Leuven, 1999. 7, 62

[23] M. Pollefeys and L. V. Gool. Some geometric insight in self-calibration and critical motion sequences. Technical Report KUL/ESAT/PSI/0001, Katholieke Universiteit Leuven, 2000. 62

[24] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 126

[25] M. Pollefeys, R. Koch, Luc, and V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *International Conference on Computer Vision*, pages 90–95, 1998. 49

[26] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67, 1994. 60

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 76

[28] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35(4):551–566, 1993. 36

[29] P. Sturm. Critical motion sequences and conjugacy of ambiguous Euclidean reconstructions. In *Scandinavian Conference on Image Analysis*, pages 439–446, 1997. 62, 65

[30] P. Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In *Image and Vision Computing: British Machine Vision Conference*, pages 63–72, 1999. 62

[31] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conference on Computer Vision*, pages 709–720, 1996. 4

[32] Y. Sugaya and K. Kanatani. Multi-stage unsupervised learning for multi-body motion segmentation. *IEICE Transactions on Information and Systems*, 87(7):1935–1942, 2004. 72

[33] C. Tomasi and T. Kanade. Factoring image sequences into shape and motion. In *International Conference on Computer Vision*, 1990. 33

[34] C. Tomasi and T. Kanade. Shape and motion without depth. In *International Conference on Computer Vision*, pages 91–95, 1990. 2, 4, 33, 136

[35] C. Tomasi and T. Kanade. Shape and motion from image streams: A factorization method part 2. detection and tracking of point features. Technical Report CMU-CS-91-105, Carnegie Mellon University, 1991. 33

[36] C. Tomasi and T. Kanade. Shape and motion from image streams: a factorization method, full report on the orthographic case. Technical Report CMU-CS-92-104, TR 92-1270, Cornell and Carnegie Mellon University, 1992. 33, 69

[37] B. Triggs. Autocalibration and the absolute quadric. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997. 4, 49, 59, 136

[38] B.-M. van Wyk. Verifying stereo vision using structure from motion. Master's thesis, Stellenbosch University, 2007. 2

[39] C. Venter. Structure from motion estimation using a nonlinear Kalman filter. Master's thesis, Stellenbosch University, 2002. 4

[40] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent; articulated; rigid; non-rigid; degenerate and non-degenerate. In *European Conference on Computer Vision*, pages 94–106, 2006. 76

Appendix A

Software overview

The supplementary information provided in this appendix act as a guide for this study as supplied on CD. The CD contains all resources including experimental sequences, SfM system developed and literature used. The directory structure of the CD is shown in Figure A.1. The CD resources are discussed in Section A.1 and a guide the SfM system provided in Section A.2.

```
CD
├─fas3d
│  ├─analysis
│  ├─data
│  ├─jhpak
│  └─results
│
├─media
│  ├─face
│  ├─Lego
│  ├─Medusa
│  ├─multiple_quasi_real_cubes
│  ├─quasi_real_cube
│  └─quasi_real_cube_bg
│
├─papers
│  ├─auto_calibration
│  ├─matrix_factorization
│  ├─misc
│  ├─motion_segmentation
│  └─quasi_affine
│
├─software
│
└─thesis
   ├─graphs
   └─images
```

Figure A.1: Supplementary CD directory structure.

A.1 Resources

This document, in electronic format, is provided in the root directory of the CD. All literature used in this study, that is available in electronic format, is provided under `papers/`. The videos and images that were used for experimentation in Chapter 8 are provided under `media/`. Any POV-Ray scripts that were used to generate images are also included there. The SfM system that was developed as an implementation of this study and used for experimentation, and all scripts used to analyze the system's results are provided (including source code) under `fas3d/`. Some of the software required for the SfM system and the compilation of this document is provided under `software/`. This document was developed with LyX, a cross platform and open source document processor. This document's source is located under `thesis/`.

A.2 Guide to the SfM system

The SfM system was developed in Python 2.5 and requires a compatible Python interpreter. The following open source Python packages are used in the system and need to be installed:

- CVXOPT 0.9 for linear programming during quasi-affine reconstruction
- Matplotlib 0.90 for the plotting of the experiment graphs
- MayaVi 1.5 for displaying the 3D reconstructed structure
- Numpy 1.0.3 for various mathematical functions
- OpenCV 1.0 for the LK feature tracker and the calculation of fundamental matrices
- PyVTK 0.4.74 for converting 3D feature positions to a format usable by MayaVi
- SciPy 0.5.2 for the COBYLA (Constrained Optimization by Linear Approximation) nonlinear optimizer used during auto-calibration and for text file input and output
- wxPython 2.8.4.0 for generating the system GUI

All of these packages are found in the Ubuntu Linux repository. Since Python and all of the listed packages are cross-platform, the SfM system can be run on multiple platforms.

The SfM system is located in the `fas3d/` directory on the supplementary CD, shown in Figure A.1. The main system modules are located in the root directory, additional supporting modules are located in the `jhpak/` subdirectory, the files of the 2D features to be reconstructed are located in the `data/` subdirectory, all the reconstruction results are stored in the `results/` subdirectory and the Python scripts that analyze the results are stored in the `analysis/` subdirectory.

A.2.1 Module descriptions

The system performs 3D reconstruction from 2D features which it accepts in a single text file. The feature file is typically located in the `data/` subdirectory and has a table structure — each column represents the trajectory of a single feature and each row represents a frame and lists the coordinates of all features for that frame.

We provide a short description for the most important Python modules of the system:

- `ReconFrontEnd.py` creates and manages the user interface for a single rigid object
- `ReconMultiFrontEnd.py` extends `ReconFrontEnd.py` to create and manage the user interface for multiple rigid objects
- `SingleBodyReconstruct.py` manages the reconstruction of features belonging to the same rigid object
- `MultiBodyReconstruct.py` extends `SingleBodyReconstruct.py` to manage the reconstruction of multiple rigid objects
- `getfeat_text.py` retrieves the 2D features from file
- `MotionSegmentation.py` segments the 2D features into different motions (rigid objects)
- `./jhpak/NormalizedCuts.py` performs the graph cutting tasks related to the spectral clustering algorithm for motion segmentation
- `ProjectiveReconstrut.py` does a projective reconstruction of a single rigid object
- `MatrixFact.py` performs tasks related to the matrix factorization algorithm, e.g. calculating projective depths
- `QuasiAffineReconstruct.py` upgrades a projective reconstruction to a quasi-affine reconstruction

- `AutoCalibration.py` rectifies a projective or quasi-affine reconstruction to a metric or affine reconstruction through auto-calibration
- `./jhpak/jhmayavi.py` uses MayaVi and VTK to generate 3D figures of the reconstructed structure

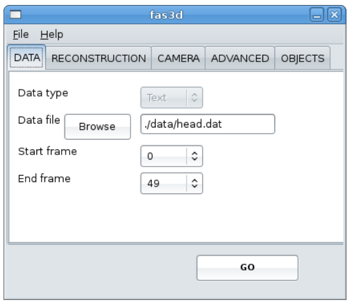
Following the reconstruction of the features assigned to rigid object i, all of the estimated results (if desired) are written to a text files:

- `./results/Featresultsi` contains the segmented features
- `./results/Sresultsi` contains the reconstructed structure
- `./results/Presultsi` contains the projection matrix for each of the F frames, the projection matrix for frame f stores its reconstructed motion and camera calibration, $P^f = K^f \left[R^f \,|\, \mathbf{t}^f\right]$
- `./results/K1resultsi` contains the first (reference) camera calibration matrix K^1 that is required to calculate each of the succeeding calibration matrices K^f

A.2.2 Script description

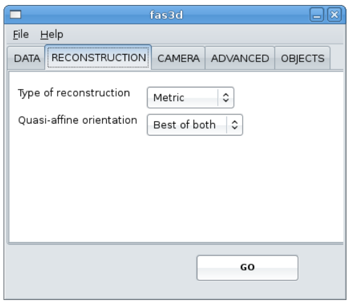
Additional scripts are provided for functionality relating to the SfM system and to perform analysis of the system's results. We provide a short description of each:

- `Tracker.py` tracks features across a sequence of images
- `Video2Imgs.py` converts a video to a sequence of images
- `./analysis/Experiments.py` creates pure and noisy cube data sets
- `./analysis/Analysis.py` analyzes and plots the reconstruction results for a synthetic cube sequence
- `./analysis/Analysis_quasi.py` analyzes and plots the reconstruction results when the ground truth is known
- `./analysis/Analysis_real.py` analyzes and plots reconstruction results when no ground truth is known
- `./analysis/MetaAnalysis.py` performs multiple reconstructions and reconstruction result analysis
- `./analysis/MotsegAnalysis.py` analyzes and plots motion segmentation results



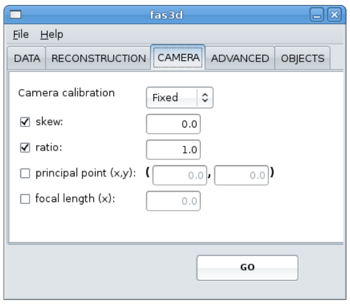
Data type: Text
Data file: ./data/head.dat
Start frame: 0
End frame: 49

(i) Data tab



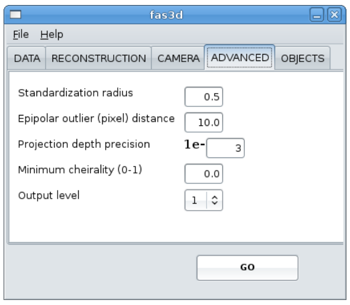
Type of reconstruction: Metric
Quasi-affine orientation: Best of both

(ii) Reconstruction tab



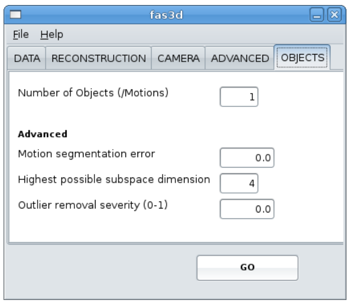
Camera calibration: Fixed
☑ skew: 0.0
☑ ratio: 1.0
☐ principal point (x,y): (0.0 , 0.0)
☐ focal length (x): 0.0

(iii) Camera tab



Standardization radius: 0.5
Epipolar outlier (pixel) distance: 10.0
Projection depth precision: 1e- 3
Minimum cheirality (0-1): 0.0
Output level: 1

(iv) Advanced tab



Number of Objects (/Motions): 1

Advanced

Motion segmentation error: 0.0
Highest possible subspace dimension: 4
Outlier removal severity (0-1): 0.0

(v) Objects tab

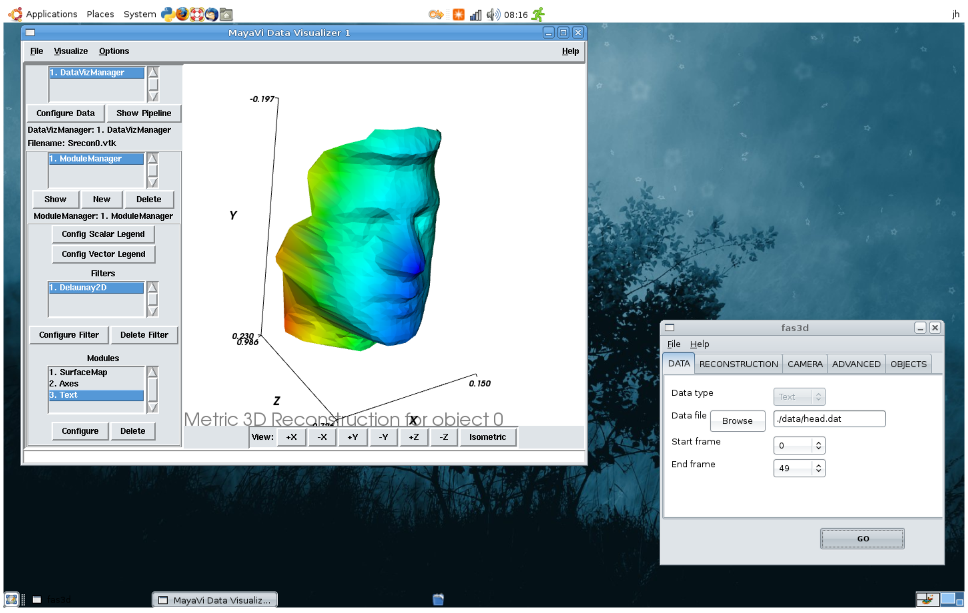
Figure A.2: The SfM GUI with tabs.

A.2.3 Graphical user interface (GUI)

The SfM system comes with its own GUI for user-friendly operation. The GUI is named fas3d (factorize auto-calibrate segment 3D) pronounced "fazed". It was developed using wxPython and is initialized by running `fas3d` (`fas3d.bat` or `fas3d.pyw` under Windows). The GUI consists of a number of tabs, each tab contains user definable options that influence the reconstruction process and results. The GUI and its different tabs are displayed in Figure A.2. Note that all of the GUI images in this section were taken of the GUI run under Ubuntu 7.10 and appearance may differ under other operating systems.

To attempt a reconstruction — we start in the "DATA" tab, Figure A.2 (i), and select which text file, containing the 2D features for all of the frames, should be reconstructed. There is also the option to select which subset of frames are to be considered in the reconstruction process. We proceed to the "RECONSTRUCTION" tab, Figure A.2 (ii), and select the type of reconstruction from the list (in decreasing quality): metric, affine, quasi-affine, projective and 2D. The orientation of the quasi-affine solution to be considered is also selected, the options are either positive, negative or the recommended option of selecting the best of both automatically. In the "CAMERA" tab, Figure A.2 (iii), one can choose whether the camera observing the scene has a fixed or variable calibration and which calibration parameters, if any, are known. The "ADVANCED" tab, Figure A.2 (iv), contains options for advanced users. It makes it possible to change the image standardization size, the minimum distance of a feature from the epipolar line before it is considered an outlier, the desired projection depth precision, the minimum cheirality of the final solution and the output verbosity during reconstruction. Finally, the "OBJECTS" tab, Figure A.2 (v), is where the observed number of independent motions or rigid objects are indicated and it contains further advanced options relating to motion segmentation. It provides options for selecting a motion segmentation error, the dimension of the motion subspaces and the severity of outlier removal. A GUI configuration can be saved to disk on the menu bar, File → Save configuration file, and later loaded, File → Open configuration file.

When the desired feature file is chosen and the options changed to the desired values — click on the "GO" button to begin reconstruction. When 3D reconstruction is completed, all of the calculated motion segmentation, structure, motion and camera calibration values are written to file and the 3D structure is displayed using MayaVi. The GUI displaying the 3D structure, following the reconstruction of the face sequence from Section 8.2, is given in Figure A.3.



1. DataVizManager

DataVizManager: 1. DataVizManager
Filename: Srecon0.vtk

1. ModuleManager

ModuleManager: 1. ModuleManager

1. Delaunay2D

1. SurfaceMap
2. Axes
3. Text

Metric 3D Reconstruction for object 0

Data type	Text
Data file	./data/head.dat
Start frame	0
End frame	49

Figure A.3: GUI displaying the reconstructed 3D structure fitted with a flat-shaded surface.