Preface

This thesis addresses the problem of generating images of an object or a scene from arbitrary viewpoints using information from a fixed number of images. This problem is referred to as view synthesis or image based rendering. In particular we focus on using features other than points.

The work has been done within a Swedish national program for visual information technology, VISIT, funded by the Swedish Foundations for Strategic Research (SSF). The overall goal for the VISIT program is to “make visual information technology visible through new technologies, new scientific cooperations, new PhDs, new products and new companies”. This includes to generate internationally strong research activities but also to improve cooperations between university and industry, and to prepare the graduate students for work outside the university field.

The program includes 8 projects. This thesis is a part of the view synthesis project. The objective for this project is to develop methods for computing the image of a 3D scene from a novel viewpoint. This can be achieved by combining information extracted from images taken from other viewpoints.

At the start the work was focused on piecewise planar scenes, e.g. city scenes, in accordance with the project plan. The geometry for planar surfaces was investigate in [BJ1,BJ2,BJ8]. This work initiated a collaboration with School of Architecture at KTH and the company Mandator Interactive, in which new computer based visualisation tools for architects were developed [BJ9]. After the licentiate degree, [BJ3], the work was continued to more general scenes, [BJ4]. In a project in collaboration with Electrolux, a system for determination and visualisation of fridge contents was developed, [BJ10]. During a visit at the speech, vision and robotics group at Cambridge University, England, recognition and pose estimation problems were investigated, cf. [BJ5,BJ6]. A common theme for the thesis is the use of more complex features than e.g. points. In [BJ7] minimal structure and motion problems were investigated for new type of features, called quivers.

The contents of the thesis is based mainly on the following papers:

Main papers


Subsidiary papers


Acknowledgements

First of all I would like to thank my supervisors Kalle Åström and Gunnar Sparr for the encourage and the help they have provided, and for careful reading of the thesis. I have appreciated the discussions and company from all members of the Mathematical imaging group. I am also grateful to the members if department for creating a pleasant atmosphere. Among colleagues I especially would like to thank Magnus Oskarsson and Fredrik Kahl for rewarding cooperations and for valuable comments on the thesis.

Financial support from the SSF-programme Visual Information Technology (VISIT) is gratefully acknowledged. I also thank the SSF-programme Nätverk för Tillämpad Matematik (NTM) for making it possible for me to work with the company Mandator Interactive for a short period. In addition, the company Electrolux has through financial support made it possible for me to research on future vision on intelligent homes, thank you. I also had a chance to work with Roberto Cipolla, Cambridge University. This was a great experience and I am deeply grateful for this.

Finally, Susanna it means a lot that you have motivated me along the way. My dear family, thank you for supporting me at all times.
# Contents

1.1 Motivation ................................................. 7  
1.2 Outline .................................................. 7  
1.3 Organization and Contributions ..................... 10  

## I Geometry in Computer Vision  

2 Preliminaries ............................................ 15  
2.1 Features ................................................. 15  
2.2 Projective spaces ...................................... 18  
2.3 Camera modeling ...................................... 23  
2.4 Two-view geometry .................................... 26  
2.5 Image sequences ...................................... 31  

3 View Synthesis of Piecewise Planar Scenes .......... 35  
3.1 Geometry of planes in two images .................. 35  
3.2 Synthesizing novel images ............................ 40  
3.3 Experiments ............................................. 46  
3.4 Conclusions ............................................. 47  

4 3D Reconstruction of Piecewise Planar Scenes ....... 49  
4.1 Projective reconstruction ............................... 49  
4.2 Single view reconstruction ............................. 52  
4.3 Euclidean reconstruction .............................. 52  
4.4 Experiments ............................................. 53  
4.5 Rectangles .............................................. 55  
4.6 Conclusions ............................................. 61  

5 Structure and Motion using Quivers ................. 65  
5.1 Preliminaries ............................................ 65  
5.2 A point with one direction ............................ 67  
5.3 A point plus two directions ........................... 68  
5.4 A point with three directions ........................ 72  
5.5 Conclusions ............................................. 77
## II Computer Vision Systems

### 6 An Automatic System for View Synthesis
6.1 Introduction ................................................. 81
6.2 System approach ........................................... 83
6.3 Experiments ............................................... 86
6.4 Conclusions ............................................... 90

### 7 Visualization of Fridge Contents
7.1 Introduction ............................................... 93
7.2 System Approach .......................................... 94
7.3 Examples ................................................ 101
7.4 Conclusions ............................................... 101

### 8 Detecting Windows in City Scenes
8.1 Introduction ............................................... 103
8.2 The detection system ..................................... 103
8.3 Evaluation ............................................... 105
8.4 Applications .............................................. 107
8.5 Conclusions ............................................... 109

### 9 A System for Automatic Pose Estimation
9.1 Introduction ............................................... 111
9.2 Preliminaries ............................................. 112
9.3 System approach - 2D model .......................... 113
9.4 System approach - 3D Model .......................... 120
9.5 Conclusions ............................................... 124

### 10 Conclusions

### A
A.1 Support Vector Machines ................................. 129
Introduction

1.1 Motivation

In the rapidly developing field of information technology the need for visualization of 3D objects and scenes is ever increasing. For example, an Internet salesman would like to show his merchandise to his costumers, or an architect would like to show how a new building would appear in an existing environment. Visualizations like these are traditionally based on a number of images, photos or drawings, but recently we have seen interactive visualizations using manually made 3D computer models. The idea is that the user should have the ability to choose the point of view himself and thereby get a better experience of the object. A limitation of the existing methods is that the quality in the computer generated images often is not satisfying. Modeling a complex object in 3D is also very time consuming.

In order to enhance realism and geometric correctness in an interactive visualization, there is a desire to use information from real images of the object or the scene. If one could develop an automatic system for doing this, there would also be a number of exciting real time applications, e.g. using only a finite set of camera views of a football game you are able to choose your own view of the game.

In many recent film productions there are special effects based on computer graphics. In order to render computer graphics in live footage the camera motion has to be estimated. This may be achieved by equipping the camera with sensors, but a cheaper way is to use the information in the images to calculate the motion of the camera.

There are also applications to image compression. Instead of transmitting a large sequence of images, only a few images or a 3D model together with information about the view points are transmitted. The large sequence of images may then be reconstructed by the receiver.

1.2 Outline

We discuss the words in the title of the thesis and how these are related to this work.

Computer vision

Computer vision aims at automatically obtaining information about the world by interpreting images by means of a computer. A central problem for the thesis in the computer
vision area is the **structure and motion** problem. From image information we wish to determine the structure of the scene and the motion of the camera. This is often achieved by identifying corresponding features in the images. These impose constraints on the geometry of the cameras and the scene. If sufficiently many correspondences of features are available, the relative structure and motion may be determined. In this thesis we present new algorithms for reconstruction of piecewise planar scenes.

The situation in which the data from the corresponding image features exactly constrains the geometry is called a **minimal case**. Minimal cases are of both theoretical and practical interest. In e.g. RANSAC, a technique to find outliers among corresponding features, they are of great importance. Three previously unsolved minimal cases of a particular feature which will be described below, are solved.

When a 3D model of the scene has been created, it may be used to render new images from different view points. For this application the 3D structure of the scene is not necessarily relevant, but only how the scene would appear from another view. The problem to generate new views of an object using a number of images is called **view synthesis** or **image based rendering**. We present techniques for view synthesis that do not rely on 3D reconstructions.

For certain applications the 3D structure of the scene may be available. The problem to determine the camera parameters when the structure of the scene is known is called **pose estimation**. An automatic system for positioning the camera in a city scene using models of the surrounding buildings is presented.

A difficult task in the previous application is to determine which building we are viewing. **Object recognition** is an area that has been a focus of research for many years. In this thesis we present a system for finding windows in city scenes. Such a system may be used in several computer vision applications.

**Rich Features**

A common way to determine the relations between the cameras and the objects in the scene is to identify corresponding features in the image. Traditionally, point and line features have been used. The different features used in this thesis contain more information than these traditional ones. We call them 'rich' features and here give a short motivation for their use.

When there mainly are buildings in a scene, the use of **planes** instead of points has proven to be useful. A wall is naturally modeled by a plane, making it possible to incorporate constraints such as a walls being parallel or orthogonal to each other. Such constraints provide valuable information about the geometry. It is very annoying from a visual point of view if a wall that should be planar is not planar in the 3D model, or if planes that should be parallel or orthogonal are not. These constraints are easy to impose using planes as basic features.

If metric information about planes is available, this should also be used in the recon-
construction process. Rectangular shapes are common in many applications, in particular when viewing buildings or indoor scenes. Features such as walls, windows and doors define rectangles which impose additional constraints on the reconstruction.

Another feature that is common in images of buildings is a point and a number of curves meeting in the point. Considering their tangential directions, we call the feature defined by a point and $n$ directions an $n$-quiver. This may be thought of as a vertex of a polyhedron and the edges meeting in the vertex. A quiver contains more information than a point. Consequently we need fewer correspondences of quivers in order to reconstruct the cameras. This is a desirable feature for certain applications, e.g. RANSAC.

There are two common techniques to find suitable point correspondences in the images for a point based solution. The first one analyses the gradient of the intensity in a neighborhood of the point to decide if the point is suitable, but the information in the gradient is not used in the analysis of the geometry. However, the concept of quivers makes it possible to use also the gradient information. We use a quiver based on the point position and a number of directions which are derived from the gradient information. The second approach to find point correspondences is to determine the intersection of two lines. This approach localizes the point very accurately, since lines may be estimated with great precision. We propose that the line and point information should be combined into a quiver, instead of using only the point information.

One application in the thesis uses only one single feature for the reconstruction. This feature is a template indicating where the edges are located on a building. If the template has been matched to the image, the position and orientation of a calibrated camera may be computed, assuming that the location of the template is known in 3D.

If certain constraints are imposed on the objects in the images, the silhouette of an object contains much information about the 3D structure. In the thesis, we also make reconstructions of objects by using silhouettes and footprints of an object.

**Geometry**

The first part of the thesis is focused on the geometry of multiple views. The geometry of planar scenes is investigated. A plane in the scene induces a homography between images. The generalized eigenvectors and eigenvalues for two such homographies are given a geometric interpretation. This interpretation may be used to generate novel images without reconstructing the scene. They may also be used to generate 3D reconstructions and we show that if it is possible to determine the intersection lines between the planes, one uncalibrated image is sufficient to create a 3D reconstruction. It is well known that when uncalibrated images are used, the reconstruction may be determined only up to a projective reconstruction.

If a rectangle can be localized in a plane, the metric information may be used to upgrade the projective reconstruction. We note that the height-to-width ratio of a rectangle may be computed from one calibrated image.
The constraints imposed by quivers are also investigated. Among others, three minimal cases with quivers are solved. In addition to this we investigate how measurement errors affect the accuracy of reconstructions based on correspondences of quivers, points and lines. Simulations show that when the same amount of information is available for points lines and quivers, the use of quivers minimizes the reprojection error if many features are available. If only a few features are available, the point reconstruction has the smallest reprojection error.

Systems

In this part of the thesis two different systems to generate novel images of a scene are presented together with a system for pose estimation.

The first system is an automatic system for view synthesis of general scenes. It is based on some new ideas. The system is validated experimentally and compared to existing systems.

The purpose of the second system is to visualize fridge contents. It has been implemented in a fridge for demonstrations. If certain restrictions are imposed on the objects and how these are placed in the fridge we show that complete 3D models may be obtained automatically using simple model based reconstruction algorithms based on the footprint and the silhouette of the object. The footprints may also be used to keep track of the objects, when a user moves or removes objects in the fridge.

Thirdly, a system for automatic pose estimation in city scenes is presented. The objective is to determine where the camera is located when models of the surrounding buildings are available. A technique based on matching a template for the current building to the image is introduced. The template indicates where the edges are located on the building. We show how a template consisting of edges in several parallel planes may efficiently be matched to the image.

The edges in the templates often arise from windows. A system for finding windows in city scenes based on support vector machines has been developed. Such a system may be of great use in 3D reconstruction algorithms as well, e.g. for model based reconstruction or when finding point correspondences in widely separated views.

1.3 Organization and Contributions

The thesis is divided in two parts. The first part, Chapter 2 to Chapter 5, deals with the geometry of multiple views. In the second part, Chapter 6 to Chapter 9, we present computer vision systems. The order in which the authors appear in the papers referred to above, corresponds to the contribution of each author.

In Chapter 2 some basic concepts and results from the computer vision area are presented. The perspective camera model is described and the geometry of multiple views is inves-
tigated. The chapter also contains an introduction to projective geometry, a powerful mathematical tool and language in computer vision.

In Chapter 3 we develop an algorithm for direct view synthesis of piecewise planar objects based on some new theoretical results concerning homographies. Some experimental results of the algorithm are also given. The main contributions in this chapter are the geometric interpretations of the generalized eigensystem of two homographies, and the use of these to generate new images. It is surprising that view synthesis is possible without calibrating the camera. The results on occlusions and visibility are also particularly interesting.

Chapter 4 is a continuation of Chapter 3. It is shown how to create a 3D model of a piecewise planar object. The common situation in which the planar patches are rectangular, is studied. The fact that 3D reconstructions may be created using only a single image and some manual interaction is astonishing.

In Chapter 5 a new feature, a quiver, for structure and motion estimations is introduced. The solutions of three minimal case are important contributions. In addition to this, simulations showing the sensitivity to noise for reconstructions based on different types of features, are interesting. This work was performed in collaboration with Oskarsson. The authors contributed equally to the ideas, implementations, experiments and the writing.

In Chapter 6 we describe a new computer vision system for automatic view synthesis of an arbitrary object. The advantages of this system compared to existing are discussed. The ideas in this chapter originates from me and Kahl. The authors contributed equally in implementations, experiments and the writing.

In Chapter 7 we present another approach to the view synthesis problem, model based reconstruction. A computer vision system for determining the contents in a fridge is described. The system has been implemented for demonstrations. The main contributions in this chapter are the model based approach to reconstruction, and the fully automated system. The ideas for this system approach were developed by me. The final implementation of the algorithms was performed by Färnström.

Chapter 8 is a continuation of the ideas in the previous chapter and it contains an algorithm for detecting windows in images. Support vector machines has been used successfully for object recognition of other features, e.g. faces and pedestrians. Here it is shown to perform well on windows. This work was performed together with Kahl. The authors contributed equally to the ideas, implementations, experiments and the writing.

In Chapter 9 we use a model of the surroundings to estimate the position and orientation
of the camera. The general approach and in particular the matching procedure are new. It is shown that the matching may be performed efficiently using the FFT, for a template consisting of a number of parallel planes. The ideas in this chapter were derived by me, and the implementations, experiments and the writings were performed by me.

Chapter 10 gives a brief summary and conclusion of the thesis.
Part I

Geometry in Computer Vision
Chapter 2

Preliminaries

In this chapter some basic concepts and some theoretic results from the area of computer vision are presented. Among others, the perspective camera model and the geometry of points in several views are described. An introduction to projective geometry, a powerful mathematical tool in computer vision, is also given.

2.1 Features

The input to a computer vision system usually consists of a number of images. These contain huge amounts of data. One method to reduce data is to consider a number of features in the images and use only the information in these. Traditionally, features such as points and lines have been used, cf. [20]. Lately there has also been an interest in using more complex features such as curves and surfaces, cf. [9, 10, 1, 3]. In the first part of the thesis, we focus on planes as basic feature. We also discuss more complex features, e.g. rectangles and so called quivers.

2.1.1 Representation of features

As a motivation and introduction to projective geometry we start by discussing basic geometry of points and lines. A more formal treatment is given in Section 2.2

Suppose we have introduced a coordinate system in an image. An image point can then be assigned Cartesian coordinates \((X, Y)\). A line in 2D consists of all points satisfying the equation,

\[ aX + bY + c = 0. \]

Lines can thus be represented by the three coefficients \((a, b, c)\). However, the correspondence is not one-to-one, since \((ka, kb, kc)\) represents the same line for \(k \neq 0\). The equivalence classes in \(\mathbb{R}^3 - [0 0 0]^T\) under this relation are known as homogeneous vectors.

A point \((X, Y)\) lies on the line \((a, b, c)\) if and only if \(aX + bY + c = 0\). This may be written in terms of an inner product as \([X \ Y \ 1][a \ b \ c]^T = 0\). It is natural to represent the point by the vector \([X \ Y \ 1]^T\). Note that for any \(k \neq 0\) we have that \([kX \ kY \ k]^T[a \ b \ c]^T = 0\) if and only if \([X \ Y \ 1][a \ b \ c]^T = 0\). We consider each vector \([kX \ kY \ k]^T\) for \(k \neq 0\) to represent the same point \((X, Y)\) in \(\mathbb{R}^2\).

Given two lines \(l_1\) and \(l_2\) in the plane (homogeneous coordinates), the intersection between them is determined by \(x = l_1 \times l_2\), where \(\times\) is the cross product. This follows
from \( \mathbf{1}_1 \cdot (\mathbf{1}_1 \times \mathbf{1}_2) = \mathbf{1}_2 \cdot (\mathbf{1}_1 \times \mathbf{1}_2) = 0 \). In a similar way it can be shown that the line defined by two points \( \mathbf{x}_1 \) and \( \mathbf{x}_2 \) is given by \( 1 = \mathbf{x}_1 \times \mathbf{x}_2 \).

Consider the two parallel lines \([a \ b \ c]^T\) and \([a \ b \ c']^T\). The intersection point of these can be determined by the equation above to be \([b - a 0]^T\). However, points with homogeneous coordinates \([x \ y \ 0]^T\) do not correspond to any point in \( \mathbb{R}^2 \). However, \( \mathbb{R}^2 \) may be augmented by adding points with last coordinate zero, cf. Section 2.2.

A particularly interesting curve in computer vision applications is the conic. A conic is a plane section of a circular cone and it may be represented as a second order polynomial in Cartesian coordinates, \(aX^2 + bY^2 + 2cXY + 2dX + 2eY + f = 0\). In homogeneous coordinates this can be written as \(\mathbf{x}^T C \mathbf{x} = 0\), where \(C\) is a symmetric \(3 \times 3\) matrix, defined up to scale,

\[
C = \begin{bmatrix}
a & c & d \\
c & b & e \\
d & e & f
\end{bmatrix}.
\]

The conic has five degrees of freedom.

In 3D space we represent a point by a homogeneous 4-vector \(\mathbf{x} = [x \ y \ z \ w]^T\). The linear equation \(\pi^T \mathbf{x} = 0\) is the equation of a plane. The feature in 3D defined by all points satisfying the equation \(\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0\), where \(\mathbf{Q}\) is a symmetric \(4 \times 4\) matrix, is called a quadric.

### 2.1.2 Linear transformations

In order to get depth information of an object, we must in general have images of the object from at least two views. Clearly there has to be a motion involved, either of the camera or the object. A rigid motion is characterized by a translation and a rotation.

The new Cartesian coordinates of a point undergoing a rigid motion are given by

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix} = R \begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + \mathbf{t},
\]

where \(R\) is an orthogonal matrix and \(\mathbf{t}\) is a translation. In homogeneous coordinates this can be written as a linear transformation,

\[
\mathbf{x}' = T \mathbf{x} = \begin{bmatrix}
R & \mathbf{t} \\
0 & 1
\end{bmatrix} \mathbf{x},
\]

where \(\mathbf{x} = [X \ Y \ Z \ 1]^T\) and \(\mathbf{x}' = [X' \ Y' \ Z' \ 1]^T\). It is easy to see that these transformations form a group under matrix multiplications, the Euclidean transformation group. The Euclidean distance, areas and angles are examples of quantities that are preserved under the Euclidean transformation group. Such quantities are called Euclidean invariants.
2.1. FEATURES

Figure 2.1: Illustration of Euclidean, affine and projective transformations.

If $T$ instead is constrained to have the form

$$T = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix},$$

with an arbitrary non-singular matrix $A$, we get the **affine transformation group**. These transformations preserve e.g. parallel lines, ratios of distances along a line and ratios of areas in a plane.

Allowing any non-singular matrix $T$, we get the **projective transformation group**. For this group, lines are transformed into lines. There is also a projective invariant for four collinear points, the **cross ratio**. It is often defined as $\{A, B, C, D\} = \frac{AD}{BC}$, but by permutations, six different invariant ratios can be formed.

Since $T$ is defined only up to a scale the projective transformation group has 8 and 15 degrees of freedom in 2D and 3D, respectively. The different transformations in 2D are illustrated in Figure 2.1. These concepts are treated more formally in Section 2.2.

The representation of features other than points will also change under a linear transformation. If $\mathbf{x}' = T\mathbf{x}$, then it is seen from the equation $1^T T^{-1} T \mathbf{x} = 1^T \mathbf{x}' = 0$ that the representation of the line is $1' = T^{-1} 1$. In the same way it can be seen that the representation of a conic is $C' = T^{-T} CT^{-1}$. In 3D the representation of a plane is $\pi' = T^{-T} \pi$ and the representation of a quadric $Q' = T^{-T} QT^{-1}$, when the homogeneous point coordinates are changed by the transformation matrix $T$. 

17
2.2 Projective spaces

Based on the concepts introduced in Section 2.1, projective spaces, in particular the projective line, plane and space are defined. For a thorough treatment, see e.g. [59]. From now on all scalars are supposed to be real, if not stated otherwise.

Definition 2.2.1. Let $V$ be a vector space of dimension $n+1$. The set of one dimensional subspaces of $V$ is called the projective space of dimension $n$, denoted $\mathbb{P}^n$.

A point in $\mathbb{P}^n$ is represented by a vector $x = [x_1 \ldots x_{n+1}]^T$ with at least one $x_i \neq 0$. Each such representation $x$ is called the homogeneous coordinates of the point. Two vectors $x$ and $y$ represent the same point if and only if $\lambda x = y$ for some $\lambda \neq 0$.

For a fixed $u = [u_1 \ldots u_{n+1}]^T \in \mathbb{P}^n$ the equation

$$u^T x = x^T u = 0$$

determines a hyperplane, or simply a plane. Note that, formally both points and hyperplanes are represented by vectors in $\mathbb{P}^n$. This is called the principle of duality. To each statement about points in $\mathbb{P}^n$ corresponds a dual statement about hyperplanes.

Definition 2.2.2. A nonsingular $(n+1) \times (n+1)$ matrix, $A$, defines a linear transformation, or a collineation, $\mathbb{P}^n \ni x \mapsto y \in \mathbb{P}^n$, according to

$$\lambda y = Ax, \quad \lambda \neq 0.$$ 

Note that the matrix associated with a given collineation is defined up to a non-zero scale factor. It can be shown that, cf. [59], an invertible mapping $\mathbb{P}^n \mapsto \mathbb{P}^n$ is a collineation if and only if collinear points are mapped onto collinear points. Other denominations for collineation are projectivity, a projective transformation or a homography.

Definition 2.2.3. A projective basis in $\mathbb{P}^n$ is a set of $n+2$ points such that for no $n+1$ of them, the corresponding representatives in the vector space $V$ are linearly dependent.

For example, the set $e_i = [0 \ldots 1 \ldots 0]^T$, $i = 1, \ldots, n+1$, where 1 is in the $i$th position and $e_{n+2} = [1 1 \ldots 1]^T$, is a projective basis for $\mathbb{P}^n$, called the standard projective basis.

Theorem 2.2.1. Given a projective basis, $(x_i)_{i=1}^{n+2}$ in $\mathbb{P}^n$, there exists a collineation, $A$, that maps this basis onto the standard projective basis, i.e.

$$\lambda_i e_i = Ax_i.$$ 

Moreover, $A$ is uniquely determined up to a scale factor.
This theorem guarantees a one-to-one correspondence between geometric primitives and their representations, cf. [59]. We will also need projective transformations between spaces of different dimensions.

**Definition 2.2.4.** A projection from \( \mathbb{P}^m \) to \( \mathbb{P}^n \) is a surjective transformation that can be written

\[
\lambda y = Ax \quad \text{for some } \lambda \neq 0,
\]

where \( A \) denotes a \((n+1) \times (m+1)\) matrix.

Note that from the surjectivity follows that \( A \) has full rank.

**Definition 2.2.5.** The subspace

\[ A^n = \{ [x_1 \ldots x_{n+1}]^T \in \mathbb{P}^n \mid x_{n+1} \neq 0 \} \]

in \( \mathbb{P}^n \) is called the affine part of \( \mathbb{P}^n \).

Hence, we get an affine part of a projective space by singling out a hyperplane \( (x_{n+1} = 0) \), called the plane at infinity, it contains the points at infinity. The subgroup of collineations preserving the plane at infinity is called the affine transformation group. It is easy to see that this condition is fulfilled if and only if the transformation has the form

\[
\begin{bmatrix}
A & b \\
0 & c
\end{bmatrix},
\]

where \( A \) is a non-singular \( n \times n \) matrix, \( b \) denotes a \( n \) vector and \( c \) is a non-zero scalar.

In the affine space we introduce some affine concepts. Two planes are said to be parallel if they intersect in the plane at infinity. The point with homogeneous coordinates \([x_1 \ldots x_n x_{n+1}]^T\) for which \( x_{n+1} \neq 0 \), is identified with the point \((X_1 \ldots X_n) = (x_1/x_{n+1} \ldots x_n/x_{n+1})^T\) in the \( n \)-dimensional affine space.

### 2.2.1 The projective line

The simplest projective space is \( \mathbb{P}^1 \), the projective line. Here a point is defined by the pair \( \mathbf{x} = [x_1 \ x_2]^T \), where at least one component is different from zero. An important concept in \( \mathbb{P}^1 \) is the cross ratio.

Let \( \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3 \) and \( \mathbf{x}^4 \) be four points in \( \mathbb{P}^1 \). The cross ratio \( \{ \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4 \} \) is defined by

\[
\{ \mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4 \} = \frac{X^1 - X^3}{X^1 - X^4} \cdot \frac{X^2 - X^3}{X^2 - X^4},
\]

where \( X^i = x^i_1 / x^i_2 \). The cross ratio is invariant under the group of collineations, thus also independent of the choice of projective coordinates. By changing the order of the points in (2.1) five other cross ratio are obtained, all invariants under projective transformations.
2.2.2 The projective plane

The space $\mathbb{P}^2$ is called the \textbf{projective plane}. A point in $\mathbb{P}^2$ is defined by a homogeneous vector $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$, where at least one component is different from zero. A line in $\mathbb{P}^2$ is also defined by a triple $\mathbf{u} = [u_1 \ u_2 \ u_3]^T$, where at least one component is different from zero. The equation of the line is

$$\mathbf{u}^T \mathbf{x} = 0.$$ 

Thus, formally there is no difference between points and lines in $\mathbb{P}^2$. For all properties of points in $\mathbb{P}^2$, there is a dual property of lines.

The duality makes it possible to extend the notion of cross ratio to four lines of $\mathbb{P}^2$ intersecting at a point, cf. Figure 2.2. The cross ratio can be defined by introducing a new line, intersecting the four lines. The cross ratio $\{P_1, P_2, P_3, P_4\}$ then is independent of the choice of the fifth line, and we define the cross ratio for lines

$$\{l_1, l_2, l_3, l_4\} = \{P_1, P_2, P_3, P_4\}.$$ 

Another interesting feature in the projective plane is the \textbf{conic}. It is represented by a symmetric $3 \times 3$ matrix, $C$, defined up to scale, and may be thought of as the locus of points in $\mathbb{P}^2$ satisfying the equation

$$\mathbf{x}^T C \mathbf{x} = 0.$$ 

The line $l_{\infty}$ with equation $x_3 = 0$, is called the \textbf{line at infinity}. The \textbf{affine plane}, $\mathbb{A}^2$, is defined as the projective plane with omission of the line at infinity. The points in $\mathbb{A}^2$ are described by the one-to-one map

$$\mathbb{A}^2 \ni (x_1, x_2) \mapsto [x_1 \ x_2 \ 1]^T \in \mathbb{P}^2 \setminus l_{\infty}.$$ 

The collineations preserving the line at infinity can be expressed as

$$\{l_1, l_2, l_3, l_4\} = \{P_1, P_2, P_3, P_4\}.$$
2.2. PROJECTIVE SPACES

\[
\begin{bmatrix}
A & b \\
0 & c
\end{bmatrix},
\]

(2.2)

where \( A \) is non-singular and \( c \neq 0 \). These collineations form the affine transformation group in \( \mathbb{R}^2 \).

The affine properties of a projectively transformed image may be recovered by identifying the line at infinity. If a transformation that takes the line at infinity to \([0 0 1]^T\) is applied to the image, it is possible to make affine measurements directly in the transformed image.

By requiring that the transformation should not only preserve the line at infinity but also two marked complex points on that line, we obtain a subgroup of the affine transformations. The two points are called the circular points and have canonical coordinates \( I = [1 \ i \ 0]^T, J = [1 \ -i \ 0]^T \), where \( i = \sqrt{-1} \). This gives two constraints on \( A \) in (2.2), i.e. \( a_{11} = a_{22} \) and \( a_{12} = -a_{21} \). In other words, \( A \) in (2.2) has the form

\[
c \begin{bmatrix}
\cos \alpha & \sin \alpha \\
-\sin \alpha & \cos \alpha
\end{bmatrix}.
\]

Such a transformation is called a similarity transformation. A transformation with \( c = 1 \) is called a Euclidean transformation.

Once the circular points have been determined Euclidean quantities may be measured. By joining the intersection of the two lines \( l_1 \) and \( l_2 \) with the circular points \( I \) and \( J \) we get a set of four lines, cf. Figure 2.3. The angle between the two original lines is then given by the Laguerre formula, cf. [59].

\[
\alpha = \frac{1}{2i} \log \{ l_1, l_2, i_a, j_a \}.
\]

Alternatively the dual conic to the circular points, \( C_\infty^* = JJ^T + JJ^T \), may be used to measure the angles. The angle \( \alpha \) between two lines, represented by the vectors \( l_1, l_2 \), is

\[
\cos(\alpha) = \frac{1_l^T C_\infty^* l_2}{\sqrt{(1_l^T C_\infty^* l_1)(1_l^T C_\infty^* l_2)}}.
\]

It is possible to measure Euclidean quantities directly in the image if the coordinate system is chosen so that the circular points have canonical positions.

2.2.3 The projective space

The space \( \mathbb{P}^3 \) is known as the projective space. Points in \( \mathbb{P}^3 \) are represented by vectors \( \mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4]^T \neq [0 \ 0 \ 0 \ 0]^T \). In the same way, a plane

\[
\{ \mathbf{x} | u^T \mathbf{x} = 0 \}
\]
is represented by the vector \( \mathbf{u} = [u_1 \ u_2 \ u_3 \ u_4]^T \neq [0 \ 0 \ 0 \ 0]^T \). Accordingly there exists a duality between points and planes in \( \mathbb{P}^3 \). A line is often represented by two points or two planes. It could also be represented by Grassman or Plücker coordinates, cf. [20].

The plane \( \mathbf{u} = [0 \ 0 \ 0 \ 1]^T \) is called the plane at infinity. The affine space \( \mathbb{A}^3 \) is defined as the projective space with omission of the plane at infinity. The coordinate representation of the points is described by the one-to-one map

\[
\mathbb{A}^3 \ni (x_1 \ x_2 \ x_3) \mapsto [x_1 \ x_2 \ x_3 \ 1]^T \in \mathbb{P}^3 \setminus \pi_\infty.
\]

The collineations preserving the plane at infinity form the affine transformation group in \( \mathbb{P}^3 \). These collineation may be expressed as

\[
A = \begin{bmatrix} C & c \\ 0 & a_{44} \end{bmatrix},
\]

where \( C \) is a nonsingular \( 3 \times 3 \) matrix, \( c \) is a 3-vector and \( a_{44} \neq 0 \). Analogously to the planar case, identifying the plane at infinity allows for affine measurements.

We obtain a subgroup of the affine transformations \( \mathbb{A}^3 \) by requiring a special complex conic to be invariant. This conic, the absolute conic, is defined as the intersection of the quadric \( \sum_{i=1}^3 x_i^2 = 0 \) with the plane at infinity. The absolute conic may be interpreted as a circle with radius \( i = \sqrt{-1} \) on the plane at infinity. An affine transformation that keeps this conic invariant is called a similarity transformation, and can be written

\[
A = \begin{bmatrix} cR & c \\ 0 & a_{44} \end{bmatrix},
\]

where \( R \) is an orthogonal \( 3 \times 3 \) matrix, \( c \) is a \( 3 \times 1 \) vector and \( a_{44} \neq 0 \). In particular, if \( c = 1 \), \( A \) is called a Euclidean transformation. The Euclidean transformations form a subgroup of the affine transformation group. Once the absolute conic has been determined Euclidean quantities may be measured. The angle between two lines with directions (3D vectors) \( \mathbf{d}_1 \) and \( \mathbf{d}_2 \) is given by
2.3. CAMERA MODELING

\[
\cos \Phi = \frac{d_1^T \Omega_\infty d_2}{\sqrt{(d_1^T \Omega_\infty d_1)(d_2^T \Omega_\infty d_2)}}.
\] 

(2.3)

Here \(d_1\) and \(d_2\) are the points of intersection of the lines with the plane \(\pi_\infty\), and \(\Omega_\infty\) is the matrix representation of the absolute conic in that plane.

2.3 Camera modeling

To be able to draw any conclusions from images we need to have a model of how an image is created. Following [20], in this section a mathematical model of a perspective camera is presented. It is shown that the tools of projective geometry are very useful when the camera and the scene is analyzed from images.

2.3.1 The camera equation

The most commonly used model of a camera is the pinhole camera. An image point by intersecting a particular plane, the image plane, and the line passing through the object point and the camera center, cf. Figure 2.4. The projection of an object point to an image point can in this case be written

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  -f & 0 & 0 & 0 \\
  0 & -f & 0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  X' \\
  Y' \\
  Z'
\end{bmatrix},
\]

where the standard coordinate system in Figure 2.4 is used. The camera constant, or focal length, \(f\) describes the location of the camera center relative to the image plane. With \(f = -1\) the projection equation becomes particularly simple, we call it a normalised coordinate system.

It is often convenient to fix a world coordinate system in the scene different from the one attached to the camera. The standard coordinate system is then given by a Euclidean transformation of the world coordinate system,

\[
X' = \begin{bmatrix}
  R' & t' \\
  0 & 1
\end{bmatrix} X.
\]

It is also common to allow some freedom in choosing the image coordinate system. We model this by an affine change of the image coordinate system, \(\lambda x = Ax'\), where \(A\) is a matrix of type (2.2). Altogether we have

\[
\lambda x = A \begin{bmatrix}
  -f & 0 & 0 & 0 \\
  0 & -f & 0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
  R' & t' \\
  0 & 1
\end{bmatrix} X = PX.
\]

23
CHAPTER 2. PRELIMINARIES

The $3 \times 4$ matrix $P$ is called the camera matrix and it is defined only up to a scale factor. If the first $3 \times 3$ block of $P$ is decomposed into an upper triangular matrix $K$ and an orthogonal matrix $R$, this can be written as

$$\lambda x = KR[I - t] X = P X.$$  \hspace{1cm} (2.4)$$

Here $t$ is a 3 vector representing the camera center and $\lambda$ is a scalar. The vectors $X$ and $x$ are the homogeneous coordinates of the object point and the image point, respectively. In total the camera is described by 11 parameters. Note the connection between the camera matrix and a projection from $\mathbb{P}^3$ to $\mathbb{P}^2$.

A somewhat simplified camera model is the affine camera. This model assumes parallel projection, the case when the camera constant is infinite. The camera matrix will then have the form

$$P_{\text{affine}} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix}.$$ $$

The affine camera is particularly useful when there are only small perspective effects in the image, i.e. if there is not very much variation in the depth relative to the camera constant. It is often used to get an approximation of the reconstruction that afterwards can be refined using the perspective camera model.

The image often has to be corrected for non-linear distortion before any of the discussed camera models can be applied. These distortions are due to imperfections in the camera lenses, cf. [28]. Many wide angle lenses have such distortion so that a straight line may appear as a curve in the image. These errors can be corrected if the distortion parameters are known.
A general problem in computer vision is, given a number of images of a scene, to determine the camera matrices and a 3D reconstruction of the object. That is, to determine \( P_i, X_j \) and \( \lambda_{ij} \), observing only the image data \( x_{ij} \), where the index \( i \) refers to the images and \( j \) refers to the points. From (2.4) it is clear that if the perspective camera model is used, it is only possible to determine the camera matrices up to an unknown projective transformation. In fact, if \( P \) and \( X \) have been found to fulfill (2.4), then also \( P' = PT \) and \( X' = T^{-1}X \), for an arbitrary invertible \( 4 \times 4 \) matrix \( T \), will do as a solution to (2.4). Similarly, the reconstruction can only be made up to an affine transformation if the affine camera model is used.

Because of this ambiguity in the reconstruction we can always find a coordinate system in which the camera matrices have the form

\[
P_1 = [I \mid 0] \quad \text{and} \quad P_i = A_i[I \mid -e_i], \quad i = 2, \ldots, N
\] (2.5)

where \( I \) is the identity matrix, \( A_i \) a non-singular matrix and \( e_i \) is the camera center for camera \( i \). The matrix \( A_i \) may be interpreted as the projective transformation that maps the plane at infinity to the image. Also note that \( e_1 \) can be interpreted as the camera center in image 1. It is called the **epipole** of camera \( i \) in image 1.

### 2.3.2 Camera calibration

The matrix \( K \) in (2.4) represents the intrinsic properties of the camera. The parameters in \( K \) have interesting geometric interpretations. Let

\[
K = \begin{bmatrix}
f & fs & x_0 \\
0 & f\gamma & y_0 \\
0 & 0 & 1
\end{bmatrix}.
\]

Then \( f \) represents the **camera constant**, i.e. the distance from the camera center of the lens to the image plane. The point \((x_0, y_0)\) is called the **principal point** and may be interpreted as the orthogonal projection of the camera center onto the image plane. The parameter \( \gamma \) denotes the **aspect ratio**, determining the ratio of the \( x \) and \( y \) magnifications of the image axes. The **skew**, \( s \), is used to model non perpendicular image axes. Altogether, the parameters in \( K \) are called the **intrinsic camera parameters**. When they are known, the camera is said to be **calibrated**. The parameters in \( R \) and \( t \) are sometimes referred to as the **extrinsic parameters**, representing the orientation and the location of the camera.

A problem that often has to be solved in computer vision applications is the problem of **camera calibration**, where the goal is to determine the intrinsic parameters. Often a calibration grid is used. If metric information about the object is available, the camera matrix \( P \) can be found up to a scale factor, and a modified QR-factorization \( P = RQ \) gives the camera parameters \( K = R \).
It is sometimes possible to use information about the camera motion to determine the intrinsic calibration. This could for instance be the case when the camera is mounted on a robot. The method to calibrate a camera on-line without 3D information is called auto-calibration. Using assumptions on \( K \), e.g. that some of the intrinsic parameters are constant in the images, it is possible to recover the calibration if the number of images is sufficient, cf. [32, 33].

Consider a point in the plane at infinity, \( X = (d^T \ 0)^T \). It is projected to the image point \( x = PX = KRd \). This means that the mapping between the plane at infinity and the image is given by \( H = KR \), cf. (2.5). The absolute conic is located on the plane at infinity and is projected into the image as

\[
\omega = (KR)^{-T}I(KR)^{-1} = (KK^T)^{-1} = K^{-T}K^{-1}.
\]

Hence knowing the calibration \( K \), the image of the absolute conic may be determined and vice versa. In order to make metric measurements in the 3D scene we need to estimate the absolute conic in \( \mathbb{P}^2 \). This is possible if also the plane at infinity can be determined or if the image of the absolute conic is known in two images. In the latter case the absolute conic in \( \mathbb{P}^2 \) is determined as the intersection of the two cones with vertices in the camera centers and generated by the respective image of the absolute conic.

### 2.4 Two-view geometry

The geometry of two images, \( I_1 \) and \( I_2 \), of a set of points is now investigated. After introduction of coordinate systems, let the camera matrices be denoted \( P_1 \) and \( P_2 \).

#### 2.4.1 Projection of planar surfaces

A plane, \( \Pi \), in 3D-space is determined by a triplet of non-collinear points, represented in homogeneous coordinates by \( X_1, X_2 \) and \( X_3 \). Every point in the plane may be expressed as a linear combination of these, \( X = aX_1 + bX_2 + cX_3 \), cf. Figure 2.5. The projection of this point onto image one is

\[
\lambda_1x_1 = P_1X = P_1\underbrace{[X_1 \ X_2 \ X_3]}_{T_1}(a \ b \ c)^T = T_1(a \ b \ c)^T,
\]  

(2.6)

where \( T_1 \) is a \( 3 \times 3 \) matrix. The norm of \( X_1 \) has to be fixed in order for \( T_1 \) to be unique. The matrix \( T_1 \) is invertible unless the focal point lies in the plane \( \Pi \). In the second image the point is projected to

\[
\lambda_2x_2 = P_2X = P_2\underbrace{[X_1 \ X_2 \ X_3]}_{T_2}(a \ b \ c)^T = T_2(a \ b \ c)^T.
\]

(2.7)
2.4. TWO-VIEW GEOMETRY

Combining (2.6) and (2.7) we get \( \mathbf{w}_2 = T_2 T_1^{-1} \mathbf{x}_1 \). Hence, an object plane induces a projective transformation between image planes. This transformation is often called a homography and may be represented by a \( 3 \times 3 \) matrix \( H \),

\[
\lambda \mathbf{x}_2 = H \mathbf{x}_1,
\]

where \( \mathbf{x}_1 \) and \( \mathbf{x}_2 \) are the homogeneous image coordinates for the point in image one and two, respectively. Note that though \( T_1 \) and \( T_2 \) depends on the choice of the points \( \mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3 \) in the plane, the homography matrix \( H \) does not. Essentially \( H \) only depends on \( P_1 \), \( P_2 \) and the plane \( \Pi \). For a particular choice of camera matrices \( P_1 \) and \( P_2 \), there is a one-to-one correspondence between planes and homographies.

Since the matrix describing a homography is defined only up to a scale factor the homographies have eight degrees of freedom. A homography matrix can be calculated linearly from four point correspondences, if no three points are collinear. Taking the cross product

\[
\mathbf{x}_2 \times H \mathbf{x}_1 = \mathbf{x}_2 \times \lambda \mathbf{x}_2 = 0,
\]

the scale factor is eliminated. Each point correspondence gives two linearly independent constraints on the elements in \( H \). There are also nonlinear methods to estimate the homographies, cf. [35, 36]. These are in general more robust to noise than the linear algorithms.

The concept of homography will play an important role in Chapter 3 and Chapter 4.

2.4.2 Epipolar constraints

The relation between corresponding points in two images is more complicated when the object points are not coplanar. In order to describe the general situation, consider Figure 2.6. The points \( f_1, f_2, \mathbf{x}_1, \mathbf{x}_2 \) all lie in the same plane, the epipolar plane. Using a normalised coordinate system and notation from Figure 2.6, this constraint can

Figure 2.5: A plane surface \( \Pi \) induces a projective transformation \( H \) between images.
be written as
\[
x_1 \cdot (f_2 - f_1) \times R x_2 = 0.
\]

Expressing the cross product by matrices, we get
\[
\mathbf{x}_1^T T_{f_2-f_1} R \mathbf{x}_2 = \mathbf{x}_1^T E \mathbf{x}_2 = 0.
\]

Here the notation
\[
T_{(a,b,c)} = \begin{bmatrix}
0 & -c & b \\
c & 0 & -a \\
-b & a & 0 \\
\end{bmatrix}
\]

is used. The matrix \( E = T_{f_2-f_1} R \) is called the **essential matrix** for the two images.

Allowing an arbitrary affine image coordinate system,
\[
\mathbf{x}_1 = K_1 \mathbf{\mathbf{x}}_1, \quad \mathbf{x}_2 = K_2 \mathbf{\mathbf{x}}_2,
\]

we get
\[
\mathbf{\mathbf{x}}_1^T K_1^T T_{f_2-f_1} R K_2 \mathbf{\mathbf{x}}_2 = \mathbf{\mathbf{x}}_1^T F \mathbf{\mathbf{x}}_2 = 0.
\]

The \( 3 \times 3 \) matrix \( F \) is called the **fundamental matrix** for the two images. The essential matrix and the fundamental matrix both represent the relative positions of the two cameras. The essential matrix takes care of the case when the cameras are calibrated, while the fundamental matrix also works when they are uncalibrated, cf. [20].

The fundamental matrix may also be derived using algebraic arguments. This approach is later used to derive the constraints from corresponding points in three and four images. The two camera equations may be combined into one equation as,
\[
\begin{bmatrix}
P_1 \\
P_2 \\
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1 \\
0 \\
\mathbf{x}_2 \\
\end{bmatrix}
\begin{bmatrix}
X \\
-\lambda_1 \\
-\lambda_2 \\
\end{bmatrix} = 0.
\]

Since the matrix \( M \) has a non-empty null-space the determinant is equal to zero. By making a Laplace expansion of the determinant in the image coordinates we get a bilinear equation,
\[
\text{det}(M) = \mathbf{x}_1^T F \mathbf{x}_2 = 0.
\]

Again, the matrix \( F \) is the fundamental matrix. The elements of the fundamental matrix thus can be expressed by means of sub-determinants of the matrix formed by two rows of camera matrix one and two rows of camera matrix two.

If \( \mathbf{1}^T = \mathbf{x}_1^T F \), then \( \mathbf{1}^T \mathbf{x}_2 = 0 \), and consequently \( \mathbf{1} \) represents a line in image two, containing \( \mathbf{x}_2 \). In other words, the fundamental matrix together with a point in image
one define a line in image two, so that the corresponding point in image two lies on
the line. This line is called the \textit{epipolar line} in image two of the point \( x_1 \) in image
one. It is clear from Figure 2.6 that all epipolar lines intersect in one point, the \textit{epipole},
introduced in section 2.3. The geometric interpretation of the epipole is the image of the
camera center of the other camera.

Since the epipole is on every epipolar line it holds that \( x_1^T F e = 0 \) for every \( x_1 \), i.e.
\( F e = 0 \). The epipoles in image one and two may consequently be calculated as the left
and right null-spaces to \( F \), respectively. This means that the fundamental matrix has in
general rank two. Since it is determined only up to a scalar factor it has seven degrees
of freedom. Each point correspondence gives one equation in the unknown elements of
\( F \). Ignoring the rank constraint, the fundamental matrix is uniquely determined by a
linear system of equations obtained from eight point correspondences. If in addition the
nonlinear rank constraint is used, only seven points are needed. However, in this case
there are three possible complex solutions.

If more than eight point correspondences are available, the fundamental matrix is
overdetermined. It may be calculated linearly by making first a singular value decom-
position of the system of equations obtained from the point correspondences. Another
possibility is to find the matrix \( F \) that minimizes the sum of distances between the epipolar
line and the corresponding point. This method is nonlinear and the minimum is
found e.g. by the Gauss-Newton method.

The camera matrices can easily be computed from the fundamental matrix and vice
versa. Using the representation in (2.5) we have,

\[
P_1 = [I \mid 0], \quad P_2 = [Q \mid -t],
\]

where \( F = Q^T T_k \) and \( T_k \) is defined in accordance with (2.9). We estimate \( t = f_2 - f_1 \)
as the right null space of \( F \). Then \( Q \) can be calculated linearly.

\subsection{2.4.3 The correspondence problem}

The previous discussion assumes that a number of point correspondences have been de-
termined beforehand. It is difficult to construct automatic methods for finding point
correspondences, in particular when the images are taken with a considerable motion of
the cameras in between, so-called \textit{wide baseline matching}. When working with a small
number of images a wide baseline is desired. This is because the estimation of \( F \) from
point matches is better conditioned for large than for a small baseline. Also the precision
of the estimated 3D structure improves with a large base line.

There are standard ways to establish a point correspondence for small baselines,
cf. [74], but the methods are far from perfect. Some of the existing methods for view
synthesis even require a dense set of point correspondences. For this purpose it is nec-
essary to have a reliable automatic system. A commonly used technique is the Random
Sample Consensus (\textbf{RANSAC}), cf. [23]. In the first step the images are processed e.g. by
CHAPTER 2. PRELIMINARIES

Figure 2.6: The epipolar constraint can be derived from the fact that $x_1$, $x_2$, $f_1$, and $f_2$ are coplanar.

**Harris corner detector** [26], to find characteristic points. These points should generate two large eigenvalues of similar size for a local autocorrelation function. This corresponds to that a patch around the point cannot be translated in any direction without a significant change in gray levels. In the second step a hypothesis matching for all points is established based on e.g. distance and correlation. Following this the fundamental matrix is estimated from a minimal set of randomly chosen features. The number of hypothesis matches that agree with the estimated fundamental matrix is counted. This procedure is repeated until a sufficient number of acceptable matches is obtained. All the inliers are then used to get a better estimate of the fundamental matrix.

In this thesis we present new features, quivers, containing a lot of information. These may be used in RANSAC to get the initial estimate of the fundamental matrix. Since only a small number of these features are needed, the algorithm is of lower complexity. As we also will show, the estimation is more robust using quivers instead of points.

For a large baseline the problem is much more difficult than for a small baseline and there are no standard methods for the correspondence problem. See e.g. [72] for a recent paper on this problem. The new features presented in this thesis may simplify the large baseline problem. Since only a few of these are present in the images, the hypothesis matching will be less difficult. Object recognition may also be used to simplify this problem, cf. Chapter 8.

In algorithms presented in this thesis, correspondences between image patches have to be established. This is a classic problem of pattern recognition, however using it for reconstruction is not as well studied as the point correspondence problem. Nevertheless there are promising results on this problem, cf. [8].
2.5 Image sequences

The structure and the motion are estimated more robustly and accurately if more than two images are used. In addition to this the correspondence problem is easier to solve using e.g. tracking methods, [74, 51]. The geometry for three and four views of a scene is now investigated. An optimization technique for an arbitrary number of images, called bundle adjustment, is also described.

2.5.1 Trifocal constraints

The fundamental matrix encodes the geometry for two cameras. The geometry for three cameras can be encoded in a tensor, the trifocal tensor, represented by a $3 \times 3 \times 3$ array, cf. [27].

Suppose three images are taken by uncalibrated cameras. For each object point, $X$, holds

\[
\begin{align*}
\lambda_1 x_1 &= P_1 X, \\
\lambda_2 x_2 &= P_2 X, \\
\lambda_3 x_3 &= P_3 X.
\end{align*}
\]

This may be written in one system of equations

\[
\begin{bmatrix}
P_1 & 0 & 0 \\
P_2 & 0 & 0 \\
P_3 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
= \begin{bmatrix}
-\lambda_1 \\
-\lambda_2 \\
-\lambda_3
\end{bmatrix}
\begin{bmatrix}
X \\
y
\end{bmatrix}
= 0.
\]

Since the $9 \times 7$ matrix $M$ has a non-empty null-space it is rank deficient. This implies that all $7 \times 7$ sub-determinants of $M$ are equal to zero. By performing a Laplacian expansions of the sub-determinants linear constraints in the image coordinates are obtained. If two rows are chosen from the same camera matrix we get the bilinear epipolar constraints, cf. (2.10). However, if two rows are chosen from different camera matrices we get trilinear constraints on the image coordinates, typically of the form

\[
c_1 x_1 x_2 x_3 + c_2 x_1 x_2 y_3 + \ldots + c_1 x_1 y_2 y_3 = 0,
\]

where the $c_i:s$ are expressed by determinants of submatrices of $M$. Using tensor notation and the Einstein summation convention all the trilinear constraints may be written as,

\[
T^{ijk} x^i_1 \varepsilon_{jj'} x^j_2 \varepsilon_{kk'} x^k_3 = 0_{j'k''},
\]

where $T$ is a $3 \times 3 \times 3$ tensor and $\varepsilon$ is the permutation symbol, i.e. $\varepsilon_{123} = \varepsilon_{231} = \varepsilon_{312} = 1$, $\varepsilon_{321} = \varepsilon_{213} = \varepsilon_{132} = -1$ and $\varepsilon_{ijk} = 0$ if two indices are equal.
There are \(3 \cdot 3 \cdot 3 = 27\) ways to omit two rows, however only four of these lead to linearly independent constraints. Hence each point gives four constraints on the tensor. To calculate the tensor we consequently need at least 7 points, \((7 \cdot 4 \ge 3 \cdot 3 \cdot 3 - 1)\). However, the minimal number of parameters needed to define the three cameras is \(3 \cdot 11 - 15 = 18\). In order for the components to build up a trifocal tensor, a number of constraints have consequently to be fulfilled. These constraints can be expressed in polynomial equations in the tensor components, cf. [31, 22]. If these nonlinear constraints are used it is possible to calculate the trifocal tensor from 6 points. There may then be up to three solutions.

Now consider a line in 3D space. The line is projected to an image line. A line in a plane may be represented in dual form in homogeneous coordinates as \(l = [a \ b \ c]^T\). The problem of determining the 3D line from its image lines \(l_i\), with known camera matrices \(P_i\), can be solved by intersecting the planes \(P_i^Tl\), \(i = 1, 2, 3\). A necessary condition for the planes to meet in a line is that

\[
\frac{1}{\det} \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\

\end{bmatrix} = 0.
\]

These constraints may in the case of three images be formulated in terms of elements of the trifocal tensors. Consider three cameras \(\{P_1, P_2, P_3\}\) and three corresponding lines \(\{l_1, l_2, l_3\}\), where \(l = [l_1 \ l_2 \ l_3]^T\). Using Einstein’s summation convention the constraints from (2.12) may be written as

\[
I^2 \sim T_i^{jk} [l_j^1 l_k^2] \iff 3 \times T_i^{jk} [l_j^1 l_k^2] = 0.
\]

Here \(\sim\) denotes equality up to scale and \(T_i^{jk}\) denotes elements in the trifocal tensor, defined as

\[
T_i^{jk} = \epsilon_{i\nu\iota\sigma} \det \begin{bmatrix}
P_i^{\nu'} \\
P_i^{\nu} \\
P_j^{\iota} \\
P_k^{\iota}
\end{bmatrix},
\]

where \(\epsilon_{ijk}\) denotes the permutation symbol.

From this it is seen that the four constraints from three corresponding points, \(p_1, p_2, p_3\) may be expressed as

\[
p_3^j T_i^{jk} [l_j^1 l_k^2] = 0,
\]

where \(l_1, l_2\) are lines passing through the corresponding point in views one and two. Different choices of these lines give in total four linearly independent constraints on the tensor.

The necessary and sufficient conditions for the trifocal tensor for projective cameras are given in e.g. [22, 21, 30] and for an affine camera in [2].
2.5.2 Quadrifocal constraints

The analysis performed in Section 2.5.1 may also be performed in the case of four images, taken by uncalibrated cameras,

\[
\begin{align*}
\lambda_1 x_1 &= P_1 X, \\
\lambda_2 x_2 &= P_2 X, \\
\lambda_3 x_3 &= P_3 X, \\
\lambda_4 x_4 &= P_4 X.
\end{align*}
\]

This may be rewritten as

\[
\begin{bmatrix}
P_1 & x_1 & 0 & 0 & 0 \\
P_2 & 0 & x_2 & 0 & 0 \\
P_3 & 0 & 0 & x_3 & 0 \\
P_4 & 0 & 0 & 0 & x_4
\end{bmatrix}
\begin{bmatrix}
X \\
-\lambda_1 \\
-\lambda_2 \\
-\lambda_3 \\
-\lambda_4
\end{bmatrix}
= M y = 0.
\]

Since the $12 \times 8$ matrix $M$ has a non-empty null-space it is rank deficient and all $8 \times 8$ sub-determinants vanish. The Laplace expansion of the sub-determinants gives the so-called quadrilinear constraints, which can be expressed using a $3 \times 3 \times 3 \times 3$ tensor called the quadrifocal tensor. Just as for the trifocal tensor, there are non-linear constraints to be fulfilled by the elements of a quadrifocal tensor. By counting the number of equations and unknowns we find that in this case, as for any larger number of cameras, we need at least 6 points to make a reconstruction.

2.5.3 Constraints from more than four images

In order to extend the discussion in the previous section to five images we have to omit two rows from one camera matrix in the calculation of the sub-determinants. This camera matrix do not affect the calculations and the constraints are the quadrilinear constraints above.

2.5.4 Bundle adjustment

In practical situations the point correspondences contain measurement errors. A reconstruction algorithm is more robust to such errors if many point correspondences are used. We then get an overdetermined system which is solved e.g. by minimizing the norm of some error function. If the previously discussed reconstruction methods are used, it is natural to minimize an algebraic expressions.

However, a more robust technique is obtained by minimizing with respect to a geometric error, the reprojected error. This procedure, called bundle adjustment, has its
origin in photogrammetry. An initial estimate to the bundle optimization is often obtained by one of the methods in the previous sections.

Consider again the camera equations

\[ \lambda_{i,j} \mathbf{x}_{i,j} = P_i \mathbf{X}_j, \quad i = 1, \ldots, M, \quad j = 1, \ldots, N \]

Assume that the point measurements \( \hat{\mathbf{x}}_{i,j} \) are given by

\[ \hat{\mathbf{x}}_{i,j} = \mathbf{x}_{i,j} + \mathbf{n}_{i,j}, \]

where each \( \mathbf{n}_{i,j} \) denotes uncorrelated normally distributed noise with zero mean. Then it is natural to consider the reprojection error. The reprojection error is the difference between the measured points and the image points obtained by reprojection of the reconstruction computed from a set of parameters \( \lambda_{i,j}, P_i \) and \( \mathbf{X}_j \). The procedure of finding the parameters that minimize this error in some norm is called bundle adjustment. The expression to minimize is accordingly,

\[ \min_{\{P_1, \ldots, P_M, \mathbf{x}_1, \ldots, \mathbf{x}_N\}} \sum_{i,j} d(P_i \mathbf{X}_j, \hat{\mathbf{x}}_{i,j}), \]

where \( d \) is a distance function.

The minimum may be found e.g. by using non-linear methods such as the Gauss-Newton method with initial values given from the linear algorithms previously discussed. Further details are found in [61].
Chapter 3

View Synthesis of Piecewise Planar Scenes

In this chapter a novel method for synthesizing new views of piecewise planar scenes is described, using images taken with an uncalibrated camera. It is a direct approach in the sense that new views are generated without making an explicit 3D reconstruction. This allows us to generate new views without calibrating the camera. In contrast to methods based on point correspondences, our approach uses planes as basic features. A fundamental fact used in the algorithms is that a patch corresponding to a planar surface in the scene, generates a homography transformation between patches in different images, as was shown in Section 2.4.1. When a number of such homographies are known, geometric relations between the corresponding planes can be determined. These are used together with the chosen camera parameters to linearly calculate the homographies for each patch in a reference image to the new image.

A 3D reconstruction may appear unrealistic for an observer if e.g. a wall which is supposed to be planar is not planar in the model. Manipulation of the model is usually needed, enforcing coplanarity. Our method uses planarity constraints in the analysis rather than on the finished model. Another advantage of working with planes, is that common properties for buildings, such as planes being parallel or orthogonal to each other may easily be incorporated into the algorithm.

First the geometry of projections of planar scenes is studied. Then an algorithm based on these results is presented. Finally some experimental validations of the ideas are given.

3.1 Geometry of planes in two images

Suppose that we have two images, $I_1$ and $I_2$, of a scene containing two planar patches $\Pi_1$ and $\Pi_2$. The homography matrices, $H_1$ and $H_2$, from $I_1$ to $I_2$ for the corresponding patches are assumed to be known, see Figure 3.1.

Consider the generalized eigenvectors and the generalized eigenvalues of the pair $(H_1, H_2)$, which by definition fulfill

$$H_1 v = \lambda H_2 v,$$

where $\lambda$ is the generalized eigenvalue and $v$ is the corresponding eigenvector. Since homographies are non-singular, $\lambda$ and $v$ can also be thought of as eigenvalues and eigenvectors of the planar homology $H_2^{-1} H_1$, cf. [59]. It will now be shown that $\lambda$ and $v$ have natural geometric interpretations.
To begin with, the generalized eigenvectors and eigenvalues for a pair of matrices describing a coordinate change of a plane in the 3D space are examined. It is shown that these matrices correspond to the homographies in a special case and that the results hold also in the general case.

Assume that an affine world coordinate system has been fixed. Consider the coordinate change that takes the 3D coordinates of all points from a coordinate system with origin in $f_2$ to another one with origin in $f_1$. It can be written as $X' = AX'' + b$, where $A$ is a 3 x 3 matrix and $b$ a 3 x 1 matrix. The following lemma characterizes the restriction to planes of such coordinate transformations.

**Lemma 3.1.1.** Let $X' = (x', y', z')^T$, $X'' = (x'', y'', z'')^T$ denote coordinate vectors with respect to two affine coordinate systems in 3D space, and let $\Pi$ be a plane not containing the origin of the second coordinate system. Then the restriction to $\Pi$ of the coordinate transformation between the two systems can be written

$$X' = A_{\Pi}X''$$

for some 3 x 3 matrix $A_{\Pi}$.

**Proof.** Let the plane $\Pi$ be given by $\pi^T X'' = d$. The constant $d \neq 0$ since $0 \not\in \Pi$. Thus $1 = \frac{\pi^T X''}{d}$ for all points on the plane. The affine transformation can now be written

$$X' = AX'' + b = AX'' + b \frac{\pi^T X''}{d} = (A + \frac{1}{d}b\pi^T)X'' = A_{\Pi}X'',$$

where $A_{\Pi}$ is a 3 x 3 matrix.

\[\square\]
Consider the case of two planes \( \Pi_1 \) and \( \Pi_2 \), and let \( A_1 \) and \( A_2 \) denote the corresponding matrices, given by Lemma 3.1.1. The generalized eigenvectors and eigenvalues of the pair \((A_1, A_2)\) will now be examined. The points on the line of intersection of the two planes obviously fulfill (3.1), as well as all multiples of coordinate vectors of such points. Hence all vectors in the plane through this line and \( f_1 \) are generalized eigenvectors with eigenvalue 1.

Next it is shown that the last eigenvector, up to a scale, is \( f = f_1 f_2 \), and that the corresponding eigenvalue is the cross ratio \( \{f_1, f_2, C_2, C_1\} \), with notations from Figure 3.2.

Since \( C_1 \in \Pi_1 \) we have

\[
A_1 f_1 C_1^\top = f_2 C_1^\top, \\
A_2 f_1 C_2^\top = f_2 C_2^\top,
\]

where the vector on the left hand side is expressed in the coordinate system \( X' \) and the one on the right hand side in the coordinate system \( X'' \). We also know that

\[
f_i C_j^\top = \frac{f_i C_j}{f_i C_i^\top},
\]

where \( \frac{f_i C_j}{f_i C_i^\top} = \frac{f_i C_1}{f_i C_i^\top} \) denotes the “ratio” of \( |f_i C_j| \) and \( |f_i C_i| \) with negative sign if and only if the two vectors are reverse. By combining these equations we get that

\[
A_1 f_1 C_1^\top = \frac{f_2 C_1}{f_2 C_2} f_2 C_2^\top, \\
A_2 f_1 C_2^\top = \frac{f_2 C_2}{f_2 C_1} f_2 C_1^\top = f_2 C_2^\top.
\]

From this and the fact that \( f = k f_1 C_1^\top \) for some \( k \), it is clear that \( f \) fulfills (3.1) as well as that the corresponding eigenvalue \( m \) indeed is the cross ratio,

\[
m = \frac{f_1 C_2}{f_2 C_2} / \frac{f_1 C_1}{f_2 C_1}.
\]

Note that the cross ratio can be interpreted in terms of the concept of kinetic depth, cf. [63].

An exceptional case that has to be considered when \( f \) belongs to the intersection line. All eigenvalues then are equal but there are only two eigenvectors. The matrix \( A_1 Q^{-1} A_1 \) is not diagonalizable in this case.
The following lemma relates the homographies to the matrices representing the change of coordinates.

Lemma 3.1.2. Let \( X' = (x', y', z')^T \), \( X'' = (x'', y'', z'')^T \) denote coordinate vectors with respect to two affine coordinate systems in 3D space, and let \( \Pi \) be a plane not containing the origin of the second coordinate system. The matrix \( A_\Pi, \) describing the coordinate transformation, according to Lemma 3.1.1, with respect to the plane \( \Pi, \) is proportional to the matrix describing the homography, \( H_\Pi, \) defined by \( \Pi \) from the image plane \( z' = 1 \) to \( z'' = 1, \) i.e. the images produced by normalized cameras with camera center at the origin of the two affine coordinate systems and with image planes in \( z' = 1 \) and \( z'' = 1 \) respectively.

Proof. If a normalized camera is used we have that

\[
\lambda_1 x_1 = f_1 X, \\
\lambda_2 x_2 = f_2 X,
\]

\( \Rightarrow A_\Pi \lambda_1 x_1 = \lambda_2 x_2, \)

where \( X \) denotes the 3D coordinates of a point on plane \( \Pi, \) \( f_j \) is the 3D coordinate vector of the camera center \( j, \) and \( x_j \) denotes the image coordinates of this point in image \( I_j, i, j = 1, 2. \) From this it follows that \( A_\Pi = \lambda H_\Pi. \)

Since the geometric interpretations of the eigenvalue system is independent of the camera coordinate systems, the following theorem holds also for non-normalized coordinate systems.

Theorem 3.1.1. Suppose that two images, \( I_1 \) and \( I_2, \) of two planar patches, \( \Pi_1 \) and \( \Pi_2, \) are given and that affine coordinate systems are defined in the images, cf. Figure 3.1. Let \( (H_1, H_2) \) be the corresponding pair of homographies. The generalized eigenvectors of \( (H_1, H_2) \) consist of the points on the projection in \( I_1 \) of the line of intersection between the two planes \( \Pi_1, \Pi_2, \) together with the epipole in \( I_1. \) The corresponding eigenvalues are \( c \) for the points on the line and \( cm \) for the epipole. Here \( m \) denotes a cross ratio \( \{f_1, f_2, C_2, C_1\} \) defined by (3.2) and Figure 3.2. If the epipole is not on the intersection line, then

\[
H_1 \begin{bmatrix} p_1 & p_2 & e \\ v & v \end{bmatrix}_\psi = H_2 \begin{bmatrix} p_1 & p_2 & e \\ v & v \end{bmatrix}_\psi \begin{bmatrix} 0 & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & cm \end{bmatrix}.
\]

Here \( p_1 \) and \( p_2 \) are the homogeneous coordinates two different points on the projection in \( I_1 \) of the intersection line, \( e \) is the epipole in \( I_1, \) and \( c \) is a scale arising from the fact that the homography matrices can be calculated only up to a scale.
3.1. GEOMETRY OF PLANES IN TWO IMAGES

The case when the epipole is on the intersection line between the planes was omitted above. When this happens the line defined by the two camera centers intersects the two planes in the same point. The cross ratio, \( m \), in the theorem will consequently be \( m = 1 \), and the vectors \( \mathbf{e}, \mathbf{p}_1, \mathbf{p}_2 \) will not be linearly independent eigenvectors. Assume that there exists an eigenvector, \( \mathbf{v} \) linearly independent of \( \mathbf{e}, \mathbf{p}_1, \mathbf{p}_2 \) with eigenvalue \( a \). This vector would fulfill the equation \( H_2^{-1} H_1 \mathbf{v} = a \mathbf{v} \). From the geometric interpretation of the equation we deduce that the only points that fulfill the equation are points on the intersection line and the epipole. Consequently there is no such eigenvector. It is always possible to write a matrix in Jordan canonical form. Since \( H_2^{-1} H_1 \) has two eigenvectors there must be two Jordan blocks. The Jordan block of size two corresponds to the epipole. Hence, if the epipole is on the intersection line, then

\[
H_1 [\mathbf{p}_1 \ \mathbf{w} \ \mathbf{e}] = H_2 [\mathbf{p}_1 \ \mathbf{w} \ \mathbf{e}] \begin{bmatrix}
1 & c & 0 \\
0 & 1 & c \\
0 & 0 & 1
\end{bmatrix},
\]

(3.4)

for some vector \( \mathbf{w} \) given by \( H_1 \mathbf{w} = c H_2 \mathbf{w} + H_2 \mathbf{e} \). This shows that if the epipole is on the intersection line it is still possible to discriminate the eigenvector corresponding to the epipole and the one corresponding to a point on the intersection line.

One practical problem when working with homographies is that the matrices describing them are determined only up to a scale factor. This makes it difficult to solve problems using linear methods. The following theorem shows that the matrices can be scaled so that \( c = 1 \) for all pairs of homographies. This fact is used in the following section.

**Theorem 3.1.2.** Given the homographies for \( N \) object planes \( H_1, H_2, \ldots, H_N \) from image \( I_1 \) to image \( I_2 \), for which the epipole does not belong to any of the intersection lines. The matrices describing these homographies can be scaled so that for each pair \( (H_i, H_j) \), the corresponding \( c_{ij} = 1 \), where \( c_{ij} \) is defined in (3.3).
CHAPTER 3. VIEW SYNTHESIS OF PIECEWISE PLANAR SCENES

Proof. Denote by \( D_{ij} \) the diagonal \( 3 \times 3 \) matrix containing on its diagonal the generalized eigenvalues for \( H_i \) and \( H_j \). Denote by \( V_{ij} \) the \( 3 \times 3 \) matrix with the generalized eigenvectors for \( H_i \) and \( H_j \) as columns. It is first shown that the above holds for \( N = 3 \). The scale for \( H_2 \) is chosen so that \( D_{12} \) has the double eigenvalue \( c_{12} = 1 \) and the scale for \( H_3 \) so that \( D_{23} \) has the double eigenvalue \( c_{23} = 1 \). It is now shown that \( D_{13} \) also has \( c_{13} = 1 \) and that \( D_{13} = D_{12} D_{23} \).

Without loss of generality it can be assumed that the image coordinate system has been chosen so that the line at infinity do not pass any of the eigenvectors. Then it is possible to scale the eigenvectors so that the third coordinates are 1. From the three equations \( H_i V_{ij} = H_j V_{ij} D_{ij} \) we have that

\[
D_{13} = V_{13}^{-1} H_3^{-1} H_2 H_2^{-1} H_1 V_{13} = V_{13}^{-1} V_{23} D_{23} V_{23}^{-1} V_{12} D_{12} V_{12}^{-1} V_{13}.
\]

Since the last column of \( V_{ij} \) is the epipole it follows that the last column of \( V_{ij}^{-1} V_{kl} \) is \((0 0 1)^T\). This gives

\[
D_{13} = \begin{bmatrix}
* & * & 0 \\
* & * & 0 \\
* & * & 1
\end{bmatrix}
\begin{bmatrix}
* & * & 0 \\
* & * & 0 \\
* & * & m_{23}
\end{bmatrix}
\begin{bmatrix}
* & * & 0 \\
* & * & 0 \\
* & * & m_{12}
\end{bmatrix}
= \begin{bmatrix}
x & 0 & 0 \\
0 & x & 0 \\
0 & 0 & m_{23} m_{12}
\end{bmatrix}.
\]

Since

\[
m_{13} = \{f_1, f_2, C_3, C_1\} = \{f_1, f_2, C_2, C_1\}\{f_1, f_2, C_3, C_2\} = m_{23} m_{12}
\]

it follows that \( x = 1 \). The theorem follows by repeating the above argument for every triple of homographies.

3.2 Synthesizing novel images

We will now deal with the problem of generating a new image from two or more given ones. Our objective is to calculate the homography matrix for each patch from one of the given images to the synthesized one. It is here shown that this is possible when the projections of the intersection lines between the planes are known in one image.

3.2.1 Homography calculus

The camera view to be synthesized is determined by 11 parameters. There are 3 for the position of the camera, 3 for the orientation, and 5 for the intrinsic camera parameters. The new camera position is chosen along the ray from the camera center and a point in
image \( I_1 \), represented by the epipole \( e_\ast \). The location in this direction is in projective space chosen as a cross ratio \( m_\ast = \{1, f_\ast, C_2, C_1\} \) involving two reference planes and the camera centers, cf. Figure 3.2. The rotation and the camera parameters are chosen as a homography \( S_1 \) for e.g. \( \Pi_1 \) from \( I_1 \) to \( I_\ast \). If the chosen epipole lies on one of the intersection lines between two planes, then these two planes cannot be used as reference planes for the cross ratio. This is due to the fact that regardless of where \( f_1 \) and \( f_2 \) are located, the cross ratio \( m = 1 \) if \( C_1 = C_2 \).

Since the projections of the intersection lines of the planes are known in e.g. \( I_1 \), all the homographies to the synthesized image can be calculated. The homography \( S_2 \) is found from (3.3) as

\[
c S_2 = S_1 [p_1, p_2, e_\ast] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & m_\ast \end{bmatrix}^{-1},
\]

where \( p_1 \) and \( p_2 \) are known points on the projection of the intersection line and \( e_\ast \) is the chosen epipole in \( I_1 \). The corresponding eigenvalues are 1,1 and the chosen \( m_\ast \).

Knowing the projections of the intersection lines for each of the other planes with these two planes, we know how four points on each plane are transformed to the synthesized image and the homographies can be calculated according to Section 2.4.1. In fact, Theorem 2 shows that it is possible to do this from only three points on these lines. By adding the constraints from lines which do not lie on one of the two planes with known homographies and some of the constraints from the epipole, an overdetermined system of equations is obtained. Due to Theorem 3.1.2 this system is linear. The equations in the case of four planes are detailed below:

\[
\begin{align*}
S_1 p_1^2 &= S_2 p_1^2 & S_1 p_1^3 &= S_2 p_1^3 & m_1 S_1 e_\ast &= S_2 e_\ast \\
S_1 p_1^3 &= S_2 p_1^3 & S_1 p_1^4 &= S_2 p_1^4 & m_2 S_1 e_\ast &= S_2 e_\ast \\
S_2 p_2^3 &= S_3 p_2^3 & S_2 p_2^4 &= S_3 p_2^4 & m_3 S_1 e_\ast &= S_3 e_\ast \\
S_2 p_2^4 &= S_3 p_2^4 & S_2 p_2^3 &= S_3 p_2^3 & m_4 S_1 e_\ast &= S_3 e_\ast \\
S_1 p_2^3 &= S_4 p_2^3 & m_1 S_1 e_\ast &= S_4 e_\ast & S_2 p_2^3 &= S_4 p_2^3 \\
S_1 p_2^4 &= S_4 p_2^4 & m_2 S_1 e_\ast &= S_4 e_\ast & S_2 p_2^4 &= S_4 p_2^4 \\
S_2 p_2^3 &= S_4 p_2^3 & S_3 p_2^3 &= S_4 p_2^3 & S_3 p_2^4 &= S_4 p_2^4.
\end{align*}
\]

Here \( p_i^{ij} \) denotes one point on the projection of the intersection line of the planes \( \Pi_i \) and \( \Pi_j \), \( p_i^{ij} \) another point on the line, and \( e_\ast \) is the chosen epipole. These parameters are all known together with the chosen \( S_1 \) and \( m_{12} \). There also exist nonlinear constraints on the unknown parameters, e.g.

\[
S_3 e_\ast = m_3 S_4 e_\ast.
\]

These have been omitted in order to simplify the calculations.
CHAPTER 3. VIEW SYNTHESIS OF PIECEWISE PLANAR SCENES

3.2.2 Visibility and occlusions

When the homographies to the new image all are estimated, they can be used to map the textures from the reference images. One problem that may occur is when parts of several patches are mapped to the same area in the synthesized image. This corresponds to that one plane covers parts of another plane for this particular camera location. We then have to find out which patch corresponds to the plane in front of the other in this area. The concept of being infront of has no meaning in projective spaces, however here we regard oriented projective geometry, cf. [64].

We begin by assuming there are only two planes in the scene. Assume that the homography matrices $S_1$ and $S_2$ from $I_1$ to the new image are known for two patches corresponding to two object planes $I_1$ and $F_2$. By (3.3) we estimate the projection of the intersection line, the epipole and the cross ratio, $I_1F_2$. It is evident from Figure 3.3 that $I_1$ is in front of $F_2$ on one side of the projection of the intersection line and behind on the other side.

We will now use the cross ratio to determine which plane is in front of the other on the epipole side of the intersection line. It turns out that this can not be done in projective space.

Now suppose there are more than two object planes and the intersection lines and cross ratios are estimated as above. It is easy to see that the cross ratios $m_{ij} > 0$ for all $i,j$, because a negative cross ratio would indicate that a plane has been viewed on different sides. By adding the information that e.g. $I_1$ is in front of $F_2$ on the epipole side of the intersection line and knowing that $m_{12} < 1$ we now can state the following theorem.

**Theorem 3.2.1.** Suppose that $I_1$ is in front of $F_2$ on the epipole side of the intersection line and that $m_{12} < 1$ (or $m_{12} > 1$). Then $I_4$ is in front of $F_4$ on the epipole side of the
3.2. SYNTHESIZING NOVEL IMAGES

3.2.1 \textit{intersection line if and only if} $m_{ij} < 1$ (or $m_{ij} > 1$), $i \neq j$.

\textit{Proof.} For the complete proof, a number of different cases have to be examined. We sketch the proof for one case. Suppose that $\Pi_1$ is in front of $\Pi_2$ on the epipole side of the intersection line and that $m_{12} < 1$. Then the current situation is either the one in Figure 3.3 or one of the situations in Figure 3.4. Note that for all these cases, camera 2 is in front of camera 1, i.e. the image plane of camera 1 is located between the two camera centers. The a priori information in the theorem consequently determines the relation between the two camera centers and the image plane for camera 1. Consider two other planes, $\Pi_i$ and $\Pi_j$. If we fix the point $C_i$, there are two cases that have to be investigated. Figure 3.5 shows $m(C_j) = \{f_1, f_2, C_j, C_i\}$ for different positions of $C_j$, for these two cases. From this figure it may be deduced that $\Pi_i$ is in front of $\Pi_j$ on the epipole side of the intersection line if and only if $m_{ij} < 1$.

Hence, the problem of visibility has a simple solution using the intersection lines, the epipole, and the cross ratio. The image to be synthesized may be divided into a number of regions decided by the projection of the intersection lines. In each region Theorem 3.2.1 can be used to decide in which order the patches should be mapped.

3.2.3 \textbf{Synthesizing points outside the planes}

It is now shown how to synthesize points which do not belong to any of the planes, such as the point $X$ in Figure 3.6. The objective is to calculate the image point $x_i$.

Assume that the corresponding image coordinates $x_1$ and $x_2$ are known together with the homographies $H$ and $S$ for the plane $\Pi$ from $I_1$ to $I_2$, and from $I_1$ to $I_2$, and that the epipoles in $I_2$ are known. The following notation is used: $X$ is the 3D point we wish to synthesize, $X^i$ is the intersection of $\Pi$ and the line $Xf_i$. The image of a point is denoted $x_i$ where $i$ is the image number. The epipoles in $I_2$ are denoted by $e_1$ and $e_2$ respectively.
CHAPTER 3. VIEW SYNTHESIS OF PIECEWISE PLANAR SCENES

![Figure 3.5](image)

Figure 3.5: The figure shows $m(C_j) = \{f_1, f_2, C_j, C_i\}$ for different positions of the point $C_j$. The two cases correspond to different fixed positions of the point $C_i$.

It is obvious that $x_s$ belongs to the intersection line of the image plane and the plane through $X, f_s$ and $f_1$. This is the epipolar line of $x_1$, and it may be calculated from the points $e^1$ and $x_1^2$. In the same way it is clear that $x_s$ belongs to the intersection line of the image plane and the plane through $X, f_s$ and $f_2$. This line may be calculated from the points $e^2$ and $x_s^2$. The point $x_s$ is the intersection of these two lines.

This construction is also useful to determine the fundamental matrix from one homography and two corresponding points which are not in the plane corresponding to the homography. This makes it possible to determine the fundamental matrix from six points, if it is known that four of them are coplanar.

More precisely, suppose we want to estimate the fundamental matrix for the images $I_1$ and $I_s$ in Figure 3.6. It is evident that the epipole in $I_s$ is on the line defined by $x_s$ and $x_s^2$. If we have a second point correspondence we will get a second line on which the epipole also must lie. Hence, the epipole is determined as the intersection between these two lines. When the epipole and a homography is known the camera matrix is defined by (2.5).

Just as for the method suggested in [40], a problem occurs when $f_1, f_2$ and $f_s$ are collinear. Then there are infinitely many intersection point of the lines. In this case we use another approach.

Consider Figure 3.7. The cross ratio $m = \{f_1, f_2, C_2, C_1\}$ is known from above. From Section 2.2.2 it is also known that the cross ratio $\{X^1, X^2, D, C_1\} = m$, and that $\{X^1, X^2, d, e\} = m$. Hence, since $m_s, e^1$ and $x^2$ are known, the image point $d$ can be calculated. The cross ratio $m_s = \{f_1, f_s, C_2, C_1\}$ is also known. Furthermore, $\{X^1, X^s, D, C_1\} = m_s$ and $\{x^1, x, d, e\} = m_s$. Since $m_s, e^1, x^1$ and $d$ are known, the image point $x$ can be calculated.

The problem when $f_1, f_2$ and $f_s$ are collinear may also be avoided if the trifocal tensor is used to transfer the points, cf. [27].

Another degenerated situation is when $f_1, f_2$ and $X$ are collinear, this means that the epipoles and the image points coincide. In this case there is not enough information
3.2. SYNTHESIZING NOVEL IMAGES

Figure 3.6: The geometric relations used to synthesize points not on one of the given planes.

Figure 3.7: The geometric relations used to synthesize points not on one of the given planes in the case when \( f_1, f_2 \) and \( f_s \) are collinear.
3.3 Experiments

A system for generating new images from two reference images based on the results above has been implemented. The following algorithm is used:

Algorithm 3.1. (View synthesis of piecewise planar scenes)

1. Manually select patches in each image, corresponding to planar surfaces in the scene.
2. Calculate the homographies.
3. Use (3.3) to estimate the intersection lines between the planes.
4. Choose a homography, an epipole and a cross ratio for the synthesized image.
5. Use (3.5) to estimate all homographies to the synthesized image.
6. Map the image patches to the new image using the estimated homographies. The texture corresponding to the most perpendicular view is used in the synthesis.

An experiment was performed on two images of a house. The homographies for 7 planes were estimated manually by finding a small number (between five and ten) of corresponding points on each plane and then fitting a homography to these point correspondences following the linear approach discussed in Section 2.4.1.

The results are shown in Figure 3.8. The camera parameters for a new view were decided using one additional image. Then this image was synthesized from the intersection lines and the estimated camera parameters. Figure 3.8 shows on the top row the given input images and in the center row synthesized images from the new viewpoints. These should be compared with the true images of the building in the bottom row, taken with the same camera parameters and locations as those used for the synthesis. The method to deal with occlusions which was described in Section 3.2.2, was used on the roofs and the left wall.

For the roof that is visible only in one image the projection of two intersection lines was manually estimated. Hence, view synthesis is possible using only one image if we manually are able to determine the intersection lines. This observation is used in the next section to make 3D models from single images.

The result of the algorithm is satisfying. The differences between the synthesized images and the true ones are quite small and not disturbing for the human eye. For instance, from the synthesized viewpoints e.g. the door on the right wall is reasonable synthesized although it is approximated to lie in the same plane as the wall. Note that there are no reflections in the windows in the synthesized images.
3.4 Conclusions

We have proposed a method for synthesizing new images of piecewise planar scenes which have been taken with an uncalibrated camera. A main theme is to work with textured planes and corresponding homographies. A complete algebraic characterization and a geometric interpretation of the generalized eigenvalues and eigenvectors of the matrices describing the homographies is given. The generalized eigenvectors of two such matrices are shown to correspond to the epipole and points on the intersection line between the two planes. The corresponding eigenvalues are shown to be $cm$, $c$, $c$, where $m$ is a cross ratio and $c$ a scale factor. A new image is generated without making an explicit 3D reconstruction. Instead all information about the structure of the scene which is needed to generate a new image, is encoded in the images as intersection lines between the planes. Using these parameters an algorithm for calculating the homography for each plane to a new image has been presented. It has also been shown that a number of planes could be synthesized by solving a linear system of equations. A method to deal with the problem of visibility was presented. The method is based on observations regarding the intersection lines, the epipole and the cross ratio. It has been shown how to synthesize points not belonging to any of the planes. In the degenerated case, when the new camera center and the camera centers for the reference images are collinear, the cross ratio is used.

The algorithms were tested in experiments on real images. It is difficult to validate the results quantitatively, however a comparison with true images which have been taken with the same camera parameters is satisfying.
Figure 3.8: The two reference images on top, the synthesized images in the center and images taken with the camera parameters used for the synthesis at the bottom.
Chapter 4

3D Reconstruction of Piecewise Planar Scenes

One strategy to solve the view synthesis problem is to create an accurate 3D model of the scene and then use this model to calculate novel views by projection. An advantage of this approach is that there already exists standard software for manipulating 3D models and generating new views of it. There are possibilities to improve the model, to remove or add objects, change the lighting, etc. A disadvantage is that it is very difficult to create a model that produces photo realistic images.

In this chapter, results from the previous chapter are used to create 3D models of piecewise planar scenes. It is also shown that reconstruction is possible from a single image, if the projection of the intersection lines between the planes in the scene is known. These intersection lines are often easy to extract manually. A projective framework is used but methods for obtaining Euclidean structure are presented. Since the reconstruction is given as a number of planes, constraints such as parallelity and orthogonality, may easily be used to upgrade the projective reconstruction. The algorithm has been tried on a number of images. Results from some of these experiments are presented.

4.1 Projective reconstruction

An algorithm for 3D reconstruction of a piecewise planar scene is derived. The homographies are related to the planes. It is shown how to calculate the parameters of the plane from the homography matrix, and vice versa.

Reconstructing planes from homographies

The camera matrix representation given in (2.5) is used. The matrices for two images may be expressed as,

\[
P_1 = [I \mid 0], \quad P_2 = H_1[I \mid -e],
\]

where \(I\) denotes the identity matrix, \(e\) the epipole in image one and \(H_1\) is the homography for the plane at infinity. In projective space any plane \(\Pi_1\) can be chosen. Using (2.4) and (2.8) we write,

\[
\begin{align*}
\lambda_1 x^1 &= P_1 x, \\
\lambda_2 x^2 &= P_2 x, \\
\lambda_3 x^2 &= H_2 x^1, \forall x \in \Pi_2.
\end{align*}
\]
Here \( x^i \) denotes the image of the point \( X \) in image \( i \). Substituting (4.1) and (4.2) in (4.3) and rearranging we get

\[
\frac{\lambda_2}{\lambda_1 \lambda_3} [I \mid 0] X = H_2^{-1} H_1 [I \mid -e] X, \forall X \in \Pi_2. \tag{4.4}
\]

Here \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) are all functions in the unknown \( X \). The following lemma says that the ratio \( \mu(X) = \frac{\lambda_2(X)}{\lambda_1(X) \lambda_3(X)} \) is constant, independently of \( X \). It also gives an interpretation of the constant. This makes it possible to eliminate unknown variables \( \lambda_1, \lambda_2 \) and \( \lambda_3 \) in (4.4).

**Lemma 4.1.1.** For all points in the plane \( \Pi_2 \), it holds that \( \frac{\lambda_2}{\lambda_1 \lambda_3} = c \), where \( \lambda_i \) is defined in (4.1)-(4.3) and \( c \) is the double generalized eigenvalue of the pair of homographies \((H_1, H_2)\). \( H_1 \) is the homography for the plane at infinity.

**Proof.** It will be shown that the statement holds true for three non-collinear points on \( \Pi_2 \). Then, since any point on \( \Pi_2 \) can be written as a linear combination of these three points, it follows that it will also hold true for all points on \( \Pi_2 \).

We choose two of the points on the intersection line between \( \Pi_2 \) and the plane at infinity. If the representation of the camera matrices from (2.5) is used these will have homogeneous 3D coordinates \( \begin{bmatrix} p_1^T & 0^T \end{bmatrix} \), \( i = 1, 2 \). By substituting this in (4.4) we get,

\[
\frac{\lambda_2}{\lambda_1 \lambda_3} p_1 = H_2^{-1} H_1 p_1 = c p_1, \quad i = 1, 2.
\]

Consequently \( \frac{\lambda_2}{\lambda_1 \lambda_3} = c \) and the statement holds true for these two points.

The third point is chosen as the point \( C_2 \) in Figure 3.2. The homogeneous 3D coordinates for this point are given by, \( \begin{bmatrix} m' e^T 1 \end{bmatrix} = [m' e^T 1]^T \), where \( m \) denotes the cross ratio in Theorem 3.1.1. By substituting this into (4.4) we get

\[
c \frac{\lambda_2}{\lambda_1 \lambda_3} m' e = H_2^{-1} H_1 (m' e - e) \iff \frac{\lambda_2}{\lambda_1 \lambda_3} m e = H_2^{-1} H_1 e = cm e.
\]

This shows that the statement holds true also for this third point on \( \Pi_2 \) and it follows that \( \frac{\lambda_2}{\lambda_1 \lambda_3} = c \) for all points on \( \Pi_2 \).

If the left hand side in (4.4) is subtracted from both sides and Lemma 4.1.1 is used, we get

\[ [H_2^{-1} H_1 \mid -H_2^{-1} H_1 e] X = 0, \forall X \in \Pi_2. \]

Since \( \Pi_2 \) has dimension two the following theorem follows.
Theorem 4.1.1. The $3 \times 4$ matrix
\[ M = [H_2^{-1}H_1 - cI | - H_2^{-1}H_1 e], \quad (4.5) \]
has rank 1. Here $c$ is the repeated eigenvalue and $e$ is the eigenvector corresponding to the single eigenvalue of the pair of homographies $(H_1, H_2)$.

Note that $H_2^{-1}H_1$ has a repeated eigenvalue $c$. Thus $(H_2^{-1}H_1 - cI)$ has rank 1. From this it is clear that the equation for the plane $\Pi_2$ is given by any non-vanishing row of the matrix $M$.

In short, to make a 3D reconstruction from two images the first step is to estimate the homographies for each plane from image one to image two, cf. Section 2.4.1. The equations of the planes may then be calculated from (4.5). Alternatively, these homographies may be used to estimate the intersection lines between the planes, and new homographies to a virtual image are then calculated according to Section 3.2.1. If the camera parameters of the virtual image are chosen e.g. as $S_1 = I$, $e = [0 \ 0 \ -1]T$ and $m \neq 1$, the reconstruction of $\Pi_2$ is given by any non-vanishing row of the matrix,
\[ [S_1^{-1} - I | S_1^{-1}(\cdot, 3)]. \quad (4.6) \]
Here $S_1^{-1}(\cdot, 3)$ denotes the third column of $S_1^{-1}$. Note that solving (3.5) we get $c = 1$ for all pairs of homographies.

### 4.1.1 Calculating homographies from equations of planes

In Chapter 3 an algorithm for synthesizing new images without making an explicit 3D model was proposed. As new camera parameters we chose an epipole, a cross ratio and a homography. Sometimes it is more convenient to choose parameters from the new camera matrix.

Here we present a method for calculating the homographies of a plane from image one to a new image, when the camera parameters for the new image are given in a camera matrix.

Theorem 4.1.2. Using the camera matrix representation in (2.5), $P_1 = [I \ | \ 0]$, $P_i = A_i[I \ | - e_3]$, the homography for a plane $\pi$ from image one to image $i$ can be written
\[ H_i = A_i + A_i e_3 v_\pi, \quad (4.7) \]
where $[v_\pi \ 1]X = 0$ is the equation of the plane.

Proof. Since the matrix $M$ in (4.5) has rank 1 it can be written as
\[ [H_2^{-1}H_1 - cI | - H_2^{-1}H_1 e] = -H_2^{-1}H_1 e[v \ 1]. \quad (4.8) \]
The theorem now follows from

\[
H_2^{-1} H_1 - cI = -H_2^{-1} H_1 \mathbf{ev}
\]

\[\iff\]

\[
c H_2 = H_1 + H_1 \mathbf{ev},
\]

where $A_i$ is identified with $H_1$ and $H_i$ is identified with $H_2$. \[\square\]

An alternative proof may be found in [27].

Accordingly, given the homographies for a number of planes from image one to image two, the equation of the planes may be calculated following Section 4.1. A new camera matrix is then chosen as $P_n = A_n[I - \mathbf{e}_n]$. The homography for each plane may then be estimated using (4.7). The problem with visibility can be solved in the same way as in Section 3.2.2.

### 4.2 Single view reconstruction

The only information about the structure of the scene used to generate a new image according to 3.2 or to make a 3D reconstruction according to Section 4.1 is the intersection lines between the planes. If sufficiently many such lines are estimated in one image it is possible to make a projective 3D reconstruction of the planes. In Section 3 it was described how the intersection lines are estimated when a number of images are available. In many applications the intersection lines are easy to manually extract in one image. Then it is possible to create a 3D reconstruction from a single image. Single image reconstruction with manual interaction is also investigated in [66, 5, 68]. In Section 4.4 examples of reconstructions created from single images using the proposed approach are shown. The algorithm used for the reconstructions is presented.

### 4.3 Euclidean reconstruction

In order to visualize the projective reconstruction we have to upgrade it to a Euclidean coordinate system. We make use of the fact that the planes in the scene often are parallel or orthogonal to each other, and that the intersection lines between the planes also have this property.

One approach to get a Euclidean reconstruction is to use metric and affine information sufficient to determine the absolute conic in 3D. The calibration matrix $K$ determines the image of the absolute conic $\omega = K^{-T} K^{-1}$. If only one image is available, the plane at infinity also has to be determined in order to decide where in 3D the absolute conic, $\Omega$, is located.

In many real situations the camera has been calibrated in advance or some of the intrinsic parameters are known, e.g. the skew or aspect-ratio. Otherwise metric information
about the scene may be used to calibrate the camera. If the metric of a plane is known and the homography from this plane to the image is estimated, two constraints on the calibration can in general be derived, cf. [66]. These may be formulated as

\[ h_1^T \omega h_2 = 0, \quad h_1^T \omega h_1 + h_2^T \omega h_2 = 0, \]  

(4.9)

where \( h_1 \) and \( h_2 \) are the first two columns in the homography above.

If two orthogonal vanishing-points are known, they will give one constraint on the calibration. This corresponds to the first equation in (4.9). Denoting the vanishing-points by \( v_1 \) and \( v_2 \), we get

\[ v_1^T \omega v_2 = 0. \]

In order to determine the plane at infinity in 3D one could for instance use the images of three points at infinity. Such points are often determined as the intersection of two parallel lines. One could also use the images of two points at infinity together with two orthogonal planes, or two parallel planes together with two orthogonal planes.

Note again that by working with planes and lines instead of points, concepts such as parallelity and orthogonality may be used in the reconstruction process.

When a Euclidean reconstruction of the planes has been determined, the reconstruction \( X \) of the image point \( x \) is obtained by solving,

\[
\begin{cases}
\lambda x = P_1 X \\
\pi_1^T X = 0
\end{cases}
\]  

(4.10)

Here \( \pi_1 \) denotes the parameters of the plane \( \Pi_1 \) and \( X \in \Pi_2 \).

### 4.4 Experiments

In order to validate the reconstruction algorithm a simulation was done. Two images of 12 points in three planes were generated and disturbed by Gaussian noise. The homographies for the planes were linearly estimated from the points as described in Section 2.4.1. A reconstruction of the planes and then of the points was made according to Section 4.1. The reprojected errors were then estimated. They are plotted as plus marks in Figure 4.1 for different sizes of noise.

Reprojection errors were also estimated using other methods. The line in Figure 4.1 shows the result using bundle adjustment, defining a lower bound on the reprojection error. The dots and the crosses are the reprojected errors in a reconstruction where the fundamental matrix was estimated linearly and non-linearly, respectively, cf. Section 2.4.2. The non-linear approach finds the fundamental matrix which minimizes the sum of the distances between the points and the corresponding epipolar lines. The accuracy in the proposed method is not very satisfying. Similar results were obtained in [44], where the
stability of estimations of the fundamental matrix from planes was investigated. Improved results could be obtained if more than four points on each plane were used.

Another simulation was performed in order to verify the stability of the reconstruction algorithm using intersection lines. Two images of 18 points in three planes were generated and disturbed by Gaussian noise. Six of the points define the three intersection lines, which were used to make a 3D reconstruction according to Section 4.1. The reprojected errors were then estimated and plotted in Figure 4.2. Reprojection errors were also estimated when the methods described above were used. The reprojected errors in the proposed method are of the same magnitude as methods based on the fundamental matrix, and for a small amount of noise even better than these. Consequently it is critical to estimate the intersection lines accurately to get a good reconstruction with the proposed method.

Three experiments using the reconstruction algorithm for a single image on real images were performed. The working process is described below.

Algorithm 4.1. (Reconstruction from a single image).

1. Estimate in the image manually some of the projections of the intersection lines of a number of planes.

2. Let $S_1 = I$, $e_x = [0 \: 0 \: 1]^T$ and $m_{12} = 2$.

3. Solve a system of equations similar to (3.5) to estimate all homographies to a virtual image.
4. Use (4.6) to estimate the equations of the planes.

5. Follow the approach in Section 4.3 to find a Euclidean reconstruction.

6. Manually define the borders of the planes in the image.

7. Calculate the 3D reconstruction of the border points according to (4.10)

8. Map the textures from the image to the 3D reconstruction.

In Figure 4.3(top) two points on each of the lines are marked, altogether 15 points on the intersection lines of 8 planes. A reconstruction was made according to Algorithm 4.1. In Figure 4.3 two views from the VRML reconstruction are shown. Additional experiments using the reconstruction algorithm are shown in Figure 4.4 and Figure 4.5.

4.5 Rectangles

In this section the aim is to reconstruct an indoor environment, e.g. to visualize an apartment. One way to do this is combining single view reconstructions created from images similar to the ones in Figure 4.7. From the previous chapter we know that if it is possible to estimate the intersection lines between the planes, a projective reconstruction of the scene may be calculated. If the camera calibration is known and additional information

\[ \text{signal-to-noise ratio (dB)} \]

\[ \text{root mean square reprojection error for different sizes of noise.} \]

Figure 4.2: Validation of the reconstruction algorithm with given intersection lines. Root mean square reprojection error for different sizes of noise. Line: Bundle adjustment, plus-marks: algorithm proposed in section 4.1, dot: nonlinear fundamental matrix, cross-marks: linear fundamental matrix.

\[ \times 10^{-3} \]
Figure 4.3: The results of an experiment on the reconstruction algorithm. The figure shows the original image on top of two views of the VRML model.
Figure 4.4: Another result of an experiment on the reconstruction algorithm. The figure shows the original image on top of two views of the VRML model.
Figure 4.5: Another result of an experiment on the reconstruction algorithm. The figure shows the original image on top of two views of the VRML model.
4.5. RECTANGLES

about the scene is available (cf. Section 4.3), a Euclidean reconstruction may be computed. This additional information is needed to determine the plane at infinity. If it is known e.g. that the floor and the ceiling are parallel, and that they are orthogonal to one wall, or that one wall is rectangular and one wall is orthogonal to the floor, it is possible to do this. There are degenerated cases for which the plane at infinity cannot be determined. In order to be able to combine the reconstructions the images must have some overlap.

A simpler model of an indoor environment is built by a number of consecutive rectangular walls with the same height but not necessary orthogonal. If this model is used the reconstruction process may be simplified. The only unknowns are the height-to-width ratio of the walls and the angles between consecutive walls. We now describe how these parameters are estimated using single images of the scene.

4.5.1 Pose estimation from a rectangle

Suppose that one image of a rectangle is given. It is assumed that the height-to-width ratio is unknown and that the calibration of the camera is known. If the image coordinates have been compensated for the calibration, the camera may be described by the projection matrix, \( P = [R\mid t] \). A point in space is projected to the image by the equation, \( \lambda x = PX \). The corners of the rectangle may without loss of generality be assumed to have coordinates \([0\ 0\ 0\ ]^T, [0\ a\ 0\ ]^T, [1\ 0\ 0\ ]^T, [1\ a\ 0\ ]^T\). Let the projection of these be denoted \( x_1, x_2, x_3, x_4 \), in coordinates compensated for the calibration. The present situation may be described by the parameters \( \lambda_1, \lambda_2, \lambda_3, \lambda_4, a, R, t \). There are eleven degrees of freedom. The projection equations give twelve constraints. It may then be possible to estimate the unknowns, unless the equations are dependent. In addition there is one constraint on the image points to be an image of a rectangle. A point at infinity corresponding to a particular direction from the camera center is projected to the vanishing point for this direction. In a calibrated camera the homogeneous coordinates for the vanishing point coincide with the corresponding direction from the camera center. From the equation for the angle between two directions, (2.3), we get that

\[
\mathbf{v}_1^T \mathbf{v}_2 = 0.
\]

Here \( \mathbf{v}_1, \mathbf{v}_2 \) are the vanishing points in the image for the directions of two orthogonal edges of the rectangle.

Inserting the corner coordinates in the projection equation we get,

\[
\begin{align*}
\lambda_1 x_1 &= [R\mid t][0\ 0\ 0\ ]^T = t, \\
\lambda_2 x_2 &= [R\mid t][0\ a\ 0\ ]^T = \alpha_2 + t, \\
\lambda_3 x_3 &= [R\mid t][1\ 0\ 0\ ]^T = r_1 + t, \\
\lambda_4 x_4 &= [R\mid t][1\ a\ 0\ ]^T = r_1 + \alpha_2 + t.
\end{align*}
\]
CHAPTER 4. 3D RECONSTRUCTION OF PIECEWISE PLANAR SCENES

Here \( r_i \) denotes column \( i \) in \( R \).

From this it is clear that \( \lambda_1 x_1 - \lambda_2 x_2 - \lambda_3 x_3 + \lambda_4 x_4 = 0 \). Here \( \lambda_1, \ldots, \lambda_4 \) may be computed up to an unknown scale by finding the null space to the matrix

\[
[\mathbf{x}_1 - \mathbf{x}_2 - \mathbf{x}_3 \, \mathbf{x}_4],
\]

In fact this argument holds true also for uncalibrated image coordinates. Notice the relationship to the affine shape of a rectangle, cf. [62].

The first column of \( R \) may be computed from \( r_1 = \lambda_4 x_1 - \lambda_2 x_2 \). Since \( r_1 \) should have unit norm the unknown scale factor for \( \lambda_1, \ldots, \lambda_4 \) may be calculated up to sign. The unknown camera parameters and the height-to-width ratio are now given by,

\[
\begin{align*}
\mathbf{r}_1 &= \lambda_4 \mathbf{x}_4 - \lambda_2 \mathbf{x}_2, \\
\mathbf{r}_2 &= (\lambda_2 \mathbf{x}_2 - \lambda_1 \mathbf{x}_1) / |\lambda_2 \mathbf{x}_2 - \lambda_1 \mathbf{x}_1|, \\
\mathbf{t} &= \mathbf{t}_1, \\
\mathbf{a} &= |\lambda_2 \mathbf{x}_2 - \lambda_1 \mathbf{x}_1|.
\end{align*}
\]

To sum up, from one calibrated image of a rectangle it is possible not only to estimate the position and orientation of the camera, but also the height-to-width ratio. In addition, there is a constraint on the image coordinates to be an image of a rectangle. This constraint may be used to determine one parameter in the camera calibration, e.g. the camera constant, if the other parameters are known.

Another observation is that if the camera calibration and the homography from a plane with known metric to the image is known, we may ‘synthesize’ an image of a rectangle and determine the extrinsic camera parameters. The homography may according to 4.9 further be used to determine two calibration parameters. Consequently, it is possible to determine the position and orientation of a camera with two unknown calibration parameters if a homography from a plane with known metric to the image can be estimated. This fact is used in Chapter 9.

4.5.2 Calculating the angle between two planes

In order to calculate the angle between two planes the vanishing points for two intersecting lines in each plane are estimated. If the plane defined by these two lines is orthogonal to the two original planes the angle \( \theta \) between the planes fulfill

\[
\cos(\theta) = \frac{\mathbf{v}_1^T \mathbf{v}_2}{\sqrt{\mathbf{v}_1^2} \sqrt{\mathbf{v}_2^2}},
\]

cf. Figure 4.6. Here \( \mathbf{v}_1 \) and \( \mathbf{v}_2 \) are the homogeneous coordinates of the vanishing points compensated for the calibration.

60
4.5.3 Experiments

A room in an apartment was reconstructed by means of the ideas presented above. The four calibrated images in Figure 4.7 were used. The intersection lines between the walls, the floor and the ceiling were manually estimated. The height-to-width ratio for each rectangular wall and the angle between two consecutive walls were computed. A top view of the resulting model is shown in Figure 4.8, together with a floorplan of the room. The model seems to be accurate. Two textured views of the model are shown in Figure 4.9.

4.6 Conclusions

We have proposed a simple linear method for projective reconstruction of piecewise planar scenes from uncalibrated images. The main idea is to work with planes as primitives, and to use the information about the structure of the object inherited from the projections of the intersection lines. These are often easy to manually extract or could be more automatically estimated if more than one image is available. We have shown that projective 3D-reconstruction is possible from a single uncalibrated image if a number of intersection lines are estimated manually.

Ideas regarding how to upgrade the projective reconstruction to a Euclidean were presented. By using planes as basic features it is easy to incorporate conditions such as planes or lines being orthogonal or parallel to each other. Such conditions are common, particularly in man-made environments.

A few results relating the homographies to the camera matrices were derived. These are used for the direct synthesis described in Section 3.2 when the new camera parameters
CHAPTER 4. 3D RECONSTRUCTION OF PIECEWISE PLANAR SCENES

Figure 4.7: Four images of a room with the intersection lines used for the reconstruction marked.

Figure 4.8: The top view of the reconstruction compared with a floorplan of the room.
Figure 4.9: Two textured views of the reconstruction.
are chosen as a camera matrix.

A quantitative validation of the stability of the algorithm showed that it is critical to estimate the intersection lines accurately. If this can be done with the same accuracy as for the point correspondences, the proposed algorithm performs equally well or better than several of the standard approaches.

The 3D models created from a single image were also satisfactory. When working with one image there is less information to use. As a consequence of this, problems with e.g. occlusions and accuracy may appear.

An algorithm for building a 3D models of an indoor environment was presented. It is based on simple geometric relations of rectangular objects.
Chapter 5

Structure and Motion using Quivers

Traditional structure from motion is based on point correspondences. The extraction of the points of interest, is often done by estimating the size and direction of the gradient of the intensity of the image in a small region. However, even in the case of corner points, information about edge directions often is not used. To deal with such information, the following notation is introduced.

**Definition 5.0.1.** By an $n$-quiver, denoted $(\mathbf{x}, \mathbf{d}_1, \ldots, \mathbf{d}_n)$, is meant a point $\mathbf{x}$ and the directions of $n$ rays $\mathbf{d}_1, \ldots, \mathbf{d}_n$ from the point.

Such new features are particularly common in urban scenes, where there often are many structures containing corners, e.g. buildings.

A quiver with three directions corresponds to what we usually call a corner. Quivers with one or two directions can e.g. be used if not the whole of the corner is visible in all images or for corners on a planar surface.

In this chapter structure and motion estimations from quiver features are studied. One difficult step in the reconstruction process is to find correspondences between features in different views. It is here assumed that these correspondences are known. The focus is here on the description of the geometry.

Correspondences between quivers are somewhat more difficult to handle than for points or lines. Correspondence between two quivers also involve the correspondences between the directions in the quivers. This means, for example, that for two quivers with three directions, there are six possible combinations of directions. Having additional information, e.g. that the quivers stems from a corner on a solid building, the number of combinations reduce to three.

A number of minimal cases for reconstruction by means of quiver features, is investigated. By a minimal case we mean that the problem is solvable for the available data but omission of some data would give an infinite number of solutions.

The problem to solve minimal cases to perform 3D reconstruction is of both theoretical and practical importance. Algebraic solutions which are obtained from minimal cases may be used to bootstrap robust estimation algorithms such as RANSAC or LMS schema [23, 73, 80].

5.1 Preliminaries

Let $\mathbf{X}$ be the homogeneous coordinates of an object point. Using the pinhole camera model in Section 2.3, an object point $\mathbf{X}$ is projected to an image point, represented in
homogeneous coordinates by \( \mathbf{x} \), according to the equation

\[
\lambda \mathbf{x} = P \mathbf{X} = KR[I | -t] \mathbf{X}.
\]

Here \( P \) is the camera matrix and \( \lambda \) is a scalar.

The cameras \( \{P_1, P_2, \ldots, P_n\} \) and the lines in correspondence \( \{l^1, l^2, \ldots, l^n\} \), where \( l = [l_1^1, l_2^1, l_3^1]^T \) are considered. The points \( \mathbf{X} \) that project to image points \( \mathbf{x} \) on the line \( l^i \), lie on a plane \( \Pi_i^T \mathbf{X} = 0 \), where the plane parameters are \( \Pi_i = P_i^T l_i \).

These planes intersect along a line in 3D. This imposes a constraint on the data, which may be written

\[
\text{rank}[P_1^T l^1 \ldots P_n^T l^n] = 2. \tag{5.1}
\]

In the case of three cameras, this constraint may be written

\[
l^3 \times T_{ij}^{ijk} l^1_i l^2_j = 0. \tag{5.2}
\]

Here \( T_{ij}^{ijk} \) denotes elements in the trifocal tensor, defined by (2.13).

In a similar way the four constraints from three corresponding points, \( \{p_1, p_2, p_3\} \), where \( p = [p^1, p^2, p^3]^T \) may be expressed as

\[
p_3^1 T_{ij}^{ijk} l^1_i l^2_j = 0, \tag{5.3}
\]

where \( l^1 \) and \( l^2 \) are lines passing through the point corresponding to \( p_3 \) in views one and two. Different choices of these lines give in total four linearly independent constraints on the tensor.

Since a point is described by two parameters in each image and has three degrees of freedom in 3D, we have that corresponding points in three images give three constraints on the geometry and in four images they give five constraints. Corresponding lines give two and four constraints in three and four views respectively. However, for a quiver the point is located on the line causing the constraints from the point and the lines to be dependent. In Table 5.1 the number of constraints from different features in correspondence in three and four images is given.

In order to understand how much information that is needed to solve the structure and motion problem for different camera models using different kinds of features, the
5.2 A point with one direction

In this section 1-quivers are considered. A correspondence between such features give rise to four and seven constraints on the geometry for three and four images, respectively according to Table 5.1. Three such quivers seen in three affine views is shown to be a minimal case, which can be solved linearly with a unique solution.

### Table 5.1: The number of constraints from different corresponding features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>3 images</th>
<th>4 images</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>point</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>point + direction</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>point + 2 directions</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>point + 3 directions</td>
<td>6</td>
<td>11</td>
</tr>
</tbody>
</table>

### Table 5.2: The total number of degrees of freedom for different camera model systems.

<table>
<thead>
<tr>
<th>Camera model</th>
<th>3 images</th>
<th>4 images</th>
</tr>
</thead>
<tbody>
<tr>
<td>projective</td>
<td>$3 \cdot 11 - 15 = 18$</td>
<td>$4 \cdot 11 - 15 = 29$</td>
</tr>
<tr>
<td>affine</td>
<td>$3 \cdot 8 - 12 = 12$</td>
<td>$4 \cdot 8 - 12 = 20$</td>
</tr>
<tr>
<td>calibrated</td>
<td>$3 \cdot 6 - 7 = 11$</td>
<td>$4 \cdot 6 - 7 = 17$</td>
</tr>
</tbody>
</table>

The total number of degrees of freedom for different camera model systems is shown in Table 5.1. Let $j$ be the number of cameras and $p$ the number of parameters in the camera model. Then the degrees of freedom for the current situation are given by

$$j \cdot p - n,$$

where $n$ is the degrees of freedom in the solution parameter space, cf. [27].

By comparing the constraints from different features and the degrees of freedom in the cameras, several candidates for minimal cases are found. A minimal case is when the image data exactly constrains the geometry.

Three of these problems, that are of most practical use is studied. The first case is three 1-quivers in three affine views, the second is three 3-quivers in three projective views and the third is two 3-quivers in three affine views.

5.2 A point with one direction
CHAPTER 5. STRUCTURE AND MOTION USING QUIVERS

Problem formulation and solution

A 1-quiver seen in three affine views gives four constraints on the camera geometry. Since three affine cameras have twelve degrees of freedom according to Table 5.1, this define a minimal case. A point in three views gives essentially three constraints on the camera geometry but gives four constraints on the trifocal tensor, cf. [27]. Similarly it turns out that a 1-quiver gives five constraints on the affine trifocal tensor.

The necessary and sufficient conditions for the trifocal tensor for projective cameras are given in [22, 21, 30]. For a trifocal tensor to be affine 11 of these constraints are linear,

\[
\begin{align*}
T_1^{13} &= 0, & T_1^{23} &= 0, & T_1^{33} &= 0, & T_1^{32} &= 0, \\
T_1^{31} &= 0, & T_2^{13} &= 0, & T_2^{23} &= 0, & T_2^{33} &= 0, \\
T_2^{32} &= 0, & T_3^{31} &= 0, & T_3^{33} &= 0.
\end{align*}
\]

These 11 constraints plus the 15 linear constraints from (5.2) and (5.3), given by the quivers, are sufficient to linearly estimate the trifocal tensor, since the trifocal tensor has 27 entries and is determined up to scale.

Experiments

In Figure 5.2 three views with three 1-quivers are shown. The quivers were extracted from the images manually. The affine trifocal tensor was estimated linearly according to the method in the previous section. Figure 5.2 also shows a number of points that were extracted from the images to test the estimated solution. From the trifocal tensor, the camera matrices was calculated. These were then used to intersect the extracted points. The resulting 3D point reconstruction was then projected to the images using the estimated cameras. The original points are shown as asterisks and the reprojected points as circles. Most of the errors are of the order of a pixel. Points further away from the used quivers are reprojected less accurately.

5.3 A point plus two directions

In this section the 2-quiver is considered. From Table 5.1 we see that 2-quivers in correspondence give five and nine constraints, respectively, on the geometry of three and four images.

There are no minimal cases for any of the camera models in Table 5.1. However, the 2-quiver case is particularly interesting since many algorithms for finding point correspondences in images also give information about two directions at every point. For example, to find corner points with high precision, the intersection of two lines corresponding to the edges around the corner is estimated.
Figure 5.2: Three images with three extracted 1-quivers shown in each image. Also shown are a number of points used to test the estimated solution. Original points are shown as '*' and reprojected points as 'o'.
CHAPTER 5. STRUCTURE AND MOTION USING QUIVERS

An almost minimal case

The information in four 2-quivers viewed by three projective cameras gives a system that is slightly overdetermined. There are 20 equations and 18 unknowns. One 2-quiver gives six linear constraints on the trifocal tensor, so there are $26 - 6 \times 4 = 2$ parameters left using the linear constraints. There are additional necessary constraints which are non-linear, cf. [30]. These may be used to solve for the remaining parameters. All of these constraints are of degree three in the two parameters. If two of these non-linear constraints are used, up to $3^2 = 9$ solutions may be obtained. These solutions are then verified by using the other non-linear constraints to find the unique solution.

Comparing reprojection errors

A simulation was performed to compare the accuracy in the trifocal tensor estimation using point-, line- and 2-quiver-correspondences in three images. In this simulation we envision a scenario where points are extracted from the images as intersection of lines. A number of pairwise intersecting lines were randomly placed in 3D and projected into three images. Then noise was added to the image lines. First the trifocal tensor was computed using only point correspondences. The points were given as the intersections of the pair of lines. Then the trifocal tensor was computed using only line correspondences, and finally the trifocal tensor was computed using the 2-quivers given by the point of intersection and two directions on the line.

In order to compare the different tensors, the reprojection error of a lattice was calculated using the true motion and the motions derived from the estimated tensors. The lattice was created as a number of equally spaced points in a cube. The reprojection error for the tensors with different amounts of noise on the image lines is shown in Figure 5.3 for 15 and 25 features. From the figure it seems that the performance is best for the 2-quivers. This is perhaps not so surprising since there are more information in this case, but it shows that if both point and line information are available one should use quiver features to stabilize the estimation.

In a second simulation the reprojection error was estimated in the same way as above for different amount of noise. However, for this simulation the number of different features was chosen so that they impose the same number of constraints on the camera geometry. For instance, in the right of Figure 5.4 the reprojection error is shown for the motion determined by using of 40 points, 60 lines, 24 quivers and 24 lines & 24 points. The simulations show that the motion determined by quivers has the best reprojection error if a lot of information is available. In situations where only a few features are available, the motion determined by point correspondences is the best.
5.3. A POINT PLUS TWO DIRECTIONS

Figure 5.3: Reprojection error with various amounts of noise for motion estimated using points (solid), lines (dashed) and 2-quivers (dotted) for 15 features to the left and 25 features to the right.

Figure 5.4: Reprojection error for motion estimated using points (star), lines (plus) and 2-quivers (circles) and lines & points (triangle) for different number of features.
5.4 A point with three directions

In this section features which consist of a point with three directions are studied. In particular the minimal cases of three such features viewed in three projective images and two such features in three affine views are solved.

Three 3-quivers in three projective images

A feature with one point and three directions seen in three images gives six constraints on the camera geometry, cf. Table 5.1. Given three such features we have 18 constraints, which is exactly the same as the degree of freedom in three uncalibrated projective cameras. One could try to solve the problem using the trifocal tensor as in the previous section. Each feature would then give seven linear constraints on the tensor which would give the tensor up to a five parameter family of solutions. The necessary constraints on the tensor may then be used to solve for the parameters. However, by choosing a specific parameterization of the cameras the polynomial system will be of lower degree than the one given by the necessary constraints on the tensor.

Problem statement

A special parameterization of the cameras is selected. A projective coordinate system is chosen in the scene, such that the 3 points in space are assigned to the projective coordinates

\[
\begin{bmatrix}
X_1 & X_2 & X_3
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 1 & 1
\end{bmatrix},
\]

and image coordinates are chosen so that the corresponding 3 image points in each image have coordinates

\[
\begin{bmatrix}
x_1 & x_2 & x_3
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}.
\]

Nine degrees of freedom in the projective structure are fixed using the three points. The six remaining degrees of freedom in space are determined by specifying one of the directions in each quiver. These are given by specifying the following three points on each line,

\[
\begin{bmatrix}
X'_1 & X'_2 & X'_3
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & -1 \\
0 & 1 & 1 \\
0 & 0 & -1
\end{bmatrix}.
\]
Only two degrees of freedom remain in each image, so we can only specify two of the three corresponding lines. The three corresponding lines in each image are chosen as,

\[
\begin{bmatrix}
1 & l_x \\
1 & l_y \\
-1 & 0
\end{bmatrix}.
\]

By this choice of coordinates the camera matrices have the following form

\[
P = \begin{bmatrix}
    l_y/l_x - a & -l_y/l_x & a & 0 \\
    0 & b & -1 & 1 - b \\
    0 & b & a & 0
\end{bmatrix},
\]

with \((a, b)\) unknown.

Each camera is thus parameterised by two parameters. To this end, we have used the point and one of the directions in each 3-quiver. The remaining 2 \(\cdot\) 3 feature lines are used to solve for the six parameters \((a, b, c, d, e, f)\). From the rank constraints in (5.1), we get one polynomial constraint from each line. Due to our special choice of coordinate systems in the images, these turn out to be of total degree three for the lines in quiver one and two, and of degree two for the lines in quiver three. The polynomials have the following structure:

\[
\begin{align*}
*bd + *adf + *bcf + *af + *cf + *de + \\
... + *ad + *bf + *bc + *be + *bd = 0, \\
*bd + *adf + *bcf + *af + *cf + *df + *de + \\
... + *ad + *bf + *bc + *be + *bd = 0, \\
*bce + *ade + *acf + *af + *cf + *ce + *de + \\
... + *ad + *ae + *bc + *be + *ac = 0, \\
*bce + *ade + *acf + *af + *cf + *ce + *de + \\
... + *ad + *ae + *bc + *be + *ac = 0, \\
*af + *cf + *de + *ad + *bc + *be + *a + *b + \\
... + *c + *d + *e + *f = 0, \\
*af + *cf + *de + *ad + *bc + *be + *a + *b + \\
... + *c + *d + *e + *f = 0.
\end{align*}
\]

Here \(*\) denotes a known coefficient which depends only on image data.

The solution

The system (5.10) consists of six polynomials in six variables of which four are of total degree three, and two are of total degree two. A system of this type may have up to \(3^4 \cdot 2^2 = 324\) solutions according to the theorem of Bezout, cf. [16]. The number of solutions are 324 in the case of a dense system with general coefficients.
A better bound on the number of solutions is given by the so called mixed volume of a system, cf. [16]. This bound was calculated to 42 for the system in (5.10). Consequently there are at most 42 solutions to this system.

In order to find a solution a polynomial solver called PHC, which is described in [78] was used. This program starts with calculating the mixed volume of the system. After this the solver proceeds by constructing a more easily solved system with the same structure as the original problem. It solves this system and the 42 solutions are propagated to the solutions of the original system by a homotopy continuation method. A number of these solutions go out to infinity and are not solutions to the original problem.

In order to study the number of solutions of approximately 1800 examples with three 3-quivers in three images were simulated. Then PHC was used to solve the simulated cases. The solver found 12 solutions in each case, of which some were complex.

We assume that all quiver directions are known in all images. Until now we have not used this information, but have only considered lines without orientation. The information about the directions may be used to remove solutions with the wrong orientation. It turns out that in most cases there is only one true solution to our problem! To the left in Figure 5.5 a histogram of the number of solutions with the right orientation is given and to the right a histogram of the number of real solutions.

**Experiments**

In Figure 5.6, three images of a scene with three quivers are shown. The quivers were manually extracted. Then by using the specific parameterization and the solver, the cam-
era geometry was determined. In this case there were ten real solutions, but only one of them had the right orientation. In order to test this solution, a number of points were extracted in the scene. The estimated cameras were used to determine the 3D coordinates of these points. They were then reprojected into the images. The result is shown in Figure 5.6.

The reprojection errors in Figure 5.6 are quite small. The original points are shown as asterisks and the reprojected points as circles. The largest errors are for points furthest from the quivers which were used for the reconstruction, as would be expected.

**Two 3-quivers in three affine views**

A quiver with three directions gives six constraints on the camera geometry, cf. Table 5.1. Since three affine cameras have twelve degrees of freedom two such features in three affine views is possibly a minimal case.

**The solution**

The problem may be solved by using an approach similar to the one in Section 5.2. Corresponding 3-quivers give seven constraints on the trifocal tensor. Two such features together with the 11 linear constraints for the tensor to be affine consequently give a one parameter solution space, \(26 - 2 \times 7 - 11 = 1\). Next we use the nonlinear constraints on the tritensor to find the solution. These constraints give third order polynomials in the parameter. It turns out that the polynomials have the same three solutions. Consequently there may be three solutions to the trifocal tensor.

Two of the solutions have the same elements in the upper left \(2 \times 2 \times 2\) block of the corresponding tensors. The tensor corresponding to the third solution is the difference between the first two. This tensor does not correspond to a valid motion. Thus, there are up to two solutions to the problem.

Sofar the information in the orientation of the quivers has not been used. The number of solutions with the right orientations were calculated in approximately 4000 simulations. Figure 5.7 shows the result. For more than 75 percent of the cases there was a unique solution.

**Experiments**

Figure 5.8 shows three images in which two 3-quivers manually have been extracted. The camera geometry was estimated according to the previous section. In this case there was a unique solution. In order to test this solution, a number of points were extracted in the scene. The estimated cameras were used to determine the 3D coordinates of these points. They were then reprojected into the images. The reprojection errors are mainly because the affine camera model is not valid for these images.
Figure 5.6: The three images used with three 3-quivers marked. The reprojected points using the solution obtained from the quivers is also shown with original points as ‘*’ and reprojected points as ‘o’.
5.5 Conclusions

In this chapter we have introduced the notion of a quiver. A quiver is defined as a point and a number of direction from the point.

We have studied quivers with one, two and three directions. For these types of quivers there are three particularly interesting minimal cases. For 1-quivers, three such features seen in three affine views give a unique linear solution. For three 3-quivers in three uncalibrated projective views there are up to 7 solutions, but simulations show that in most cases there is a unique solution. Two 3-quivers seen in three affine views give up to two solutions, but also in this case there are in most cases a unique solution. Experiments on real data show that the solutions obtained from the minimal data gives good estimates on the camera geometry.

We have also investigated 2-quivers seen in three views, where simulations show that such features give more stable estimation of the trifocal tensor as opposed to only using line or point features.

Figure 5.7: The figure shows the number of solutions with the right orientation for approximately 4000 simulations.
Figure 5.8: The three images used with two 3-quivers marked. The reprojected points using the solution obtained from the quivers is also shown with original points as '*' and reprojected points as 'o'.
Part II

Computer Vision Systems
Chapter 6

An Automatic System for View Synthesis

6.1 Introduction

View synthesis is the problem of generating images from new viewpoints of a three-dimensional scene using the information in a number of given images. This problem has been studied in the computer vision as well as in the computer graphics community. The focus in computer vision has mainly been on geometry, creating a correct geometric model of the scene. In order to render the model it is often represented as a textured polygon, e.g. [81, 52, 71] or represented as a volume, e.g. [69, 19, 39] where each volume element has a specific RGB value. It could also be represented as a depth map [70], as layered depths and sprites [60, 6], or as a collection of images and correspondences [40].

Appropriate ways to texture the model have been given less attention. Usually the polygons are textured from the image which has the most perpendicular view of the polygon, or by a texture blended from a number of images. A volume element, a so called voxel, can e.g. be given the median RGB of a number of corresponding pixels. How to solve problems with e.g. shadows, specularities and different lighting conditions is not obvious. There has been work done in this direction, cf. [17], but surely there is still plenty to do. Methods without creating explicit 3D models have also been presented. In [40] fundamental matrices were used and in [4] trifocal tensors were used to synthesize new views from reference images.

In computer graphics the problem is often referred to as image based rendering. Since the goal is to generate new images without the use of a 3D model, the geometry is of less importance for these techniques. View interpolation or morphing techniques as a mean to generate smooth transitions between the reference images have been developed, e.g. [7, 14, 13, 79]. However, there are imagebased rendering techniques which (to some extent) generate geometrically correct new views of the scene, cf. [57, 58, 42]. A number of methods are relating to the plenoptic function, e.g. [24, 47, 41, 29, 38]. This is a 5D function describing the flow of light at every 3D position and 2D viewing direction. For a survey on the imagebased rendering techniques, see [37].

In this chapter an automatic system for generation of images from new viewpoints of an object using the information in a number of given uncalibrated images is presented. In contrast to the approaches in the previous chapters, there are no assumptions about the scene. Thus this approach is more general and may be used in a wide range of scenarios. First the cameras are reconstructed and a coarse metric 3D reconstruction
of the scene is produced. In a second step we estimate a depth map and a visibility map for the image to be generated. The depth map is then refined by a colour consistency argument using the reference images where the current depth is visible. When the new image is rendered, the RGB for each pixel is chosen as the RGB for the corresponding pixel in the closest visible image.

The images produced by view interpolation, which are close to the reference images, often look very photo-realistic. The principal weaknesses of the image-based rendering techniques are due to the lack of depth and occlusion problems. Reconstruction based methods can produce images for arbitrary viewpoints. On the other hand, it has proven to be a difficult problem to automatically obtain a geometrically correct 3D description for a complete scene. Our approach of generating novel views stems from the reconstruction based methods, but it does not rely on accurate depth information. Only an approximate model of the scene is required. The flexibility in our approach is appealing. The starting reconstruction could be of any kind, made automatically or known in advance, and the extent of the refinement may be controlled. The refinement in every generated image makes it possible to handle deforming objects, e.g. a talking head, using the same starting model. The use of view-dependent texture increases the quality of the generated images, in particular when generating in-between images.

There are mainly two types of existing automatic systems for general view synthesis that are closely related to the proposed system: (i) the approach used in e.g. Leuven/Oxford [27] and (ii) the space carving approach [39]. Disadvantages for both of these approaches compared with the one presented here are that: (i) view-dependent effects such as specularities are not handled well, (ii) the resolution is dependent on the 3D model and is not adapted to the novel view, (iii) incorrect estimation of the 3D model results in poorly synthesized images and (iv) the scenes must be rigid.

To be more precise, in contrast to space carving methods, our method does not have to consider the resolution of the model. In addition, you make the model more detailed close to the virtual camera and less detailed further away. In particular when a small number of images are generated our method is more computationally efficient than the space carving approach. For the proposed method there are no problems handling large scale environment. Another difference is that our method gives the depth in an interval which has the best colour consistency rather than giving the smallest depth for which the colour consistency is sufficiently good. Furthermore, there is no threshold for the colour consistency to be set.

If we compare our system to the system developed in Leuven/Oxford, we find that rather than making a dense depth map for each of the given images by pairwise correlation, our method makes a depth map for the image to be generated using all the images where the current depth is visible.

The background to the problem is described and a detailed description of the system is given. The ultimate verification of a view synthesis system is of course to present generated images and validation of their correctness. This is performed in the experimen-
tal section which also includes a discussion of the limitations. Finally, some concluding remarks are given.

6.2 System approach

The proposed scheme is embedded in a system that handles a wide range of scenarios, e.g. large scale environments as well as small detailed objects. The developed system has three steps. First, the intrinsic and extrinsic camera parameters are estimated, and a coarse approximate geometric 3D reconstruction is created according to the state-of-the-art reconstruction algorithms. The coarse model is then refined for this particular view, and finally the refined model is rendered with textures from the closest images. These three parts will now be described in detail.

6.2.1 Coarse 3D reconstruction

The first step is to create a coarse 3D reconstruction. This reconstruction may e.g. be a triangulated surface reconstruction, a volumetric space carving model or a silhouette based model. For a silhouette based model the proposed procedure is particularly attractive, since the silhouette based approach creates a model which is the visual hull of the object and needs additional processing. For some applications there may be an approximative model available.

The implemented system first computes the camera positions. The following procedure is applied in order to do this, cf. [27].

- Extract and track features in the images.
  We use Harris corner detector [26], to find point candidates. These points are tracked in the image sequence by performing affine correlation of a patch around each point.

- Estimate projective camera geometry.
  The outliers from the previous step are sorted out by the RANSAC algorithm, [23]. Then a primary projective reconstruction is calculated by use of trifocal tensors.

- Auto-calibration.
  In order to obtain a Euclidean reconstruction, constraints are imposed on the calibration, e.g. that certain intrinsic parameters should be constant during the sequence. A reconstruction fulfilling these constraints is found by the bundle adjustment algorithm, cf. [32, 33]. At the same time, this procedure minimizes the reprojection error.

The problem of automatically calculating an accurate geometrical 3D description for a general 3D scene has proven to be quite difficult. Our approach does not rely on an accurate 3D model, only a coarse model is needed. For complex scenes, a visibility model
is also needed in order to determine which scene elements are visible in a given image. Two possible approaches to create the coarse model are:

- Triangulation.
  Given the tracked image points these may be triangulated in order to obtain a surface model. A popular method to triangulate a set of points is the Delaunay triangulation.

- Space carving.
  Space carving is a volumetric technique. It starts with a filled voxel space, and then carves off inconsistent voxels in the model [39]. A voxel is inconsistent if the corresponding pixels in different images do not agree, e.g. in colours. A visibility model for the volume is automatically obtained as well.

Other approaches, e.g. finding dense correspondences across the views, or the previously discussed silhouette based schemes may also be adopted.

### 6.2.2 Refining the depth map

In order to generate a novel image, the camera parameters for the virtual view are chosen. From the 3D reconstruction and the chosen camera parameters, a depth map and a visibility map for the new image is created. For each pixel in the new image, we associate a depth and a number specifying in which of the reference images a point at this depth is visible. As the depth map may be inaccurate, it will be refined. In order to do this, consider the line defined by the center of a specific pixel and the camera center of the new image, cf. Figure 6.1. Along this line search on an interval, given by the current depth of the pixel and a percentage specifying the length of the interval, to find a number of local minima of a colour consistency function. Such a function measures how well the colours of a projected point in all of the images agree for a particular point in 3D space. If the consistency is high it is likely that the 3D point corresponds to a point on the surface of the object. In the experiments, see Section 6.3, a colour consistency function $C$ defined as

$$ C(X) = \sum \text{Var}(I_i(P(X)) | \text{X visible in image } i) $$

was used. Here $X$ is a point in the interval along the line, and the variance is calculated using only the images for which the current depth is visible. The sum is taken over a neighborhood of the current pixel. In order to cope with fine structures, it is required that the neighborhood is small, e.g. a single pixel or $3 \times 3$ pixels. On the other hand, such small support increases the frequency of outliers. Therefore we choose the depth so that the depth map is as smooth as possible. In our implementation the depth map is first filtered by a median filter, then the depth for each pixel is chosen to be the local minimum which is closest to the filtered map. This heuristic preserves fine structures and at the same time it keeps the number of outliers low. In contrast to space carving, this
6.2. SYSTEM APPROACH

Figure 6.1: An illustration of the refinement of the depth map.

approach gives the depth which has the best colour consistency in an interval rather than giving the smallest depth for which the colour consistency is good enough.

6.2.3 Rendering novel the image

When the precise depth for each pixel in the new image is known, the next step is to determine the RGB-value of the pixels in the novel view. This may be achieved in several ways. Here two possible approaches are presented.

- Interpolation.
  The RGB-value for a pixel in the new image is interpolated from all corresponding pixels in the (visible) reference images. In theory, it is possible to obtain a higher resolution than in the reference views.

- View-dependent texture.
  The RGB-value for a pixel in the new image is chosen as the RGB-value in the (visible) reference image which is closest to the novel view. In this way, the texture is not smoothed and view-dependent effects are captured.

We have adopted the second approach. This view dependent texture has nice properties, e.g. errors in the geometry will be less obvious in the generated images and it handles specularities and occlusions well.

In order to render a new image we start with the pixel which has the smallest depth and give it the RGB for the projection of a points at this depth into the closest camera for which this point is visible. The pixel from which the RGB is taken is now marked, so that no pixels in the generated image, other than neighbor pixels with similar depth, will be
given an RGB from this pixel. We do this so that if the depth for the background is not correctly estimated, the background in the generated image will not contain texture from the foreground. The procedure now continues to the pixel with the second smallest depth, and so on. Pixel values close to a change of image texture may have to be interpolated from both images in order to avoid edge effects in the generated image.

6.3 Experiments

The system has been tested in different situations and using different starting models. All the results shown here concern interpolated images, i.e. the camera for the new view is located in the complex hull of the camera centers in the reference images. The generated image is compared with a real image with the same camera parameters. The image for comparison has not been utilised for synthesizing the new image. The search interval for the refinement of the depth map was chosen as 10 percent of the distance between the camera center and the initial depth, if not otherwise stated.

All images are in colour, except for the first experiment, where gray-scale images were used. The results are illustrated below. At the end of this section, limitations of the proposed scheme are described.

6.3.1 Experiment 1: Shoe

In the first experiment, the system was tested on four images of a shoe, see top of Figure 6.2. The cameras were estimated by manually identifying a number of corresponding points and the starting 3D model was chosen as a single 3D plane. In order to compensate for the simple starting model, the length of the search interval was in the same order as the distance between the plane and the camera centers. The first approach was used for the rendering, i.e. the RGB was determined as the median for the corresponding points in the reference image. The generated image is shown at the bottom right of Figure 6.2. On the left, the true image with the same camera parameters, is shown. In spite of the poor initial 3D geometrical model, the result is appealing. This experiment indicates that the quality of the starting model is not crucial, as long as the search interval is chosen in accordance with the quality of the model.

6.3.2 Experiment 2: Flower

In a second experiment, a sequence of seven images of a flower was used to generate a new image. The first and the last image is shown in Figure 6.3(top). The cameras were automatically reconstructed using the technique described in Section 6.2.1. In total, 250 point features were tracked and reconstructed.

Based on the positions of the 3D scene features, a voxel model of size $160 \times 160 \times 160$ was created using space carving, cf. [39]. The model is far from perfect. There is a carved
Figure 6.2: **Top:** Four images of a shoe. **Bottom:** (left) A real image taken at the same position as the synthesized image, (right) synthesized image using a plane as initialization.
hole in one of the leaves and there are many voxels that should be removed to reflect the true geometry of the flower. In the center of Figure 6.3, an image of the model seen from the novel viewpoint is given.

Another drawback of the model is that it covers only parts of the scene - the flower, while the majority of the background is not present in the model. As the main focus in the sequence is the flower, the precision of the background is not crucial. Moreover, the background is farther away than the flower and thereby less sensitive to errors in depth. For the generation of the novel image, the background depth was simply set to a constant value based on the maximum depth of 3D scene features. The result is compared with a real image taken from the same viewpoint in Figure 6.3.
6.3. EXPERIMENTS

6.3.3 Experiment 3: Face

In order to see how the system would handle a deforming object, an experiment was performed on a human face. The idea is to create a simple 3D model from one series of images and use this model for other series of images.

The first series of five images (see Figure 6.4(top)) was used to make a polygon model of the face and the background. The model is based on a Delaunay triangulation of the tracked image features, as described in Section 6.2.1. The frontal view of the face was synthesized based on the two images in Figure 6.4(top). The frontal view was also created directly from the polygon model by mapping texture from the two side views. The resulting images are shown in Figure 6.4(bottom) together with a real image from the same viewpoint. The angle between the viewing directions for the two reference images is approximately $40^\circ$. The quality of the two computer generated images of the frontal view is high - the quality is slightly better with depth refinement.
The second series consists of three images of the same person, but with a completely
different facial expression, see Figure 6.5. We use the polygon model created from the
first series of images to generate novel views, but now with different reference images.
The left and right images in Figure 6.5(top) were utilised as reference images to generate
a frontal view of the face. The result is shown in the left of Figure 6.5(bottom). The
angle between the two viewing directions of the reference images is approximately 60°.
Notice for example that the left and right areas around the neck have severe distortions.
This is due to that these parts are only visible in one image, and consequently the depth
refinement is not applicable. In the right of Figure 6.5, the same procedure was applied
to produce a novel image, using the left and center views in Figure 6.5 as reference. The
result looks strikingly life-like.

6.4 Conclusions

In this chapter an automatic system for generating new images of a scene using uncali-
brated images has been presented.

The system first produces a coarse 3D starting model. From this a depth map and a
visibility map is generated for the new view. The depth map is then refined using a colour
consistency argument on the images for which the current depth is visible. This procedure
also enables generating new views of deforming objects using the same starting model.
The model is rendered with view dependent textures. In our experience, this increases
the quality of the generated images and problems with specularities and occlusions may
be handled reasonably. In our opinion it is possible to do the depth refinement and
rendering in real-time.

A drawback with our approach is that if one wants to generate many new views of
a static scene, then a depth map is refined for each new view which could be computa-
tionally ineffective. The system would therefore, in comparison with other methods,
perform best when the task is to generate single images or images of a non-static scene.
The presented method has no ideal solution to problems with occluding boundaries and
occlusions. In the current implementation the refinement is performed using multiple in-
tervals near depth discontinuities, but this is something we have to study more to be able
to handle better. Similar to the space carving method, the colour constancy condition
does not work well for images taken far apart and with varying lighting condition.
Figure 6.5: **Top:** Three images of a sad looking person. **Bottom:** Two computer generated face images, with initial 3D model from the first series of face images.
Chapter 7

Visualization of Fridge Contents

In this chapter we present a model based approach to the view synthesis problem. The idea is to adapt a pre-defined model of the scene to the images. This requires some knowledge of the objects in the scene, or possibly that the objects can be recognized in the images. In particular we will describe a completely automatic model-based computer vision system for visualization of fridge contents. However the ideas are applicable to a number of scenarios including building reconstruction. In this case characteristic features of a building, e.g. windows, doors etc., may be recognized and modeled in detail.

7.1 Introduction

The intelligent home was recently a very discussed concept. The idea is to connect home devices to a central computer. As a part of this there is a desire to extract information about the contents in a refrigerator and communicate this to a remote user. A typical scenario is that a user on his computer at work wishes to see what is currently in the fridge, or that the user may call the fridge from the supermarket to find out if there is e.g. milk at home. There are also ideas about automatic generation of shopping lists and a self-ordering fridge. When the fridge is running out of a product it can order a new one from a net-store.

Different techniques have been proposed to deal with this problem, e.g. barcode tags and electronic tags. A more appealing technique is to use cameras and computer vision techniques. In this case the products do not have to be modified, e.g. by the use of tags, neither does the user have to change the behaviour when manipulating the contents of the fridge.

This work was done in collaboration with Electrolux, a large manufacturer of refrigerators. The objective of this part of the project is to detect events, to segment the scene and to build a 3D model of the contents of the fridge for visualization, as well as to extract 3D information and texture to be used for identification. The identification problem was handled by another participant in the project and will not be considered here.

One central idea is that the footprint, i.e. the bottom surface of objects in the fridge, contains valuable information about its contents. In order to obtain these footprints, it is assumed that the shelves of the fridge are semi-transparent. In this way the footprints of the objects are visible for a camera located below a shelf. All objects in the fridge have to be placed directly on the shelf, and may not be placed on top of other objects. In addition they have to be inserted so that the entire object is visible for the camera. They may then be moved behind other objects.
7.2 System Approach

This section describes the proposed system. The focus of the work is to build a complete and automatic system for demonstration purposes. Due to limitations in time, simple solutions have been chosen to many of the image processing problems. If the system was to be implemented for commercial use, more sophisticated methods would have been used to solve the basic image analysis problems. This would definitely improve the quality of the generated models.

7.2.1 Overview

A number of cameras are placed at fixed positions in the fridge, so that they view the objects in the fridge. There is one image taken from below and one taken from above each shelf. The shelf is prepared with a plastic film so that the footprint of the objects clearly stands out in the image taken from below. In Figure 7.2.1 a fridge equipped with a number of cameras is shown. The cameras are placed so that in each image you view the items on one shelf and the footprints of the shelf above. For the bottom shelf a mirror is used to capture the footprints. The objects in the fridge are assumed to be standing on the shelf, not on top of each other.

Figure 7.1: A fridge equipped with the proposed system.
Assume that a model of the current contents of the fridge is available. The system detects a change in the fridge, e.g. insertion, and updates the model.

The first stage of this procedure is the **motion detector**. The motion detector compares images from the same camera taken at different time-instants, to detect when there is any motion in front of the cameras. One or more cameras may be used for this. The processing at this stage is simple but requires a video flow from the cameras, making it an ideal candidate for hardware implementation. A few images are then forwarded to the next stages of processing.

Useful information may be achieved from an image of the **footprint** of the objects standing on a shelf in the fridge. From these images, it is possible to determine what the user has been doing with the contents of the fridge, for example if an object has been inserted or moved. Depending on the type of action, the system moves, removes, or adds an object to the current model.

When a new object is inserted the information from the footprints is used to determine the 3D shape of the inserted object. A 3D model is adapted to the footprint and a contour image of the object. The footprint also provides the position of the object.

The 3D information and the footprint information may together with the texture of the segmented object be supplied as input to an identification system. This part of the problem is not considered here.

Finally the contents of the fridge is presented to the user. This is achieved by showing two-dimensional images, three-dimensional models which may be viewed from different directions, or a simple list of objects.

A schematic overview of the system can be seen in Figure 7.2. In the following sections the different parts of the system are described in detail.

Figure 7.2: Overview of the system solution. Images are captured by the cameras. These are then processed and 3D information is extracted. A database keeps track of objects currently in the fridge. The contents of the database can be visualized by the user in different ways.
7.2.2 Motion detection

The first part of the system is the motion detector. The purpose of the motion detector is to detect when changes occur in the fridge. When this happens one image is captured by each of the cameras. These images are stored for further processing.

In order to detect a change in the fridge we must be able to measure the difference between two images. Let

\[ A(x, y) = [A_r(x, y) \ A_y(x, y) \ A_b(x, y)], \quad B(x, y) = [B_r(x, y) \ B_y(x, y) \ B_b(x, y)] \]

be two colour images captured with a camera. Define a measure of the total difference between images \( E(A, B) \), as

\[
E(A, B) = \sum_{x,y} |A(x, y) - B(x, y)|^2. \quad (7.1)
\]

We say that there is motion in front of the cameras when the difference defined by (7.1) of two consecutive images is larger than some threshold. An action is defined to have occurred when there is a sequence of no motion, then motion and then no motion again from any of the cameras. Every time an action has occurred, an image is stored by each camera for further processing.

![Figure 7.3: To the left an image of some objects taken from below the transparent glass shelf is shown. The right image shows the same objects on a shelf covered with a plastic film.](image)

7.2.3 Footprint view

The footprint image is an image showing the bottom surface of every object placed on a shelf, see Figure 7.3. It may be rectified to a correct metric by a homography specific for each camera and shelf pair, c.f. 2.2.2. Since we assume fixed cameras and fixed shelves, this homography is estimated in advance and only once. It would however be possible to
allow moving cameras and determine the rectifying homography on-line. By processing the footprint images it is possible to get important information about the objects, e.g. for detecting and classifying actions. These are used to determine the position and for certain objects the orientation, in addition they are used in the reconstruction of the 3D shape. The footprint may also be used in the identification process.

In order to decide what type of action that has occurred, we analyze the footprint images. We define four different types of basic actions:

- **Insertion.** A new object has been put on a shelf.
- **Removal.** An object has been removed from a shelf.
- **Movement.** An object on the shelf has been moved (translated and/or rotated).
- **No action.** Nothing has changed, the two images show the same objects at the same position.

These are the only allowed manipulations in the fridge in the current implementation. It is for instance not allowed to move an object at the same time as a new object is inserted, although it would be possible to cope with such actions.

Define \( D(x, y) \) to be the difference image between footprint views before and after an action has occurred, cf. Figure 7.4. We wish to segment the regions which correspond to the footprint of an inserted and a removed object. This may be performed by using sophisticated segmentation algorithms, e.g. using statistical properties of the footprints. However, in the implemented system the difference image is thresholded with two different thresholds. The detected regions are then processed by morphological operations to get a smoother appearance. The classification is performed by comparing the area of the inserted and removed regions. Denote by \( A_I \) and \( A_R \) the area of the inserted and removed regions respectively. The classification is performed as follows:

- If \( A_R < A_0 \) and \( A_I < A_0 \) then no action has been performed,
- else if \( A_I \geq A_0 \) and \( \frac{A_I}{A_R} > d \) then an object has been inserted,
- else if \( A_R \geq A_0 \) and \( \frac{A_R}{A_I} > d \) then an object has been removed.
- otherwise, an object has been moved.

Here \( A_0 \) is the smallest area that a region has to have to be regarded as an inserted/removed object. If an object has been moved, then \( A_I \approx A_R \) and \( \frac{A_I}{A_R} \approx 1 \). Consequently, the threshold \( d \) is used to differ a movement from an insertion or a removal.

This procedure fails if objects are put on top of each other. The current implementation only handles actions on single objects.
7.2.4 Handling actions

Depending on the type of detected action, the response of the system varies as discussed below.

Removal

If the action is decided to be a removal, the object corresponding to the largest removed region in the difference of the footprint images is deleted from the model. This object is found by comparing the center of mass of the removed region with the location of the objects in the model.

Movement

In the case of a movement, the object in the model corresponding to the largest removed region is moved to the center of mass of the largest inserted region. The orientation of the object is also updated by determining the central axis of the footprint. If there is an overlap between the regions, cf. Figure 7.4, we have to use the original footprint images as well, in order to decide which object has been moved and where to. A disadvantage of only analyzing the movement in the footprint views is that the new orientation of e.g. a cylinder cannot be determined. For rectangular objects there is an ambiguity of $180^\circ$.

Insertion

When an insertion is discovered the position of the inserted object is determined by analyzing the footprint images in the same way as above. In order to determine the 3D shape of the object we also use the difference between a side view of the object before and after the action. From this image the contour of the inserted object is estimated, cf. Figure 7.5. In order to do this, the inserted object must not be occluded by any other object. In the current implementation the contour is determined by thresholding a difference image which has been compensated for shadow effects.
7.2. SYSTEM APPROACH

Figure 7.5: The two images to the left show the shelf before and after insertion of the object. The third image from the left shows the difference image. The rightmost image is a thresholded image.

Since the projection matrices for the cameras are known, we may adopt a shape from a predefined class of objects to the footprint- and contour image. Objects suiting the model are objects with uniform cross-sections parallel to the shelf, and with the central axis orthogonal to the shelf. This class includes many of the most common objects in a fridge, e.g. cylinders with central axis orthogonal to the shelf, boxes, cones with the top on the central axis and bottles. A number of these objects are shown in Figure 7.6. Spherical objects, e.g. apples, are approximated reasonably within the class, however certain objects, e.g. cylinders lying down, do not fit at all.

The parameters in the model which we have to determine are the scales of the footprint for different heights above the shelf, c.f. Figure 7.7. Since the cameras are fixed, the camera matrix $P$ may be assumed to be known. The planes defined by the shelves are also known in advance, thus the location of the footprint in 3D may easily be determined from the rectified images. For every height above the shelf, different scales of the footprint shape are tested to find the largest scale for which the projected shape fits in the contour
image of the object. When the scales are determined for every height they are filtered to achieve a smoother appearance.

### 7.2.5 Visualization

In order to visualize the contents of the fridge the 3D model was implemented as a texture-mapped polygon in OpenGL. The front textures were used to texture both the front and the back of the models. This procedure gives a complete textured model that can be viewed from any angle. The quality of the reconstruction was satisfying for the application.
7.3 Examples

A completely automatic system for reconstruction and visualization of the contents in a fridge based on the discussed ideas, was implemented. A few frames from a movie sequence of a user inserting objects in a fridge are shown in Figure 7.8. Images of the 3D reconstruction are presented in Figure 7.9. The quality of the reconstruction is satisfactory and would be even better if more sophisticated methods were used to solve the image processing problems. Another attractive feature is that the reconstruction is segmented into objects. This makes it possible to remove objects to see what is in the middle of the shelf. It is also valuable for the identification.

7.4 Conclusions

We have presented an automatic vision system for determining and visualizing the contents of a fridge. The system detects a change in the fridge and updates the current 3D model by moving, removing or inserting objects.

The focus of the work so far has been to build a complete system. In the future it would be desirable to implement more sophisticated algorithms for some of the basic image processing problems. Another interesting continuation is to mount the cameras in the door and use the image sequence produced when opening and closing the door. If an approximative 3D model is created and the motion is known up to one parameter, there are great possibilities to improve the model.

The refrigerator has to have prepared shelves and there are some restrictions on the user, e.g. that objects must be inserted fully visible and standing on the shelf. Is spite of this the system is very flexible compared with non-vision solutions based on e.g. tags or bar-codes.
Figure 7.9: Four images of the 3D reconstruction created from the images in Figure 7.8
Chapter 8

Detecting Windows in City Scenes

The ideas to use complex models to create 3D reconstructions, may be applied not only to the application in the previous chapter but to building reconstruction as well. Features such as windows, doors etc. are detected and a model of the current feature is applied to the images. In this chapter a recognition system for detecting in particular windows in one image is presented. Such a system may be used in a numerous applications. Besides model-based building reconstruction it may also be used in a city guide application, where the location and orientation of a person is estimated from an image of the surroundings. The location of the windows of the current building is a good cue that may be used in order to simplify the problem. This could be used, e.g. with handheld devices such as personal digital assistants (PDA) and mobile phones equipped with cameras.

8.1 Introduction

The focus of the chapter is on detecting windows from learned image templates in a city scene. A detection system is presented which locates windows with various appearances, poses and under different lighting conditions. Furthermore, several possible applications are given for which the detection system can be utilised, e.g. rectification, 3D reconstruction including wide baseline matching and as just mentioned recognition and pose estimation.

Object recognition is a complex problem and has been the subject of intensive research for many years, cf. [25]. In the case of buildings, the recognition problem can be simplified if all windows can be reliably located. For example, the invariants of the window positions can be used to index a large database of models, cf. [55]. A further simplifying circumstance is that the windows often are located on a small number of planes. Recognizing planar objects has been studied in e.g. [53].

The proposed system is trainable, i.e. it learns from examples. This makes the system flexible and it is straightforward to extend it to other objects of interest. The learning technique is based on support vector machines [12, 77], which has been successfully applied to, for example, face detection [49], text categorization [34] and pedestrian detection [50].

8.2 The detection system

In this section, the architecture of the system is described. There are three main parts: feature extraction, support vector learning and window detection in new images.
8.2.1 Feature extraction

The first issue to be addressed is that of representation of the data. We have scaled and registered the images of windows to a template of size $64 \times 48$ pixels. In Figure 8.1 some examples of typical windows that we want to detect are shown. The aspect ratio is not changed when an image is rescaled. The window frame is placed at the top left corner.

As the system learns from examples, or more precisely, it learns from features extracted from the examples, one has to decide which features to use. Wavelet features have proven to be useful for many applications of object detection, e.g. [48, 50]. Inspired by these results, we have also chosen to use a wavelet basis, but it is modified to suit our application in mind.

The first set of features that are extracted from the templates use the Haar wavelet [46],

$$\begin{bmatrix}
1 & \ldots & 1 & -1 & \ldots & -1 \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
1 & \ldots & 1 & -1 & \ldots & -1
\end{bmatrix},$$

(8.1)

with filter kernels of sizes $16 \times 16$ and $32 \times 32$. In contrast to the wavelet transform, the kernels are shifted by $1/4$ of the support of the kernel. Both horizontal and vertical kernels are applied which results in a total of 264 features.

It is evident from the examples in Figure 8.1 that the frame of the window is a common characteristic. In order to capture the window frame more accurately in the feature set, we have used filter kernels of the following type:

$$\begin{bmatrix}
1 & \ldots & 1 & 0 & \ldots & 0 & -1 & \ldots & -1 \\
\vdots & \ddots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & \ldots & 1 & 0 & \ldots & 0 & -1 & \ldots & -1
\end{bmatrix},$$

(8.2)

as well as its transpose. The submatrix of 1’s (and $-1$’s) was chosen to be of size $16 \times 8$. The number of zeroes was chosen to match the horizontal and vertical border frames for the two filter orientations. This resulted in 72 additional elements in the feature vector.

The above filters assume gray-scale images. In order to incorporate colour variations as well, we apply the filters to each of the three colour channels R, G and B, and then the one with maximum norm is kept.

Thus, in total a window template is represented by a $264 + 72 = 336$ feature vector. This representation will be evaluated on a large training set in Section 8.3.
8.2.2 Support vector machines

In recent years, support vector machines (SVM) have become very popular in statistical learning theory [12, 77]. In contrast to traditional learning techniques, such as neural networks which only minimizes training set error, SVM optimizes both training error and the complexity of the classifier at the same time. This makes it possible to capture the statistics of sparse, high-dimensional spaces with relatively few examples. Another attractive feature is that the problem can be solved with quadratic programming and thereby the optimal solution is always guaranteed. Again, compare this with neural networks where a nonlinear optimization problem is solved with back propagation.

The SVM classification is based on computing the sign of the following function:

\[ f(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b, \]  

(8.3)

where \( \mathbf{x} \) is the feature pattern to be classified, \( \mathbf{s}_i \) are the support vectors, \( N_s \) is the number of support vectors, \( K(\cdot, \cdot) \) is the kernel function and \( \alpha_i, y_i, b \) relate to the trained decision surface. In all our experiments, we have used a polynomial kernel of degree three and the software provided by [45] to learn the parameters in the classification function (8.3). In Appendix A an introduction to support vector machines is given.

8.2.3 Detecting windows in new images

Given a new image, the objective is to determine the positions of all windows in the image. We use a direct approach by shifting the template window of size \( 64 \times 48 \) over the whole image. The procedure is repeated at different scales. One example is illustrated in Figure 8.2 where the white rectangles show the detected windows. In this image multiple detections have been removed.

This brute force method is time consuming as the classification function (8.3) at each template position in the input image at several scales has to be calculated. There are several possible methods to speed up the computations, such as reducing the number of feature to represent the data [50] or reducing the number of support vectors [11] making it possible to run in near real-time. The reduction of the feature set is further discussed in Section 8.3. Our implementation, which is done in Matlab, may also be speeded up by using the fast Fourier transform (FFT) for the filter calculations. Currently, the computation time for one image of size \( 640 \times 480 \) pixels with all 336 wavelet features is approximately 30 minutes.

8.3 Evaluation

The system has been trained on a database consisting of 1290 window templates (including mirror images) and 13972 templates of negative examples. See Figure 8.1 for some
CHAPTER 8. DETECTING WINDOWS IN CITY SCENES

Figure 8.2: An image processed by the detection system. The white rectangles mark the detected windows.

examples of window templates.

The negative examples were collected in two phases. First, a preliminary version of the system was trained on a subset of the negative examples. The resulting system was applied to images containing no windows. The falsely detected windows in these images were then added to the database of negative examples, and a second training of the system was done on the complete database.

In order to evaluate the performance of the system, an additional set of 150 window templates and 1000 negative templates was collected. These templates were not used in the training of the system. By varying the threshold for the classification function $f(x)$ in (8.3), the detection rate and the number of false positives can be varied. Ideally, the detection rate should be 100% and the false alarm rate 0%.

For comparison, four different ways of extracting features from the templates were tested, cf. Section 8.2.1.

- Colour templates with all 336 wavelet features.
- Gray-scale templates with all 336 wavelet features.
- Colour templates with 264 wavelet features, with filters only of the first type (8.1).
- Colour templates with 72 wavelet features, with filters only of the second type (8.2).
The result is shown in Figure 8.3. As can be seen, the colour and gray-scale versions of the system with all 336 wavelet features perform equally well on the test data, keeping the detection rate over 95% even with very a low false alarm rate. As expected, when reducing the size of the feature set the result gets worse, but still acceptable.

8.4 Applications

The detection system described above may be used in numerous applications. We discuss a few here and show some experimental results.

8.4.1 Rectification

In order to do metric measurements in a perspective image of a plane, the image first has to be rectified. This corresponds to a transformation of the original image to an image that would have been obtained from a fronto-parallel view of the plane. The rectification is essential in many applications, e.g. the recognition and pose estimation described below.

The frame of a window is usually rectangular and has two horizontal and two vertical edges. In addition to this the windows are often located next to each other at the same height, or below each other. These properties make it suitable to rectify an image by first detecting windows in an image and then find the map which makes the windows rectangular and located below or next to each other. In the literature there are several suggestions on how to find the map, cf. [43, 76]. To the left in Figure 8.4 an image of a building is shown and to the right the automatically rectified image. Note that this
Figure 8.4: To the left an image of a building is shown, and to the right the rectified image.

rectification only allows for metric measurements if the calibration is known or if there is only one unknown parameter in the calibration.

8.4.2 Recognition and pose estimation

By using a single image of a building in a city scene we want to determine which buildings are present and estimate the relative position of the observer. We assume to have models of a number of buildings. The models consist of a metric template indicating where the windows are located.

We begin by detecting all windows in the image. From a number of invariants based on the window positions, a hypothesis building is chosen in the database. The homography between the image and the template of the current building is then estimated. Knowing the calibration of the camera, the position and orientation may now be calculated from the homography with the algorithm described in [67].

This application will be discussed in detail in the following chapter.

8.4.3 3D reconstruction

A challenging application would be to create 3D reconstructions of large scale environments based on recognition. A number of features, e.g. windows, doors, trees, cars, etc. may be recognized and complete 3D models of these features could be used in the reconstruction. Similar ideas have been investigated in [18]. Recognition algorithms may further be used in traditional 3D reconstruction. A major problem in reconstruction is to establish point correspondences between images, particularly for images taken with a wide base-line. If e.g. the windows are detected in a number of images, these may be used to simplify the correspondence problem.
8.5 Conclusions

A system with the capability of detecting windows in images has been presented. The system is based on support vector machines for learning the appearance of window templates. An experimental evaluation has shown good performance making it applicable in man-made environments.

Several applications of the detection system have been described, such as recognition, pose and rectification including experimental validations. The extension to incorporate other objects in the detection system is straightforward and this opens up an even wider range of application scenarios.
Chapter 9

A System for Automatic Pose Estimation

9.1 Introduction

In this chapter the problem of automatic determination of the position and orientation of a camera using a priori information about the surroundings is considered. In particular we study the application of pose estimation in a city scene when models of the surrounding buildings are available.

The objective is to develop a system that handles different lighting conditions as well as shadows, specularities and occlusions. In order to handle different lighting either you have to compensate for the lighting or you have to use features which are indifferent to the lighting, or possibly do both. This work has been focused on the use of edges, which are not so sensitive of the lighting.

The first part of the problem is to recognize the building from a set of models. Object recognition is a complex problem and has been the subject of intense research for many years. However, assumptions about buildings simplifies the task for this application. It is assumed that the buildings are mainly planar and that these planes have both horizontal and vertical edges. This is often the case for buildings as they have windows and doors. To each building to be recognized in the images, it is assumed that have a model is given. The model consists of a number of planes in 3D, indicating where dominant edges are situated. For each of the building candidates, the corresponding model is matched to the image and a measure of fit is estimated. The building which has the best fit is chosen, providing it exceeds a threshold.

The second part of the problem consists of estimating the position and orientation of the camera. This is possible if metric information about the model is available and the model has been matched to the image.

Recognizing planar objects has been studied in e.g. [53, 54]. The approach presented here has similarities with the approach in [15]. However, searching for the best match in [15], a local search in six parameters is performed whereas we reduce the search to two searches in two parameters. These are efficiently implemented by fast convolution. In addition to this, we consider matching with 3D models and it is shown that also in this case the search space can be reduced to two dimensions. We further deal with uncalibrated as well as calibrated cameras. This automated system approach for camera pose estimation is validated by experiments on images from real city scenes.
9.2 Preliminaries

We model the camera as a pinhole camera. A point in 3D space with homogeneous coordinates \( \mathbf{X} \), is projected onto an image point with homogeneous coordinates \( \mathbf{x} \), according to,

\[
\lambda \mathbf{x} = P \mathbf{X}.
\]

Here \( P \) is a \( 3 \times 4 \) matrix and \( \lambda \) is a scale. The camera matrix \( P \) may be decomposed as,

\[
P = KR(I \mid - t),
\]

where \( I \) is the identity matrix, \( R \) is a \( 3 \times 3 \) orthogonal matrix representing the orientation of the camera, \( t \) is a 3 vector representing the position of the camera, and \( K \) is the calibration matrix.

A plane \( \pi \) in the scene is mapped to an image by a homography. The homography can be represented by a \( 3 \times 3 \) matrix \( H_\pi \) determined up to scale. For corresponding points, in the plane \( \mathbf{x} \) and in the image \( \mathbf{x}' \), the mapping can be written \( \lambda \mathbf{x}' = H_\pi \mathbf{x} \), for some scale \( \lambda \). Without loss of generality a set of parallel planes may be assumed to have equations, \( z = k_i \). Then the homography from this plane to an image of the plane may by letting in \( X = [x \ y \ k_i \ 1]^T \) in (9.1) be expressed as,

\[
H_i = KR \begin{bmatrix}
1 & 0 & -t_1 \\
0 & 1 & -t_2 \\
0 & 0 & k_i - t_3
\end{bmatrix}.
\]

Here the camera used for the projection is \( P = KR(I \mid - t) \) with \( t = [t_1 \ t_2 \ t_3]^T \).

The purpose of the camera calibration is to determine the intrinsic parameters. To do this either you have to use metric or affine information about the scene, or you use prior information about the intrinsic parameters in a sequence of images, e.g. that some parameters are constant during the sequence.

In our application we work with a single plane in a single image. The camera has 11 parameters, 6 external and 5 internal. Since the metric of the plane is known, the homography (eight degrees of freedom) gives two constraints on the calibration cf. [65, 75]. When the equation of the plane has the above form these constraints can be expressed as,

\[
\mathbf{h}_1^T \mathbf{h}_1 - \mathbf{h}_2^T \mathbf{h}_2 = 0, \quad \mathbf{h}_1^T \mathbf{h}_1 = 0.
\]

Here \( \mathbf{h}_1 \) is the first column of the homography, \( \mathbf{h}_2 \) is the second column and \( \omega = K^{-T}K^{-1} \) is the image of the absolute conic.
9.3 System approach - 2D model

Associate a template corresponding to a dominant plane for each of the buildings. The template indicates where the dominant edges on this plane are located and we assume to know the metric in the plane. The template should further consist of mainly horizontal and vertical edges. A typical template may consist of windows and edges of the building. To the left in Figure 9.1 an image of a building is shown and to the right the corresponding template. We assume to have been given such templates to each building that we want to recognize in the image.

The task is now to recognize one of the buildings in the image and then to determine where the camera is located and how it is oriented.

9.3.1 Recognition

The first part of the problem is to determine which building we are viewing. This may be achieved by locating the windows on the building, e.g. by using the approach in the previous chapter. Projective invariants based on the window information may then be used in order to distinguish between buildings. If there is only a small number of possible buildings, the following matching procedure may be adopted to all of the templates and the one which has the optimal match is chosen.

9.3.2 Matching

The template corresponding to the building is matched to the image, i.e. the homography $H$ is estimated so that the template is mapped correctly to the image.
It is impractical to search in eight parameters in $H$ to find the best homography. We show that, with certain assumptions, the search can be reduced to a number of searches in two parameters. These can be performed efficiently with fast convolution.

The assumption that there are dominant edges in both the vertical and horizontal directions makes it suitable to decompose the homography into two parts. The first part transforms the image so that lines in the image, corresponding to horizontal/vertical lines in the dominant plane, are horizontal/vertical in the transformed image. Such transformation is sometimes called rectification and we denote it by $T$. This transformation has four degrees of freedom, but may be decomposed into two parts $T = T_x T_y$. The first one makes horizontal lines horizontal in the image, and the second one makes vertical lines vertical in the image. The second part of the homography determines the scale and translation parameters in the $x$- and $y$-directions. This transformation also has four degrees of freedom. But since the template mainly consists of horizontal and vertical edges the $x$ parameters may be separated from the $y$ parameters. The homography may consequently be decomposed as,

$$H = S_x S_y T_x T_y,$$

where

$$S_x = \begin{bmatrix} c_x & 0 & d_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_y & d_y \\ 0 & 0 & 1 \end{bmatrix},$$

and $T = T_x T_y$ is the rectifying homography. The homography $H$ is determined by estimating the components $S_x$, $S_y$, $T_x$ and $T_y$. 

Figure 9.2: Rectification of the image in Figure 9.1(left).
Rectification

The rectifying homography $T$ has four degrees of freedom, thus it is enough to know how two points are mapped in order to determine it. We know that the vanishing points for horizontal and vertical edges should be mapped to $[1 \ 0 \ 0]^T$ and $[0 \ 1 \ 0]^T$ respectively. If these can be determined the rectifying homography may be computed. It is also clear that the vanishing points for the horizontal direction determines $T_x$ and the vanishing points for the vertical direction determines $T_y$.

A number of approaches for determining vanishing points have been proposed, cf. [43, 56, 76] where earlier references also are given. In [43] it is suggested to use a Maximum Likelihood estimator in order to determine the vanishing points. In [76] the cascaded Hough transformation is used on the derivative of the image. Each point in the derivative of the image votes on a number of lines. After thresholding the lines, each line votes on a number of points, corresponding to possible intersection points.

When the two vanishing points are determined the image can be rectified by applying a homography, $T$, that maps the vanishing points to infinity in the horizontal and vertical directions. Figure 9.2 shows the rectification of the uncalibrated image in Figure 9.1.

If the camera calibration is known other constraints must be fulfilled as well. Firstly if $v_1$ and $v_2$ are the image coordinates compensated for the calibration of two orthogonal directions in the scene, then $v_1^T v_2 = 0$, cf. (2.3). In this case there are consequently only three degrees of freedom for the two vanishing points. Secondly, since the rectified image has the camera matrix $P_r = [I \ | -t]$, for the homography $T$ it holds that $T R [I \ | -t] = [I \ | -t]$. Then $T$ must be an orthogonal matrix. In order to determine $T$ when the vanishing points are known, the concept of quaternions may be used, cf. [20].

Translation and scale

There are four parameters left to be determined in the uncalibrated case; scale and translation in $x$ and $y$. We use the fact that there are mainly vertical and horizontal edges in the template and determine the scale and translation in $x$ independently of the scale and translation in $y$. It is possible to determine the scale and translation in $x$ when the horizontal rectification part $T_x$ is known but not the vertical rectification part $T_y$.

The columns of the thresholded derivative of a rectified image is added into a vector. Such a vector is shown in Figure 9.3. If the low frequency component is subtracted, the algorithm becomes more robust to unwanted derivatives, e.g. the trees in Figure 9.1. We want to match this to a similar plot derived from the template by finding the correct scale and translation. The corresponding template plot is shown in Figure 9.3.

One way to do this is to create a matrix where each row is a different scaling of the plot in Figure 9.3. The template plot is then correlated with the matrix to find the scale and translation in $x$. The correlation can be performed efficiently by the use of FFT. The same procedure is carried out on the rows. Figure 9.4 shows the correlation matrix for an experiment. Each element in the matrix encodes the correlation for a particular choice of
Figure 9.3: At the top a plot of the column sum of the derivative of the image in Figure 9.2. At the bottom the corresponding plot for the template is shown.
translation and scale. Figure 9.5 illustrates the best choice of translation and scale in the experiment.

When a number of local maxima for the $x$ and $y$ parameters has been determined, the next step is to find the best pair of $x$ and $y$ parameters. This is performed by, for all possible parameter combinations, estimating the number of pixels which is decided to be edge pixels in the template as well as in the derivative image. The parameter combination which has the greatest number of such pixels is chosen. The quotient between these two numbers of edge pixels may serve as a measure of quality for the match and may be used to decide if the current building is present in the image. The template may also have a number of pixels indicating where there should be no edges in the images. This corresponds to smooth areas on the building. The number of such pixels which are mapped to edge pixels in the image may also be used in the search for the best parameter combination.

In the calibrated case there are only three degrees of freedom left when the image has been rectified, one scale and translation in $x$- and $y$-direction. The same procedure as above may be applied, however when the parameter combinations are chosen the scale in $x$ and $y$ must be equal. Consider the vector formed by for each scale taking the best correlation among all possible translations, i.e. taking the row maximum in the correlation matrices. The scale is determined by adding two such vectors formed by the $x$ and $y$ direction and then finding the maximum. When the scale is determined the translation in $x$ is found as the maximum of the row in the correlation matrix corresponding to the
chosen scale, and the same procedure determines the translation in $y$.

### 9.3.3 Experiment 1

The matching algorithm has been tested on a number of different images with satisfying results. Figure 9.6 shows the results of experiments on uncalibrated images. In some cases templates indicating where both edges and smooth areas are located were used. The smooth areas are marked white in the template. For the case in the lower left of Figure 9.6, the algorithm failed to find the leftmost part of the building. The template used here was probably too simple to give a clear maximum.

### 9.3.4 Pose estimation

The estimated homography is now used to determine the position and orientation of the camera.

A homography, $H$, from a plane with known metric to an image impose the two constraints in (9.4) on the calibration matrix. From a calibrated camera it is well known that pose estimation is possible from a single homography. This was noted in Chapter 4. Hence, if we have a camera model with two unknown internal parameters, e.g. the camera constant and the aspect-ratio, it is possible to determine the position of the camera from a single homography. First the two intrinsic camera parameters are estimated using (9.4). Then, if the plane is assumed to have equation $z = 0$, the position, $t$, and orientation, $R$, of the camera may be recovered by a QR-decomposition of

$$K^{-1}H = R \begin{bmatrix} 1 & 0 & -t_1 \\ 0 & 1 & -t_2 \\ 0 & 0 & -t_3 \end{bmatrix}.$$
Figure 9.6: Results of the matching algorithm for six images. Black pixels correspond to edges and white pixels to smooth areas.
This decomposition is not unique. There are in general two solutions which correspond to optical centers on two sides of the plane. This is not a problem in practice since it is usually known which side of the plane that is visible.

In the presence of noise the QR-decomposition will not have the form above. In e.g. [67] techniques to find the optimal $\mathbf{R}$ and $\mathbf{t}$ are discussed.

9.3.5 Experiment 2

In a second experiment, 11 images of a scene from different views and at different times of the day were used. A template of a building was automatically matched to the images without any information about the calibration. The images are shown in Figure 9.7 with the best template match. The position and orientation of the camera was then calculated according to the above. For this part the calibration of the camera was used. The resulting reconstruction is shown from above in Figure 9.8. The asterisks are the estimated camera positions and the arrow shows the orientation. The rectangle at the bottom is the building. In order to validate the results, a reconstruction based on manually determined point correspondences and a standard structure and motion algorithm is shown as circles. As the two reconstructions are very similar, this indicates that the pose recovery is accurate.

9.4 System approach - 3D Model

In this section we consider the use of a 3D model. When a 3D model is used instead of a single plane the discrimination of buildings improves and the accuracy in the pose estimation increases. The matching process for a 3D template is now described.

9.4.1 Matching

Assume that the model is given as a number of templates as described above for a number of parallel planes and that the relative translations between the planes are known. As for the case of a single planar template, the image is first rectified. Since vertical/horizontal edges in parallel planes have the same vanishing point the rectification procedure is identical to that of the single plane case. The rectification in the horizontal direction may be separated from the rectification in the vertical direction also for this case.

The next step is to find the optimal match of the rectified image and the projected template. The projection of the template is achieved by scaling and translating each plane. However, when the scale and translation in e.g. the horizontal direction is chosen for one plane it is not clear how the templates in other planes should be scaled and translated. We know that when the scale and translation in $x$ and $y$ have been chosen, then the pose can be estimated and the homography for every plane may be computed. It will now be shown that if the scale and translation in $x$ but not in $y$ is known, it is possible to compute the scale and translation in $x$ for every other parallel plane. However, this is only possible
Figure 9.7: Images of a building from different viewpoints taken at different times of the day with the template matched to a building
when the calibration matrix has at the most one degree of freedom. We will also show that the best match may be found as the maximum of the sum of the correlation matrices, created from the different planes in the 3D model.

We assume that the image has been rectified and the unknown parameter in the internal calibration has been obtained from the equation \( \mathbf{v}_1^T \mathbf{K}^{-1} \mathbf{K}^{-1} \mathbf{v}_2 = 0 \) by the use of the vanishing points. We further assume that the given image is taken with a perspective camera. The camera matrix for the rectified image \( \mathbf{P}_r \) may be written as

\[
\mathbf{P}_r = \mathbf{T} \mathbf{K} \mathbf{R} [\mathbf{I} | -\mathbf{t}] = \mathbf{K}_r \mathbf{R}^T \mathbf{R} [\mathbf{I} | -\mathbf{t}] = \mathbf{K}_r [\mathbf{I} | -\mathbf{t}].
\]

Since \( \mathbf{K} \) and \( \mathbf{T} \) are known, the calibration matrix for the rectified image, \( \mathbf{K}_r \) may be calculated by a modified QR decomposition. It is now assumed that the coordinates have been compensated for the calibration, i.e. the matrix \( \mathbf{K}_r \) above is the identity.

In order to illustrate the matching procedure we show how it is performed in the case of a template consisting of two planes. It is straight-forward to generalize the procedure to a template consisting of an arbitrary number of planes. Figure 9.9 shows the current situation; a calibrated camera is viewing a simple model consisting of two planes. The objective is to find the homographies \( \mathbf{H}_1 \) and \( \mathbf{H}_2 \) for the corresponding templates to the image. The derivative of the image is shown to the left in Figure 9.10 and the templates are shown to the right.

We know from (9.3) that the homography for the plane \( \mathbf{z} = k_1 \) to the rectified image

Figure 9.8: The pose estimation from the matches in Figure 9.7 (asterisk) compared with a reconstruction based on manually estimated point correspondences (circles).
Figure 9.9: A calibrated camera is viewing a template model consisting of two parallel planes.

is given by

\[
H_i = \begin{bmatrix}
\frac{1}{k_i - t_3} & 0 & \frac{t_2}{k_i - t_3} \\
0 & \frac{1}{k_i - t_3} & \frac{t_2}{k_i - t_3} \\
0 & 0 & 1
\end{bmatrix}.
\] (9.5)

where \( t = [t_1, t_2, t_3]^T \) is the camera center. It turns out that it is more practical to search for the inverse homographies, i.e. the homographies from the image to the templates. These may be expressed as

\[
H_1^{-1} = \begin{bmatrix}
k_1 - t_3 & 0 & t_1 \\
0 & k_1 - t_3 & t_2 \\
0 & 0 & 1
\end{bmatrix}, \quad H_2^{-1} = \begin{bmatrix}
k_2 - t_3 & 0 & t_1 \\
0 & k_2 - t_3 & t_2 \\
0 & 0 & 1
\end{bmatrix}.
\]

By inspecting these matrices we find that for a particular choice of scale of the image to fit template one, \( c_x = k_1 - t_3 \), we may compute the scale of the image to fit template two, \( c'_x = k_2 - t_3 \), from

\[
c'_x = k_2 - k_1 + c_x.
\] (9.6)

Here \( k_1 \) and \( k_2 \) are the known equations of the planes. We also note that the translation part is the same for the two homographies.

In order to find the scale and translation in \( x \) for template one, we create a matrix in which each row is a scaling of the column sum of the derivative of the image. We also create such a matrix for template two, but for this matrix the scale for row \( i \) is calculated from (9.6), were \( c_{x,i} \) is the scale for row \( i \) in the first matrix. We then calculate the correlation with the corresponding template for these two matrices, cf. Figure 9.11.

The correlation matrices were created so that row \( i \) in the first correlation matrix is compatible with row \( i \) in the second, but what about the columns? Since the translation part of \( H_1^{-1} \) equals the translation part of \( H_2^{-1} \) the columns will also be compatible.
Consequently, pixels \((i,j)\) in the correlation matrices correspond, and we may simply add the correlation matrices from each template into a total correlation matrix, cf. the right of Figure 9.11. Each element in this matrix corresponds to geometrical valid choices of scales and translations of the templates.

When a number of \(x\)- and \(y\)-parameter candidates are determined, the same procedure as for the single plane case is performed in order to find the best combination of \(x\)- and \(y\)-parameters. Finally the pose of the camera may be computed.

### 9.4.2 Experiment 3

A model consisting of two planes was matched to three images according to the above. The result is shown in Figure 9.12. Here there are great differences in lighting and minor occlusions.

### 9.5 Conclusions

We have presented an automatic system for pose estimation from a single image in a city scene using a 2D or a 3D model. It is difficult to validate the system objectively, but in a number of experiments the system performs well. It handles different lighting as well as minor occlusions reasonably.
Figure 9.11: To the left is the correlation between the image and template one shown for different scales and translations in $\mathcal{Z}$. In the center is the correlation between the image and template two shown. The sum of these are shown to the right. The $x$-axis corresponds to different scales and the $y$-axis correspond to different translations.

Figure 9.12: A two-plane model matched to three images of a building.
Chapter 10

Conclusions

In this thesis a number of computer vision problems has been discussed. The focus has been on using complex features to determine the geometry of the cameras and the scene.

In particular, the problem of generating images of an object or a scene from arbitrary viewpoints using information from a fixed number of images, has been studied. This problem is referred to as view synthesis. According to the project plan, the work was at the start focused on piecewise planar scenes with applications on buildings in mind. It was then continued to more complex scenes, and a system that in experimental studies handle e.g. flowers and faces reasonable, was developed.

In many approaches to view synthesis, e.g. the last mentioned, the camera geometry is determined by identifying points corresponding in several images. In the thesis a new type of feature, quiver, was introduced. The advantages of using these instead of points were discussed.

The aim for view synthesis is, given a number of images of a scene, to generate photorealistic images from new view points. In my opinion this is very difficult without a priori information about the scene. For applications with high requirements on the quality of the generated images, I believe that the use of a model based approach would be most successful. Detailed 3D models of objects are adapted to the images manually or automatically. For the building reconstructions in the thesis it would be interesting to incorporate models of windows, doors etc. This would surely improve the quality in the generated images.

A model based approach to visualize the contents in a refrigerator was developed in a project in collaboration with the company Electrolux. A prototype of the system was implemented and connected to a fridge equipped with cameras. Since the objective was to implement a fully automatic system in a time limited project, simple solutions had to be chosen for many of the image analysis problems. It would be interesting to see the performance of the system if it was implemented for commercial use. However, the concept of the intelligent home was closed down at Electrolux, so I guess there is little chance we will see intelligent refrigerators in a near feature.

For future view synthesis algorithms I believe that object recognition will have a central role. Instead of having a priori information about what kind of objects are in the scene, a number of objects may be recognized in the images. For the application to reconstruct buildings a system for detecting windows based on support vector machines was implemented. Our next step would be to incorporate this system in the reconstruction algorithm.

A system for detecting windows, also has applications to building recognition and pose estimation. These applications are discussed in the thesis, and a number of examples
are shown.
Appendix A

A.1 Support Vector Machines

Support Vector Machines (SVM) was introduced by Vapnik already in 1979 although it has been given a lot of attention just recently. This section gives an introduction to SVM for pattern recognition. The objective is to create a learning machine that responds binary to test data. In order to train the machine we assume to have a set of training data and the correct response to the data.

A.1.1 Linear SVM

Denote the training data

\[ \{x_i, y_i\}, \quad i = 1, \ldots, l, \]

where \( x_i \in \mathbb{R}^d \) are the input measures and \( y_i \in \{-1, 1\} \) is the binary response. Suppose that there exists a hyperplane,

\[ x_i \cdot w + b = 0, \]

with parameters \((w, b)\), separating the positive from the negative examples, cf. Figure A.1. Let \( d_+(d_-) \) be the shortest distance from the hyperplane to the closest positive (negative) point. The margin of the hyperplane is defined as the minimum of \( d_+ \) and \( d_- \). In the linear case the support vector algorithm looks for the separating hyperplane with the largest margin.

The constraints that the hyperplane must separate the points,

\[ x_i \cdot w + b \geq 1, \quad \text{for } y_i = +1, \]

\[ x_i \cdot w + b \leq -1, \quad \text{for } y_i = -1, \]

may be combined into one set of linear inequalities

\[ y_i(x_i \cdot w + b) - 1 \geq 0, \quad \forall i. \tag{A.1} \]

Now consider points which lie on the hyperplane \( H_1 : x_i \cdot w + b = 1 \) with perpendicular distance from the origin \(|1 - b|/||w||\), and the points which lie the a hyperplane \( H_2 : x_i \cdot w + b = -1 \) with perpendicular distance from the origin \(|-1 - b|/||w||\). Accordingly the margin is \( 1/||w|| \). The hyperplane that gives the maximal margin is found by minimizing \( ||w|| \) subject to (A.1). This is a convex quadratic programming problem.

In order to solve this optimization problem it is useful to study the dual optimization problem. Introduce positive Lagrange multipliers \( \alpha_i, \quad i = 1 \ldots l \), one for each inequality
constraint in (A.1). This gives the Lagrangian

\[ L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i. \]  

(A.2)

This function is minimized with respect to \( w \) and \( b \), and simultaneously it is required that the derivatives of \( L_p \) with respect to all the \( \alpha_i \) vanish. This is a convex quadratic programming problem and we may equivalently solve the dual problem to maximize \( L_p \) subject to the constraints that the gradient of \( L_p \) with respect to \( w \) and \( b \) vanish. These constraints give that

\[ w = \sum_i \alpha_i y_i x_i, \]
\[ \sum_i \alpha_i y_i = 0. \]

By substituting these in (A.2) we get

\[ L_d = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j. \]  

(A.3)

Support vector training aims at maximizing \( L_d \) with respect to \( \alpha_i \) subject to the constraints \( \sum \alpha_i y_i = 0 \) and \( \alpha_i \geq 0 \). There are efficient methods to solve convex quadratic programming problems. An important property of strictly convex quadratic programming problems is that every such problem has a unique global optimal value and that every local solution is also global.
A.1. SUPPORT VECTOR MACHINES

Note that while $\mathbf{w}$ is explicitly determined in the optimization the threshold $b$ is not. However, $b$ is easily found by the Karush-Kuhn-Tucker (KKT) conditions. According to these we have

$$\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0.$$  

Any $i$ for which $\alpha_i > 0$ may be used to determine $b$.

For every training point $\mathbf{x}_i$ there is an $\alpha_i$. The points for which the corresponding $\alpha_i > 0$ in the solution are called support vectors. These points lie on one of the hyperplanes $H_1$ and $H_2$. The support vectors are the critical elements in the SVM. Only these affect the decision of the machine.

Once the SVM has been trained it may be used to classify new test data. We simply decide on which side of the separating hyperplane the test data is, i.e. we take the class of $\mathbf{x}$ to be $\text{sign}(f(\mathbf{x}))$ where $f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$.

A.1.2 Non linear SVM

The previous method may be generalized to a non linear decision function. Using a function $\Phi$, the data is mapped to another Euclidean space, $\mathbf{H}$, in which the data is linearly separable.

$$\Phi : \mathbb{R}^d \rightarrow \mathbf{H}.$$  

The separating hyperplane may now be found according to the above. Note that the only way in which the data appear in the training is in the form of scalar products. Hence the function $\Phi$ need not to be known, but only the kernel function $K$, such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$.

In the test phase the sign of the function $f(\mathbf{x})$ is used to classify the data. Here

$$f(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(s_i) \cdot \Phi(\mathbf{x}) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, \mathbf{x}) + b$$

and $s_i$ are the support vectors. Thus, again we avoid computing $\Phi(\mathbf{x})$ explicitly and use the kernel $K$ instead. This makes it possible to use a kernel that corresponds to a mapping of the input data to a Euclidean space of infinite dimension.

131
Bibliography


BIBLIOGRAPHY


[52] M. Pollefeys, R. Koch, M. Vergauwen, B. Devernay, and L. Van Gool. Three-
dimensional scene reconstruction from images. *proceedings SPIE Electronic Imaging,*  

using projective shape representation. In *Proc. Int. Conf. on Computer Vision,* pages  

for planar object recognition. In *Proc. 2nd European Conf. on Computer Vision,*  
*Santa Margherita Ligure, Italy*, pages 757–772, 1992.

library access by projectively invariant indexing functions. In *Conf. Computer Vision  

[56] F. Schaffalitzky and A. Zisserman. Planar grouping for automatic detection of van-


1996.


Falls Church, VA, 1980.

[62] G. Sparr. A common framework for kinetic depth, motion and reconstruction. In  
in Computer Science*, pages 471–482. Springer Verlag, Lecture Notes in Computer  

[63] G. Sparr. Simultaneous reconstruction of scene structure and camera locations from  
uncalibrated image sequences. In *Int. Conf. Pattern Recognition*, volume I, pages  
328–333, Vienna, Austria, 1996.


137


Index

3D Reconstruction, 49

absolute conic, 22

affine camera, 24
part, 19
plane, 20
space, 22
transformation group, 17, 19, 21, 22
aspect ratio, 25
auto-calibration, 26

Bezout, 73
bundle adjustment, 33

calibrated camera, 25
camera calibration, 25
center, 23
cartesian coordinates, 15
Cartesian coordinates, 15
circular points, 21
collineation, 18
conic, 16, 20
correspondence problem, 29
cross ratio, 17, 19
dual conic, 21
epipolar line, 29
epipolar constraints, 27
epipolar plane, 27
epipole, 25, 29
essential matrix, 28
Euclidean reconstruction, 52
euclidean transformation, 21, 22
transformation group, 16
extrinsic parameters, 25
focal length, 23
footprint, 95, 96
fundamental matrix, 28
generalized eigenvalues, 35
generalized eigenvectors, 35
Grassmann coordinates, 22
Harris corner detector, 30
homogeneous coordinates, 18
homogeneous vectors, 15
homography, 18, 27
hyperplane, 18
image based rendering, 8, 81
image plane, 23
intrinsic camera parameters, 25
invariants, 16
Jordan canonical form, 39
kinetic depth, 37
line, 15, 20
line at infinity, 20
linear transformation, 16, 18
minimal case, 8, 65
mixed volume, 73
morphing, 81
motion detector, 95
non-linear distortion, 24
normalised coordinate system, 23
object recognition, 8
parallel, 19
parallel projection, 24
permutation symbol, 31
pinhole camera, 23
Plücker coordinates, 22
plane, 8, 18
plane at infinity, 19
plane at infinity, 22
point, 15, 18
points at infinity, 19
pose estimation, 8
principal point, 25
principle of duality, 18
projection, 19
projective
basis, 18
line, 19
plane, 20
space, 18, 21
transformation, 18
transformation group, 17
projectivity, 18
quadric, 16
quadrifocal tensor, 33
quadrifocal constraints, 33
quiver, 9
quivers, 65
RANSAC, 29
rectangle, 9
reprojection error, 34
rich features, 8
rigid motion, 16
silhouette, 9
similarity transformation, 21, 22
single view reconstruction, 52
skew, 25
standard projective basis, 18
structure and motion, 8
support vector machines, 105
template, 9
trifocal constraints, 31
trifocal tensor, 31
vanishing point, 59
view interpolation, 81
view synthesis, 8, 35, 81
visibility, 42
voxel, 81
wide baseline matching, 29