MANUAL AND AUTO CALIBRATION OF STEREO CAMERA SYSTEMS

MUSTAFA ÖZUYSAL

AUGUST 2004

MANUAL AND AUTO CALIBRATION OF STEREO CAMERA SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA ÖZUYSAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

AUGUST 2004

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Mübeccel Demirekler
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Uğur Halıcı
Supervisor

Examining Committee Members

Prof. Dr. Kemal Leblebicioğlu	(METU,EE)	
Prof. Dr. Uğur Halıcı	(METU,EE)	
Prof. Dr. Turgut Tümer	(METU,ME)	
Assoc. Prof. A. Aydın Alatan	(METU,EE)	
Dr. Uğur M. Leloğlu	(BİLTEN)	

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Mustafa Özuysal

Signature :

Abstract

MANUAL AND AUTO CALIBRATION OF STEREO CAMERA SYSTEMS

Özuysal, Mustafa
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Uğur Halıcı

August 2004, 90 pages

To make three dimensional measurements using a stereo camera system, the intrinsic and extrinsic calibration of the system should be obtained. Furthermore, to allow zooming, intrinsic parameters should be re-estimated using only scene constraints. In this study both manual and autocalibration algorithms are implemented and tested. The implemented manual calibration system is used to calculate the parameters of the calibration with the help of a planar calibration object. The method is tested on different internal calibration settings and results of 3D measurements using the obtained calibration is presented. Two autocalibration methods have been implemented. The first one requires a general motion while the second method requires a pure rotation of the cameras.

The autocalibration methods require point matches between images. To achieve a fully automated process, robust algorithms for point matching have been implemented. For the case of general motion the fundamental matrix relation is used in the matching algorithm. When there is only rotation between views, the homography relation is used. The results of variations on the autocalibration methods are also presented.

The result of the manual calibration has been found to be very reliable. The results of

the first autocalibration method are not accurate enough but it has been shown that the calibration from rotating cameras performs precise enough if rotation between images is sufficiently large.

Keywords: camera calibration, autocalibration, stereo camera system.

Öz

STEREO KAMERA SİSTEMLERİNİN ELLE VE ÖZDEVİMLİ KALİBRASYONU

Özuysal, Mustafa
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Ağustos 2004, 90 sayfa

Stereo kamera sistemleri kullanılarak üç boyutlu ölçüm yapılabilmesi için kamera sisteminin iç ve dış parametrelerinin kalibre edilmesi gerekmektedir. Ayrıca kameralardaki yakınlaştırma özelliğinin kullanılabilmesi için iç parametrelerin ortama bağlı kısıtlamalar kullanılarak tekrar hesaplanması gerekir. Bu çalışmada hem elle hem de özdevimli kalibrasyon yöntemleri hazırlanıp test edilmiştir. Hazırlanan elle kalibrasyon yöntemi, düzlemsel bir kalibrasyon nesnesi yardımıyla kalibrasyon parametrelerinin hesaplanması için kullanılmıştır. Hazırlanan sistem, farklı kamera ayarlarıyla test edilmiş ve hesaplanan parametreler kullanılarak yapılan üç boyutlu ölçüm sonuçları verilmiştir. İki özdevimli kalibrasyon yöntemi denenmiştir. Bunlardan ilki genel bir hareket için geçerli olup ikincisi ise kameraların sadece dönme hareketi yapmasını gerektirmektedir.

Özdevimli kalibrasyon yöntemleri resimler arasında eşleştirilmiş noktaları kullanır. Tamamen otomatik çalışan bir sistem elde edilebilmesi için nokta eşleştirmelerini elde eden, hatalara dayanıklı yöntemler kullanılmıştır. Nokta eşleştirmede genel hareketler için temel matris ilişkileri, sadece dönme hareketi olan görüntülerde ise düzlemsel dönüşüm ilişkileri kullanılmıştır. Özdevimli kalibrasyon yöntemleri üzerinde farklılıklar yaratılarak bunların sonuçları sunulmuştur.

Elle kalibrasyon sonuçlarının oldukça güvenilir olduğu gözlenmiştir. İlk özdevimli kalibrasyon yönteminin sonuçları yeterli doğrulukta olmamakla birlikte, eğer görüntüler arasındaki dönme açısı yeterliyse dönen kameralardan kalibrasyon sonuçlarının istenen hassasiyette olduğu gösterilmiştir.

Anahtar Kelimeler: kamera kalibrasyonu, özdevimli kalibrasyon, stereo kamera sistemi.

To My Family

Acknowledgments

I would like to thank Prof. Dr. Uğur Halıcı and Prof. Dr. Kemal Leblebicioğlu for the warm and comfortable laboratory environment and also for the guidance that they have provided. I would like to thank all of the members of the Computer Vision and Artificial Intelligence Research Group, in particular Dr. İlkay Ulusoy, Zafer Arıcan and Erdem Akagündüz who has provided invaluable feedback, support and at times the necessary motivation for the completion of this thesis study.

I would like to express my gratitude to Assoc. Prof. Aydın Alatan for introducing me into the field of "Uncalibrated Vision" at the first place.

I would like to acknowledge Uğur Ömür from the Mathematics Department of Middle East Technical University for providing the LaTeX 2_ε style file necessary in the preparation of this document.

Finally I would like to thank my parents, my brother and my beloved one for their endless support, patience and trust.

This thesis is partially supported by BAP Projects, BAP-2002-03-01-05 ("Active Robot Vision") and BAP.04.03.01.02.

Table of Contents

APPENDICES

List of Tables

List of Figures

List of Symbols

$\mathbf{x}$	A 2D homogeneous point (3×1)
$\mathbf{X}$	A 3D homogeneous point (4×1)
$\mathbf{l}$	A 2D line (3×1)
$\mathbf{\Pi}$	A 3D plane (4×1)
$\mathtt{C}$	A conic in 2D (3×3)
$\mathtt{C}^*$	A dual conic in 2D (3×3)
$\mathtt{Q}$	A quadric in 3D (4×4)
$\mathtt{Q}^*$	A dual quadric in 3D (4×4)
$\mathbf{l}_\infty$	The Line at Infinity (3×1)
$\mathbf{\Pi}_\infty$	The Plane at Infinity (4×1)
Ω_∞	The Absolute Conic
$\mathtt{Q}^*_\infty$	The Absolute Dual Quadric (4×4)
ω	Image of the Absolute Conic (3×3)
ω^*	Dual Image of the Absolute Conic (3×3)
$\mathtt{F}$	Fundamental Matrix (3×3)
$\mathtt{H}$	Homography Matrix (3×3)
$\mathtt{P}$	Camera Projection Matrix (3×4)
$\mathtt{K}$	Calibration Matrix (3×3)
f_x	Focal length in x direction
f_y	Focal length in y direction
s	Skew
u_0	x coordinate of the principal point
v_0	y coordinate of the principal point
κ_1	First radial distortion coefficient
κ_2	Second radial distortion coefficient
$\mathtt{A}^{-\mathsf{T}}$	Transposed inverse of the matrix $\mathtt{A}$

Chapter 1

Introduction

1.1 Motivation and Problem Definition

In this thesis, the aim is to develop a stereo camera system which can be calibrated manually with good precision and also can maintain its calibrated state in the case of zooming using autocalibration techniques.

The developed system can be used in visual guidance of robotic systems, surveillance systems as well as in map building tasks. Such systems will generally require 3D measurements and sometimes the flexibility of zooming. To be able to obtain 3D measurements, the internal and external parameters of the camera system should be found. The internal parameters define how a 3D point is imaged on the 2D imaging surface. The most notable internal parameter is the focal length and is a measure of the magnification of the camera. The focal length should be known to good precision in order to obtain reliable measurements. The external parameters refer to the 3D relation between the position of the cameras with respect to each other and possibly with respect to another coordinate frame. To be able to make any measurements, the external parameters should be calculated.

Although the parameters of calibration can be calculated before the system is started, not all of these parameters are constant during operation. Most notably, when the camera is zoomed, the focal length changes as well as other internal parameters. One alternative is to stop the process and calibrate the camera using the manual calibration techniques. However, this is not flexible and even impossible in some situations. Recent developments in the uncalibrated vision area allows the computation of internal parameters without the use of a calibration object with known 3D structure.

These methods only require the rigidity of the scene and by placing constraints on 3D projective objects, generally over an image sequence, the internal parameters are recovered.

The following general constraints are considered in the design of the system. The manual calibration should be easily performed and the calibration object should allow simple construction to achieve higher precision. The autocalibration process should not take too much time and should be completely automated.

The above requirements are due to the fact that the system will be used in robot navigation tasks. An example of such a simulated system is given by Ulusoy et al [31]. This simulated system is also being tested on a real robot platform. For localization studies using the vision system constructed in this study, the reader may refer to [2]. The vision system will be used to obtain three dimensional data from the environment. Hence, the performance measure for the developed system is the measured 3D lengths and angles. The calibration system can be used on a variety of stereo systems for different scenarios.

1.2 Literature on Manual Calibration

The manual calibration methods are based on the knowledge of three dimensional structure of a calibration object. This knowledge is then used to find the internal and external calibration of the camera system.

The manual calibration of camera systems is a well studied problem. The earlier work is done by researchers in the field of photogrammetry. A seminal paper on the subject providing a closed form solution to the problem is written by Tsai [30]. The method requires a 3D calibration object and begins the computation by using a closed form expression for the rotation and the first two components of the translation. The main difficulty with the method is that it can not calculate the aspect ratio of the pixels if a planar object is used for calibration [20]. Therefore a calibration object spanning a volume should be used to recover the aspect ratio. The construction of such a calibration object provides difficulties.

A more recent work is done by Heikkilä and Silven [19]. The presented approach emphasizes both the feature extraction and solution steps. The method is also implemented as a Matlab toolbox in the public domain available through Internet.

The method used in this study is due to Zhang [33]. The method uses a planar calibration pattern and by calculating the relation between 3D and imaged corners, constraints are formed on the internal calibration. The external calibration is then easily achieved.

Although the methods above use point correspondences between the calibration pattern and the images, there are other alternatives. A method that uses images of circles to calculate the internal parameters is presented by Agrawal and Davis [1]. It is shown that the projection of the circle is a conic on the imaging plane and the dual to this conic is directly related to the internal calibration. Then a semi-definite programming (SDP) algorithm is used to achieve calibration.

1.3 Literature on Autocalibration

The autocalibration algorithms try to obtain the calibration information using only the rigidity of the scene. Generally, projective objects that are constant for a specific class of transformations are located by placing constraints on their representation. These objects are then used to obtain calibration parameters. The first autocalibration methods make use of the Kruppa equations. The Kruppa equations are quadratic constraints on epipolar lines that are tangent to a conic [18]. Although Kruppa equations are the traditional route to autocalibration, they have several drawbacks. The solution of the Kruppa equations have ambiguities due to the formulation of the problem [18].

The Kruppa equations solve for the Euclidean components directly. An alternative route is provided by stratified approaches. Using these methods, one achieves a projective reconstruction and then proceeds to recover affine and Euclidean structure. It has been shown that it is possible to achieve Euclidean structure from a projective reconstruction by using scene constraints only [14]. The affine calibration has been shown to be difficult to achieve. The method presented in [24] recovers the affine structure first by locating the plane at infinity using the modulus constraint. Then the Euclidean structure is restored by using constraints on the absolute conic.

There are other methods which use a more direct approach and achieve Euclidean structure right after the projective one. The method presented by Pollefeys and Van Gool uses constraints on the absolute dual quadric to recover Euclidean information [23]. The method can also work with varying intrinsic parameters.

The above methods use a projective structure to start with. The projective camera matrices can be obtained from the computed fundamental matrices [18]. An optimal way to achieve projective reconstruction of points is given by Hartley and Sturm [17].

The autocalibration methods generally assume that the motion in the sequence is general enough so that the constraints lead to a unique solution for the calibration. However, when the motion is not general enough, there are more than one, generally infinite solutions for intrinsic parameters. A classification of such critical motion sequences in the case of constant calibration parameters has been made by Sturm [25].

Aside from the general autocalibration problem, there has been research on particular cases. A method for affine calibration of a mobile platform making a planar motion is shown by Beardsley and Zisserman [3]. A closed-form formula to achieve calibration of a mobile device both in the general motion and planar motion cases is given by Csurka et al [6]. The method uses real Jordan factorizations of a 3D homography between projective reconstructions achieved by a moving stereo system. It is also possible to autocalibrate cameras if homographies between images of the ground plane can be computed [21].

Given some constraints on the internal parameters it is also possible to calibrate the camera directly using the form of the recovered fundamental matrix. One such method is presented by Brooks et al [4].

It is also possible to perform autocalibration from purely rotating cameras. For the case of constant calibration a method is given by Hartley [15] and the extended version to the case of varying intrinsic parameters is presented by Agapito, Hartley and Hayman [7].

1.4 Methods Used

The system is designed in the Computer Vision and Artificial Intelligence Laboratory of the Electrical and Electronics Engineering Department of the Middle East Technical University. Two analog cameras are attached to a planar platform almost in parallel form. The video signals from the cameras are then connected to the frame grabber card installed on a personal computer.

The manual calibration method uses a planar calibration object. The stereo images

of the calibration object is taken by capturing single frames with the frame grabber. The calibration object is composed of a checker board pattern printed on A4 paper and attached to a planar surface. The corners of the calibration pattern squares are then extracted automatically using OpenCV C++ Library functions. The four extreme corners are then selected by the user in each image to allow for a consistent ordering of the corner points. The calibration software then processes the ordered corner lists to produce the internal calibration. The algorithm is due to Zhang [33].

The first autocalibration method is the absolute dual quadric method of Pollefeys and Van Gool[23]. The projective reconstruction is achieved by robust fundamental matrix computation over the image sequences. Then the linear algorithm processes the obtained projective camera matrices to place constraints on the internal parameters.

The second autocalibration method uses pure rotations of the camera. Different versions of the algorithm are implemented [7, 15] . The homography relation between views are computed in a robust manner much like the fundamental matrix computation. Then the calculated homograhies are used to find calibration parameters. Since the method requires pure rotations, pan-tilt-zoom cameras are used to capture image sequences. The camera rotations and zoom are controlled using a C++ software library written by Zafer Arıcan by a serial port connection.

1.5 Organization of the Thesis

The organization of the thesis is as follows:

- Chapter 2 gives the necessary mathematical background. Some of the algorithms as well as the camera model are detailed.
- In Chapter 3, implemented manual calibration method, the computer setup used and test results are given.
- Chapter 4 details the implementation of an autocalibration method based on the estimation of the Absolute Dual Quadric. Robust point matching using RANSAC algorithm, based on fundamental matrix estimation is also explained. The test results are given at the end of the chapter.
- Chapter 5 presents another autocalibration method based on pure rotations of a camera. The robust point matching is implemented using RANSAC based on

homography estimation. The results are given at the end of the chapter with a discussion of the method.

- Chapter 6 begins with a summary of the obtained results in this study. Then a discussion and possible future enhancements concludes the chapter.

Chapter 2

Theoretical Background

2.1 Introduction

In this chapter, the necessary mathematical background is presented. Understanding these basic mathematical concepts is necessary to follow the derivations presented in Chapters 3, 4 and 5. Section 2.2 presents elements of 2D and 3D projective geometry. The representation of basic geometric entities as well as particular geometric objects are described. The concept of duality is also introduced. Section 2.3 describes the projective, affine, similarity and Euclidean classes of transformations of three dimensional space. For each class the associated invariants and the form of the transformation matrices are given. Section 2.4 presents the camera model used. Any deviations from this model are explained in the relevant chapters. The section ends with "two view relations". With two views the correspondence problem becomes more constrained. The related mathematical concepts are presented such as the "Fundamental Matrix" and "Epipolar Lines". Section 2.5 presents methods to compute the fundamental matrix. In this section, the normalization of data points is also introduced. This is required for good numerical conditioning of the problem. Section 2.6 presents a way to triangulate rays in order to infer 3D information from 2D image pixels using the camera model developed. Section 2.7 shows that when special conditions occur, the relation between images is a simple homography. Also an algorithm to compute this homography is given.

For a more complete discussion of projective geometry in computer vision, the reader should refer to the background chapters of [9] or [18]. References for particular elements are given in the related sections.

2.2 Projective Geometry

This section will present the two and three dimensional projective entities. In the representation of these geometric entities, homogeneous coordinates will be used. The homogeneous coordinates of a 2D point with Cartesian coordinates $(x/z, y/z)$ is denoted as follows

$$\mathbf{x} = [x\ y\ z]^\mathsf{T}.$$

The advantage of the homogeneous representation is its ability to treat infinite points in the same way as finite ones. The points with $z = 0$ represent *points at infinity*.

In the homogeneous representation, multiplication by a non-zero scalar does not change the Cartesian coordinates of the point hence $k \cdot \mathbf{x} \equiv \mathbf{x}$. This is true for all homogeneous quantities and the sign "$\approx$" will be used to denote equality up to scale.

The line $ax + by + c = 0$ is represented as $\mathbf{l} = [a\ b\ c]^\mathsf{T}$. The line equation can be written as $\mathbf{x}^\mathsf{T}\mathbf{l} = [x\ y\ 1]\,[a\ b\ c]^\mathsf{T} = 0$. Following these conventions, the line joining two points $\mathbf{x}$ and $\mathbf{x}'$ is represented by $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$. The intersection point of two lines $\mathbf{l}$ and $\mathbf{l}'$ is $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$. The cross product of two 3-vectors is given as

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}.$$

In the above formulas it can be seen that the role of points and lines are interchangeable. This is known as the *duality principle* and points and lines are called *the dual* of each other.

A conic in two dimensions can be written as $\mathbf{x}^\mathsf{T}\mathtt{C}\mathbf{x} = 0$ where $\mathtt{C}$ is a 3×3 symmetric matrix. The same conic equation can be written in terms of lines tangent to the conic as $\mathbf{l}^\mathsf{T}\mathtt{C}^*\mathbf{l} = 0$ where $\mathtt{C}^*$ is a 3×3 symmetric matrix and is called *the dual conic*. In the case where $\mathtt{C}$ is invertible $\mathtt{C}^* = \mathtt{C}^{-1}$.

A projective transformation of 2D preserves the collinearity of three points and is represented by an invertible 3×3 matrix $\mathtt{H}$. The points can be transformed by $\mathbf{x}' \approx \mathtt{H}\mathbf{x}$. To preserve collinearity the lines are transformed by $\mathbf{l}' \approx \mathtt{H}^{-\mathsf{T}}\mathbf{l}$ and conics are transformed by $\mathtt{C}' \approx \mathtt{H}^{-\mathsf{T}}\mathtt{C}\mathtt{H}^{-1}$. A dual conic transforms as $\mathtt{C}^{*\prime} \approx \mathtt{H}\mathtt{C}^*\mathtt{H}^\mathsf{T}$. The notation $\mathtt{H}^{-\mathsf{T}}$ is used to denote $(\mathtt{H}^{-1})^\mathsf{T} = (\mathtt{H}^\mathsf{T})^{-1}$.

In two dimensions, objects of particular interest are *the line at infinity* and *the circular points*. The line at infinity is represented by $\mathbf{l}_\infty = [0\ 0\ 1]^\mathsf{T}$. It is easy to see that if

$\mathbf{x}$ is a point at infinity then it lies on $\mathbf{l}_\infty$. Circular points are two complex point on the line at infinity with homogeneous coordinates $\mathtt{I} = [1\ i\ 0]^\mathsf{T}$ and $\mathtt{J} = [1\ -i\ 0]^\mathsf{T}$. These objects are invariants of affine and similarity transformations of 2D space respectively. Groups of transformations and their invariants (including affine and similarity transformations) are explained in Section 2.3 for three dimensional space.

In three dimensions a point can be represented by a 4-vector as $\mathbf{X} = [X\ Y\ Z\ W]^\mathsf{T}$ which has Cartesian coordinates $(X/W, Y/W, Z/W)$. As in the case of two dimensions the points with $W = 0$ represent points at infinity and multiplying a point vector by a non-zero scalar does not change the point it represents, $k \cdot \mathbf{X} \approx \mathbf{X}$.

The plane in three dimensions with equation $aX + bY + cZ + d = 0$ is represented by a 4-vector $\mathbf{\Pi} = [a\ b\ c\ d]^\mathsf{T}$. The plane equation can be written as $\mathbf{X}^\mathsf{T}\mathbf{\Pi} = [X\ Y\ Z\ 1]\,[a\ b\ c\ d]^\mathsf{T} = 0$. In three dimensions points and planes are dual to each other.

In three dimensions the representation for lines is more complicated than the two dimensional case. The reader may refer to Section 2.2.2 of [18] for various representations of lines in three dimensions.

A *quadric* equation in three dimensions can be written as $\mathbf{X}^\mathsf{T}\mathtt{Q}\mathbf{X} = 0$, where $\mathtt{Q}$ is a 4×4 symmetric matrix. The same equation can be written in terms of planes tangent to the quadric as $\mathbf{\Pi}^\mathsf{T}\mathtt{Q}^*\mathbf{\Pi} = 0$, where $\mathtt{Q}^*$ is a 4×4 symmetric matrix and is called *the dual quadric*. As in the case of conics of two dimension, if $\mathtt{Q}$ is invertible then $\mathtt{Q}^* = \mathtt{Q}^{-1}$.

Table 2.1: Transformation rules for three dimensional objects

Object	**Transformed Object**
Point	$\mathbf{X}' = \mathtt{T}\mathbf{X}$
Plane	$\mathbf{\Pi}' = \mathtt{T}^{-\mathsf{T}}\mathbf{\Pi}$
Quadric	$\mathtt{Q}' = \mathtt{T}^{-\mathsf{T}}\mathtt{Q}\mathtt{T}^{-1}$
Dual Quadric	$\mathtt{Q}^{*\prime} = \mathtt{T}\mathtt{Q}^*\mathtt{T}^\mathsf{T}$

A projective transformation of three dimensions is represented by a 4×4 invertible matrix $\mathtt{T}$. Table 2.1 shows the transformation rules for various three dimensional objects [18].

In three dimensions there are three objects of particular interest. The plane represented by $\mathbf{\Pi}_\infty = [0\ 0\ 0\ 1]^\mathsf{T}$ is called *the plane at infinity*. It is easy to see that all points at infinity are on this plane. The points on this plane correspond to directions in 3D. All lines parallel to each other intersect at a unique point on this plane. Hence any transformation that preserves parallelism of lines should preserve $\mathbf{\Pi}_\infty$.

The second object is called *the absolute conic* and represented by Ω_∞. Ω_∞ is a point conic on $\mathbf{\Pi}_\infty$ with $\mathtt{C} = \mathtt{I}_{4\times4}$ and is composed of only imaginary points. The importance of Ω_∞ is that it can be used to measure angles (See [18] for more information). Hence if a transformation changes Ω_∞ then there is no trivial way to measure angles. The image of Ω_∞, ω is also closely coupled with intrinsic camera calibration (See Section 2.4).

The third object is *the absolute dual quadric*, denoted by $\mathtt{Q}^*_\infty$ which is dual to Ω_∞. In a metric frame, $\mathtt{Q}^*_\infty$ is represented by the following 4×4 homogeneous matrix

$$\mathtt{Q}^*_\infty = \begin{bmatrix} \mathtt{I}_{3\times3} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix}.$$

$\mathtt{Q}^*_\infty$ combines the information encoded by both Ω_∞ and $\mathbf{\Pi}_\infty$. The use of the absolute quadric in autocalibration is due to Triggs [29].

2.3 Stratification of 3D Space

In this section transformations in three dimensions are divided into classes. This classification helps in the understanding of projective concepts and in the development of new algorithms. This classification has been introduced to the computer vision literature by Faugeras [8].

With each class of transformations there are a number of associated invariant properties which do not change under a transformation of the respective class. These invariants have helped the development of stratified approaches to many problems in computer vision. These stratified approaches start with the most general class and then restrict the solution by recovering invariants of the more restrictive classes while updating the geometry of the problem.

The most general class of transformations is *the group of projective transformations*. As stated in Section 2.2, projective transformations preserve the collinearity of points. Projective transformations of 3D are represented by 4×4 homogeneous matrices which

are invertible. As in the case of homogeneous vectors, homogeneous matrices have arbitrary scale such that $k \cdot \mathtt{T} = \mathtt{T}$. The matrix entries are not constrained. Since projective group is the most general case, it has only a few invariants. Most notably projective transformations preserve intersection, tangency and *the cross ratio*. The cross ratio is a ratio of ratios. The cross ratio of four collinear points is defined to be

$$\mathrm{Cr}(\mathbf{x}_1, \mathbf{x}_2; \mathbf{x}_3, \mathbf{x}_4) = \frac{\|\mathbf{x}_1 - \mathbf{x}_3\| \, \|\mathbf{x}_2 - \mathbf{x}_4\|}{\|\mathbf{x}_1 - \mathbf{x}_4\| \, \|\mathbf{x}_2 - \mathbf{x}_3\|}.$$

The next class is *the group of affine transformations*. Affine transformations preserve every property that is preserved by the projective group and has a number of very useful invariants itself. Most notably parallelism is preserved by the affine group. This corresponds to the fact that the plane at infinity is preserved by affine transformations and if a transformation preserves the plane at infinity then it is an affine transformation. For affine transformations the last row of the transformation matrix is [0 0 0 1]. Hence the matrix has the special form

$$\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$$

and to see that this preserves $\mathbf{\Pi}_\infty$ consider

$$\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ a \end{bmatrix} = \begin{bmatrix} \mathtt{A}\mathbf{X} + a\mathbf{t} \\ a \end{bmatrix}.$$

If $a = 0$ then the point is at infinity as well as the result, if $a \neq 0$ then point is finite and so is the result. Hence this preserves $\mathbf{\Pi}_\infty$ as a set however the points change their places over $\mathbf{\Pi}_\infty$.

A more restrictive class of transformations is *the group of similarity transformations*. A similarity transformation preserves every invariant of the affine group plus angles. This is only possible since a similarity transformation preserves the Ω_∞. In the case of a similarity transformation the transformation matrix is even more constrained. The upper left 3×3 part of the matrix is a scalar times a rotation matrix. A similarity transformation is a combination of rotation, isotropic scaling and translation. The matrix has the form

$$\begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$$

where R is a rotation matrix.

The most constrained class is *the group of Euclidean transformations*. These transformations correspond to a rotation about an arbitrary axis plus translation and are

represented by a matrix of the form

$$\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix}$$

where R is a rotation matrix. Since there is no scaling absolute lengths can be measured after a Euclidean transformation. In the autocalibration process there is no way to recover this level of representation without external information such as a reference length.

See related book on the sucject for a more complete discussion [9, 18].

2.4 Camera Model and Two View Geometry

In this section the camera model used, namely *the pinhole camera model* is described. The detailed derivation of the model is presented using the notation of Hartley and Zisserman [18]. The model presented in this section does not include any distortion model. In Chapter 3, the radial distortion is also added to this model. In the first part of this section, the relation between the 3D points and pixel values of their projections is investigated. Meanwhile a projection matrix composed of internal and external parameters is formed. In the second part, the relation between the 2D image points of two views of the same scene is derived.

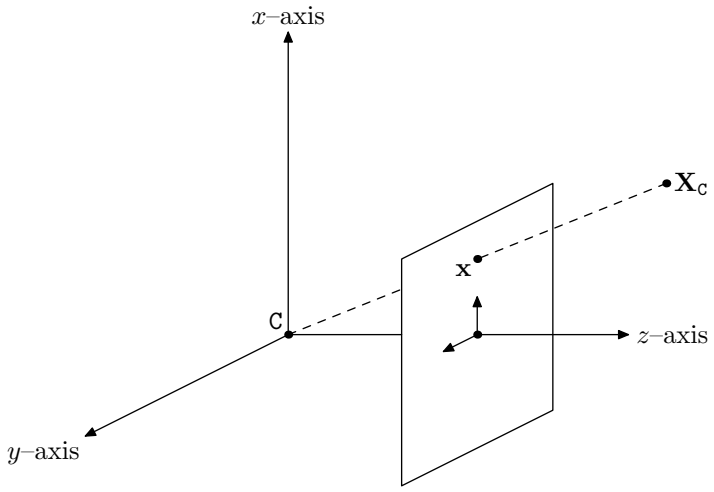

Figure 2.1: Camera coordinate system and projection of a point on the imaging plane

To develop a consistent and clear model of camera geometry, first a number of coordinate systems should be introduced. The first of these coordinate systems is the "Camera Coordinate System". The center of this coordinate system is the camera center. The imaging plane is parallel to the xy plane of this coordinate system, placed at a distance f on the z-axis. f is called *the focal length*. The point at which the z-axis intersects the imaging plane is called *the principal point*. This setup is shown in Figure 2.1. The perspective projection equations are valid in this coordinate frame. Hence if point $\mathbf{X}_\mathtt{C}$ is a point in this coordinate frame with homogeneous coordinates $(X_\mathtt{C}, Y_\mathtt{C}, Z_\mathtt{C}, W_\mathtt{C})$ then its projection can be written as

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_\mathtt{C} \\ Y_\mathtt{C} \\ Z_\mathtt{C} \\ W_\mathtt{C} \end{bmatrix}$$

where the projected point $\mathbf{x}$ is in a 2D homogeneous coordinate system in the imaging plane with the principal point as its origin.

When the image is digitized and represented in a computer system the points are measured in *pixels*. Hence a transformation from the imaging plane coordinates to pixel coordinates is necessary. The resulting projection matrix can be written as

$$\mathtt{P} = \begin{bmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The different focal lengths are due to the fact that pixels can be rectangular. The ratio f_y/f_x is called *the aspect ratio*. In the case of square pixels the aspect ratio is equal to 1. s is called *the skew* and represents a tilted imaging plane or equivalently non-rectangular pixels. Skew is given by the equation $s = \cot(\alpha) f_y$ where α is the skew angle. Today most of the cameras are manufactured precise enough and skew is generally assumed to be 0 and $\alpha = 90°$. u_0 and v_0 represent a shift in the origin of the coordinate system. This is due to the fact that pixel units are measured from the top-left or bottom-left corner of the image. If no information is present then they are taken to be the coordinates of the mid-point of the image.

The matrix $\mathtt{P}$ is called *the camera matrix* [18]. $\mathtt{P}$ is generally decomposed into parts representing the internal parameters such as focal length and representing perspective

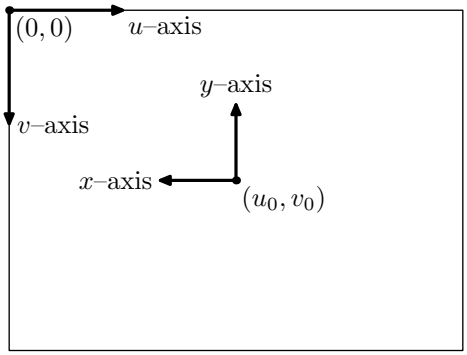

Figure 2.2: Pixel coordinate system

projection. The form of the matrix is

$$\mathtt{P} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathtt{K}\left[\mathtt{I}_{3\times 3} \mid \mathbf{0}\right]$$

where $\mathtt{K}$ is called *the internal calibration of the camera*. The resulting pixel coordinate system is shown in Figure 2.2 where (u, v) are the pixel coordinates. Note that x-axis and u-axis point at opposite directions. This is easily achieved by reversing the sign of f_x. A similar argument is valid for f_y.

Up to now it is assumed that the measurements for 3D points are taken in the coordinate system of the camera. However, in practice the 3D measurements are done in a *world coordinate system*. Hence, the 3D measurements should be transformed from the world coordinates to the camera coordinates. This is easily achieved by a Euclidean transformation of 3D as

$$\mathbf{X}_{\mathtt{C}} = \begin{bmatrix} \mathtt{R}^{\mathsf{T}} & -\mathtt{R}^{\mathsf{T}}\mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \mathbf{X}_{\mathtt{W}}$$

where $\mathtt{R}$ and $\mathbf{t}$ are the rotation and translation of the camera in the world coordinate system. Hence the resulting camera matrix can be written as

$$\mathtt{P} = \mathtt{K}\left[\mathtt{I}_{3\times 3} \mid \mathbf{0}\right] \begin{bmatrix} \mathtt{R}^{\mathsf{T}} & -\mathtt{R}^{\mathsf{T}}\mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} = \mathtt{K}\left[\mathtt{R}^{\mathsf{T}} \mid -\mathtt{R}^{\mathsf{T}}\mathbf{t}\right]. \tag{2.1}$$

To simplify the notation, the following form of the camera matrix will be used:

$$\mathtt{P} = \mathtt{K}\left[\mathtt{R} \mid \mathbf{t}\right] \tag{2.2}$$

where $\mathtt{R}$ and $\mathbf{t}$ represent an inverse motion of the camera in the world coordinate system. Of course the two representations are equivalent. Since $\mathtt{R}$ and $\mathbf{t}$ depend on the choice of the world coordinate system, they are called *the external calibration* of the camera.

The camera model presented above can be used to analyze the projection of 3D points and the resulting image. In the following part of this section the inverse problem will be investigated. The problem is to infer 3D information from several images. It turns out that a single image is insufficient for this task since a point in an image back projects to a line and depth is ambiguous (Although scene constraints can be used to recover depth information, this is not relevant to the study in this thesis. See [5] for more information on single view metrology). Hence more than one image is necessary to recover 3D information. The rest of this section is devoted to the analysis of the case of two images. The derivations will assume the images form a stereo pair obtained from two different cameras however the same equations apply to the case of images taken by a single, moving camera.

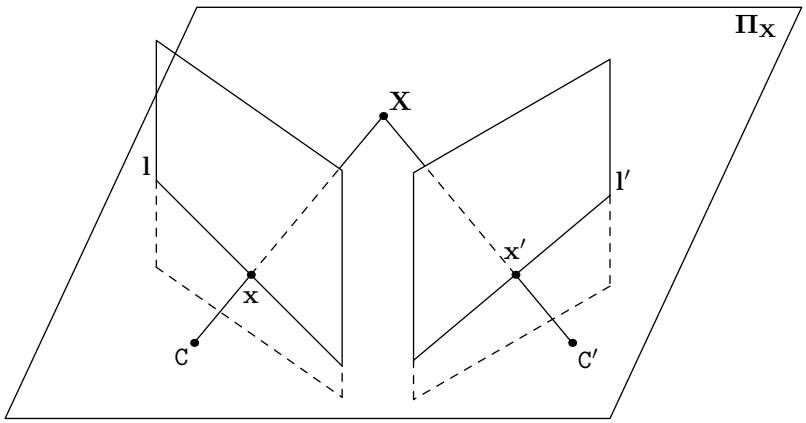

Figure 2.3: The corresponding points $\mathbf{x}$ and $\mathbf{x}'$ lie on the epipolar lines

Consider the case that two images of the same scene are taken by two cameras with camera matrices $\mathtt{P}$ and $\mathtt{P}'$. To recover the depth of a 3D point $\mathbf{X}$, the pixel coordinates of its projections, $\mathbf{x}$ and $\mathbf{x}'$ should be found. If $\mathbf{x}$ and $\mathbf{x}'$ are found then the back projected lines from these points intersect at $\mathbf{X}$. Hence the problem is to find $\mathbf{x}'$ given $\mathbf{x}$ where $\mathbf{x}$ and $\mathbf{x}'$ are called *corresponding points*. If no further constraints are provided, a two dimensional search over the second image is required. However, the geometry of the problem narrows the search to a single dimension. Let the camera

centers be $\mathtt{C}$ and $\mathtt{C}'$. The 3D point $\mathbf{X}$ and the camera centers form a plane $\mathbf{\Pi_X}$ as shown in Figure 2.3. This plane intersects the imaging planes of the two cameras at two lines $\mathbf{l}$ and $\mathbf{l}'$. Note that projections of $\mathbf{X}$ must lie on these lines. Hence if $\mathbf{x}$ is given then the search for $\mathbf{x}'$ is constrained to the line $\mathbf{l}'$.

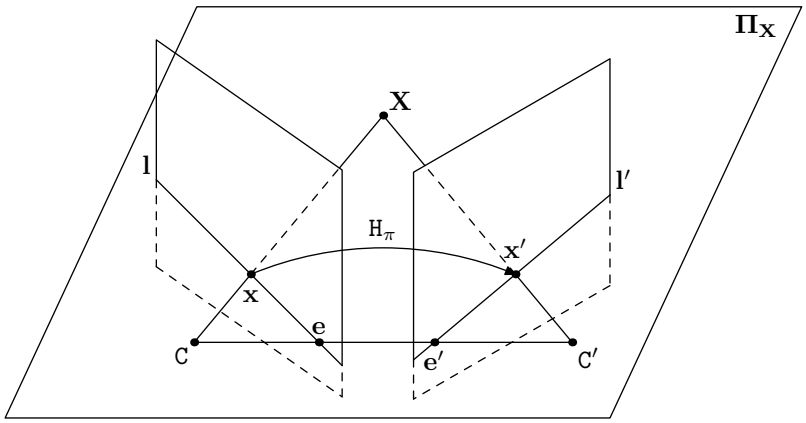

Figure 2.4: The epipolar lines pass through the epipoles $\mathbf{e}$ and $\mathbf{e}'$

What remains is a way to obtain $\mathbf{l}'$ given $\mathbf{x}$. The line joining $\mathtt{C}$ and $\mathtt{C}'$ intersects the imaging planes at two points $\mathbf{e}$ and $\mathbf{e}'$ which are called *the epipoles* as shown in Figure 2.4. Note that the lines $\mathbf{l}$ and $\mathbf{l}'$ must pass through $\mathbf{e}$ and $\mathbf{e}'$, since they are in the imaging planes and $\mathbf{\Pi_X}$ at the same time. Hence $\mathbf{l}$ and $\mathbf{l}'$ are called *the epipolar lines* corresponding to points $\mathbf{x}$ and $\mathbf{x}'$. Since $\mathbf{x}$ and $\mathbf{x}'$ are in the same plane, there is a homogeneous transformation of 2D taking $\mathbf{x}$ to $\mathbf{x}'$ as

$$\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}.$$

The line $\mathbf{l}'$ passes through $\mathbf{x}'$ and $\mathbf{e}'$ hence can be written as (See Section 2.2)

$$\mathbf{l}' = \left[\mathbf{e}'\right]_\times \mathbf{x}' = \left[\mathbf{e}'\right]_\times \mathtt{H}_\pi \, \mathbf{x}$$

where

$$\left[\mathbf{e}'\right]_\times = \begin{bmatrix} 0 & -e'_z & e'_y \\ e'_z & 0 & -e'_x \\ -e'_y & e'_x & 0 \end{bmatrix} \tag{2.3}$$

is the matrix multiplication form of the cross product of two vectors. The matrix $[\mathbf{e}']_\times \mathtt{H}_\pi$ is called *the fundamental matrix* and denoted by $\mathtt{F}$. The fundamental matrix has been introduced to the literature by O. Faugeras and Q.-T. Luong (See [22]).

The equation of the epipolar line can now be written as

$$\mathbf{l}' = \mathtt{F}\mathbf{x}. \tag{2.4}$$

The fact that $\mathbf{x}'$ is on $\mathbf{l}'$ can be written as ${\mathbf{x}'}^{\mathsf{T}}\mathbf{l}' = 0$. Which is equal to

$${\mathbf{x}'}^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0. \tag{2.5}$$

Equation (2.5) is called *the epipolar constraint* and plays an important role in the computation of the fundamental matrix. An important property of the fundamental matrix is that it is of rank 2. The null space is spanned by $\mathbf{e}$. Note that in the case there is no rotation between the cameras, the imaging planes are parallel to each other and the epipoles are at infinity.

2.5 Computation of the Fundamental Matrix

The fundamental matrix encodes the geometry of two views. In this section, methods of computing the fundamental matrix based on known corresponding points is presented. In particular, solutions for the case of seven and the case of eight or more correspondences are shown.

When there are eight or more point correspondences, the fundamental matrix can be found by placing constraints on the solution using equation (2.5). A constraint, linear in the entries of $\mathtt{F}$ is obtained by rewriting the epipolar constraint as

$$\begin{aligned} {\mathbf{x}'}^{\mathsf{T}}\mathtt{F}\mathbf{x} &= \begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= f_1xx' + f_2yx' + f_3x' + f_4xy' + f_5yy' + f_6y' + f_7x + f_8y + f_9 \\ &= \begin{bmatrix} xx' & yx' & x' & xy' & yy' & y' & x & y & 1 \end{bmatrix} \mathbf{f} = 0. \end{aligned} \tag{2.6}$$

If there are eight correspondences then each equation of the form (2.5) can be stacked together to obtain a linear equation of the form

$$\mathtt{A}\mathbf{f} = 0 \qquad \text{subject to} \qquad \|\mathbf{f}\| = 1 \tag{2.7}$$

whose solution is the right null vector of $\mathtt{A}$. If there are more than eight correspondences than the problem is over constrained and a least-squares solution is found by the singular value decomposition (SVD) of $\mathtt{A}$. The SVD of a rectangular matrix is of

the form $\mathtt{A} = \mathtt{U}\mathtt{W}\mathtt{V}^{\mathsf{T}}$ where $\mathtt{U}$ and $\mathtt{V}$ are orthogonal matrices and $\mathtt{W}$ is a diagonal matrix with non-negative entries. The diagonal elements of $\mathtt{W}$ are called the singular values of $\mathtt{A}$. The columns of $\mathtt{U}$ and $\mathtt{V}$ are the left and right singular vectors of $\mathtt{A}$ respectively. In this case the solution is the unit right singular vector corresponding to the smallest singular value of $\mathtt{A}$. The constraint $\|\mathbf{f}\| = 1$ is chosen to ensure a unique solution since $\mathtt{F}$ is a homogeneous matrix whose scale is arbitrary. In any case, the algorithm is called *the eight point algorithm.*

The computed $\mathtt{F}$ matrix satisfies the epipolar constraint, but is not necessarily of rank 2. As a result, the epipolar lines corresponding to this $\mathtt{F}$ matrix do not intersect at a single point $\mathbf{e}$, but are scattered in the vicinity of the epipole [18]. One way to solve this problem is to compute a new matrix $\hat{\mathtt{F}}$ of rank 2 that is closest to the computed $\mathtt{F}$ matrix in the sense that the Frobenius norm $\left\|\mathtt{F} - \hat{\mathtt{F}}\right\|_{\mathbf{F}}$ is minimized. The solution is found by taking the singular value decomposition of $\mathtt{F}$ as

$$\mathtt{F} = \mathtt{U} \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \end{bmatrix} \mathtt{V}^{\mathsf{T}}$$

and then computing the new fundamental matrix $\hat{\mathtt{F}}$ as

$$\hat{\mathtt{F}} = \mathtt{U} \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathtt{V}^{\mathsf{T}}$$

assuming that the singular values are in descending order.

Although the above procedure is mathematically correct, the obtained solution is not robust to noise in data points. Hartley has pointed out that the reason for this is the numerical conditioning of the matrix $\mathtt{A}$ [16]. The entries of the matrix are of different orders where sometimes two pixel coordinates are multiplied and sometimes the entry is simply equal to 1. Hartley also showed that an affine transformation of the coordinates can help to increase the numerical conditioning of $\mathtt{A}$ dramatically. The affine transformation is chosen to center data points around origin and scale the average distance of the points from the origin to $\sqrt{2}$. The computations are done using the normalized data and the resulting $\mathtt{F}$ matrix is converted so that the epipolar constraint is satisfied for the unnormalized data. The procedure is as follows:

1. Compute the mean values of the x and y coordinates of the data points in the

first and second images as

$$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad m_y = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad m_{x'} = \frac{1}{n}\sum_{i=1}^{n} x'_i, \quad m_{y'} = \frac{1}{n}\sum_{i=1}^{n} y'_i$$

2. Compute the average distance of the data points to the origin as

$$\sigma_{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \sqrt{[(x_i - m_x)^2 + (y_i - m_y)^2]}$$
$$\sigma_{\mathbf{x}'} = \frac{1}{n}\sum_{i=1}^{n} \sqrt{\left[(x'_i - m_{x'})^2 + (y'_i - m_{y'})^2\right]}$$

3. Transform points as

$$\hat{x} = \sqrt{2}\cdot\left(\frac{x - m_x}{\sigma_{\mathbf{x}}}\right), \quad \hat{y} = \sqrt{2}\cdot\left(\frac{y - m_y}{\sigma_{\mathbf{x}}}\right)$$
$$\hat{x}' = \sqrt{2}\cdot\left(\frac{x' - m_{x'}}{\sigma_{\mathbf{x}'}}\right), \quad \hat{y}' = \sqrt{2}\cdot\left(\frac{y' - m_{y'}}{\sigma_{\mathbf{x}'}}\right)$$

4. Compute the matrix $\mathtt{F}'$ using $(\hat{x}, \hat{y}, \hat{x}', \hat{y}')$ with the 8-Point algorithm.

5. Compute the matrix $\mathtt{F} = \mathtt{T}'^{\mathsf{T}}\mathtt{F}'\mathtt{T}$ where

$$\mathtt{T} = \begin{bmatrix} \frac{\sqrt{2}}{\sigma_{\mathbf{x}}} & 0 & -\frac{\sqrt{2}m_x}{\sigma_{\mathbf{x}}} \\ 0 & \frac{\sqrt{2}}{\sigma_{\mathbf{x}}} & -\frac{\sqrt{2}m_y}{\sigma_{\mathbf{x}}} \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathtt{T}' = \begin{bmatrix} \frac{\sqrt{2}}{\sigma_{\mathbf{x}'}} & 0 & -\frac{\sqrt{2}m_{x'}}{\sigma_{\mathbf{x}'}} \\ 0 & \frac{\sqrt{2}}{\sigma_{\mathbf{x}'}} & -\frac{\sqrt{2}m_{y'}}{\sigma_{\mathbf{x}'}} \\ 0 & 0 & 1 \end{bmatrix}$$

The above normalization procedure should be used at any point where numerical conditioning may present a problem. In the computation steps generally the normalization is carried out in the beginning, all computations are done on the normalized data and then an inverse transformation is carried out on the computed entities. However it should be noted that at intermediate steps some quantities such as distance measures may need to be computed. The computed values may not be the same for normalized and unnormalized data. Hence if a distance is computed at intermediate steps either the unnormalized entities should be used in the computation, or if possible the computed distance itself should be renormalized.

When only seven corresponding points are known, up to three solutions for the fundamental matrix are possible. Although the solution may not be unique *the seven point* algorithm plays an important role in robust estimation algorithms as will be explained in Chapter 4. In the case of seven points, the matrix $\mathtt{A}$ in equation (2.7) is 9×7 hence is of rank 7. This results in a null space of dimension 2. Let $\mathtt{F}_1$ and

$\mathtt{F}_2$ be the $\mathtt{F}$ matrices corresponding to the unit vectors spanning the null space of $\mathtt{A}$. Then the solution is of the form

$$\mathtt{F} = \alpha\mathtt{F}_1 + (1-\alpha)\mathtt{F}_2. \tag{2.8}$$

Applying the constraint $\det \mathtt{F} = 0$, a third order equation in α is obtained. The real solutions for α is used to compute $\mathtt{F}$, hence there are either one or three solutions. One advantage of the 7-Point algorithm is that the solutions are of rank 2. Hence further steps of ensuring the rank constraint are not necessary.

For the computation of the fundamental matrix, there are some degenerate configurations of 3D points. For example, if all the 3D points used in the $\mathtt{F}$ computation lie on a plane then the corresponding points are related by a homography, hence the camera geometry is not constrained enough to allow a unique solution for the $\mathtt{F}$ matrix. These degenerate cases should be detected and avoided if possible. One way to increase robustness is to choose corresponding points that are scattered evenly across the image which decreases the probability of a degenerate case. This also increases the accuracy of the estimated $\mathtt{F}$ matrix. For a comparison between methods for fundamental matrix computation the reader may refer to [27].

2.6 Triangulation

As stated in Section 2.4 to reconstruct a 3D point, the corresponding points are back projected to lines. Ideally the back projected lines intersect at the 3D point. However in practice due to noise in the detection of corresponding points, the back projected lines do not intersect. Hence, there is a need to select a point in 3D, that is as close as possible to the back projected lines. One solution is to choose the mid-point of the line segment that joins the points on the lines that are closest to each other. The other solution is to use the camera matrices in the reconstruction process. In this thesis the latter method is preferred due to its applicability in projective reconstruction.

Suppose that the camera matrices for the images are $\mathtt{P}$ and $\mathtt{P}'$. Then a projected point can be written as $\mathbf{x} = \mathtt{P}\mathbf{X}$. Hence the cross product $\mathbf{x} \times \mathtt{P}\mathbf{X}$ must be zero. This provides two linear constraints on the entries of $\mathbf{X}$. To completely determine $\mathbf{X}$, two other constraints come from the equation $\mathbf{x}' \times \mathtt{P}'\mathbf{X} = 0$. To obtain a unique solution, the homogeneous scale of $\mathbf{X}$ should be eliminated hence the constraint $\|\mathbf{X}\| = 1$ is

added to the problem. Assume that the structure for the camera matrices is

$$\mathtt{P} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}, \quad \mathtt{P}' = \begin{bmatrix} \mathbf{r}'_1 \\ \mathbf{r}'_2 \\ \mathbf{r}'_3 \end{bmatrix}$$

where $\mathbf{r}_i$ and $\mathbf{r}'_i$ are row vectors of size 1×4 corresponding to the i^{th} rows of $\mathtt{P}$ and $\mathtt{P}'$ respectively. Now the form of the equations is

$$\mathtt{A}\mathbf{X} = \begin{bmatrix} x\mathbf{r}_3 - \mathbf{r}_1 \\ y\mathbf{r}_3 - \mathbf{r}_2 \\ x'\mathbf{r}'_3 - \mathbf{r}'_1 \\ y'\mathbf{r}'_3 - \mathbf{r}'_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{0} \qquad \text{subject to } \|\mathbf{X}\| = 1. \tag{2.9}$$

The solution is the unit eigenvector of $\mathtt{A}$ corresponding to the smallest singular value. The solution is a 4-vector and can represent points at infinity. This is particularly useful when computing a projective reconstruction of the scene, since in this case some parts of the scene may be reconstructed at infinity. In the case of projective reconstruction there are other issues that should be considered and an optimal method is presented by Hartley and Sturm [17]. The main difficulty is that since Euclidean information is not available, terms like middle-point do not have any meaning and a method invariant to changes in the projective frame is needed. For other triangulation methods, along with a comparison between them, the reader may refer to [17].

2.7 Homography Relations

When some special conditions are satisfied, points in two different views are related to each other by a transformation of 2D, called a homography. A homography is represented by a 3×3 homogeneous matrix, $\mathtt{H}$. The homography relation is given as

$$\mathbf{x}' = \mathtt{H}\mathbf{x} \tag{2.10}$$

where $\mathbf{x}$ is a point in the first view and $\mathbf{x}'$ is the corresponding point in the second view. In effect, $\mathtt{H}$ can be used to transfer points in the first view to corresponding points in the second view.

When multiple images of a planar scene is acquired, the obtained images are related by a homography. To see that this is possible let the world coordinate system z-axis be orthogonal to the plane and let the origin of the world coordinate system be on

the plane. Hence, for all points on the plane z coordinate is equal to zero. Now consider a point of this plane $\mathbf{X} = [X, Y, 0, 1]^\mathsf{T}$. Assuming the camera matrix is $\mathtt{P}$, the projected point can be written as

$$\mathbf{x} = \mathtt{P}\mathbf{X} = \mathtt{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathtt{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{2.11}$$

Writing $\mathtt{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] = \mathtt{H}_1$ and $\hat{\mathbf{X}} = [X, Y, 1]^\mathsf{T}$ the equation becomes

$$\mathbf{x} = \mathtt{H}_1\hat{\mathbf{X}}. \tag{2.12}$$

Following a similar argument, it can be shown that for another view, the same point $\mathbf{X}$ projects onto

$$\mathbf{x}' = \mathtt{H}_2\hat{\mathbf{X}}. \tag{2.13}$$

Writing $\hat{\mathbf{X}} = {\mathtt{H}_1}^{-1}\mathbf{x}$,

$$\mathbf{x}' = \mathtt{H}_2{\mathtt{H}_1}^{-1}\mathbf{x} = \mathtt{H}\mathbf{x} \tag{2.14}$$

is obtained. Note that the computed homography is independent of the choice of the world coordinate system.

When the camera motion has only a rotational component, the acquired images are also related by a homography. To see this, assume that the first image is located at the origin and hence has the projection matrix

$$\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}].$$

Let the camera matrix for the second view be

$$\mathtt{P}' = \mathtt{K}'[\mathtt{R}' \mid \mathbf{0}]$$

where $\mathtt{R}'$ is the rotation between the first and the second views. Hence a world point $\mathbf{X} = [X\ Y\ Z\ 1]^\mathsf{T}$ is projected as

$$\mathbf{x} = \mathtt{K}\tilde{\mathbf{X}}, \quad \mathbf{x}' = \mathtt{K}'\mathtt{R}'\tilde{\mathbf{X}}$$

where $\tilde{\mathbf{X}} = [X\ Y\ Z]^\mathsf{T}$. Writing $\tilde{\mathbf{X}} = \mathtt{K}^{-1}\mathbf{x}$,

$$\mathbf{x}' = \mathtt{K}'\mathtt{R}'\mathtt{K}^{-1}\mathbf{x}$$

is obtained. The matrix $\mathtt{H} = \mathtt{K}'\mathtt{R}'\mathtt{K}^{-1}$ is the homography between the first and the second views.

The Direct Linear Transformation (DLT) algorithm is used to compute a homography between two images. The relation given as Equation (2.10) can be rewritten as

$$\mathbf{x}' \times \mathtt{H}\mathbf{x} = 0.$$

Hence, the following linear system of equations is obtained

$$\mathtt{A}\mathbf{h} = \begin{bmatrix} \mathbf{0}^\mathsf{T} & -\mathbf{x}^\mathsf{T} & y'\mathbf{x}^\mathsf{T} \\ \mathbf{x}^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x'\mathbf{x}^\mathsf{T} \\ -y'\mathbf{x}^\mathsf{T} & x'\mathbf{x}^\mathsf{T} & \mathbf{0}^\mathsf{T} \end{bmatrix} \mathbf{h} = \mathbf{0}. \tag{2.15}$$

where $\mathbf{x}' = [x'\ y'\ 1]^\mathsf{T}$ and $\mathbf{h}$ is the vector containing the elements of $\mathtt{H}$ in row major order [18]. Note that only two of the rows of $\mathtt{A}$ are linearly independent. If four corresponding points are given, this system can be solved for a unit vector $\mathbf{h}$ and the homography can be obtained. The vector is chosen to be a unit vector since $\mathtt{H}$ is of arbitrary scale and a unique solution should be guaranteed. If more than four correspondence is available, then the least squares solution is obtained by SVD as the unit right singular vector of $\mathtt{A}$ corresponding to the smallest singular value.

Of course the correspondence data should be normalized as detailed in Section 2.5. Then the computed homography $\mathtt{H}'$ is normalized as

$$\mathtt{H} = \mathtt{T}'^{-1}\mathtt{H}'\mathtt{T}.$$

For a detailed discussion of DLT see [18].

2.8 Summary

In this chapter background information on projective geometry, camera model, two-view relations as well as some algorithms to compute the fundamental matrix and 3D points based on corresponding points are presented.

It is shown that the use of projective geometry both simplifies the notation and provides invaluable insight into the problem structure. The equations such as (2.2) can not be written in the convenient matrix form without the homogeneous notation or the presented projective concepts. The projective entities and transformations introduced in this chapter are used extensively in the following chapters.

In sections 2.4 and 2.5 the fundamental matrix along with methods to compute it from point correspondences is described. The Fundamental matrix is an essential

tool in the computation of the multiple view geometry. The point correspondences depend on `F`, which in turn depends on the point correspondences. This coupling itself presents a difficulty in the solution of structure and geometry computation. However the combination of robust algorithms and powerful optimization methods such as bundle adjustment, uses this coupling to solve both problems simultaneously.

The triangulation method of Section 2.6 will be used in Chapter 3 to provide measurements to evaluate the performance of the calibration procedure.

The homography relations are used in Chapters 3 and 5 in the calibration processes. Chapter 3 uses the homography relation for the case of a planar scene and Chapter 5 uses a set of homographies to relate images taken by a rotating camera.

Chapter 3

Manual Calibration

3.1 Introduction

In this chapter the implementation of the manual calibration system based on the method presented in [33] is explained. The method has been chosen for its flexibility in the construction of the calibration pattern and ease of implementation. In general manual calibration methods use a calibration object with precisely known 3D structure to obtain camera calibration. To achieve this the corresponding imaged points of the calibration object should be detected with high precision in the images. Then the relations between imaged points and 3D points are used to obtain parameters of the camera model.

Section 3.2 describes the camera setup and the calibration pattern used. The image sequences used in the calibration process are also explained. Section 3.3 will show how the imaged points are detected and the computation of a planar homography between the imaged points and the 3D coordinates of the calibration pattern. In Section 3.4 the constraints on the internal calibration are obtained using the computed homography and then an initial solution is found using a linear method. Section 3.5 shows how the distortion effects are added to the camera model developed in Section 2.4 and the initial solution found in Section 3.4 is used to compute a better calibration using a non-linear algorithm including the distortion model. Section 3.6 shows a way to obtain external calibration of the stereo camera system. Finally in Section 3.7 the obtained calibration results for the stereo system is presented and the stereo system is used to obtain 3D information from the acquired images.

3.2 Camera Setup

The camera system used in this thesis is a stereo pair of cameras attached to a fixed platform approximately in a parallel position. The cameras have zoom capability but during the manual calibration procedure this capability should not be used. The cameras have PAL output connectors which are connected to the Matrox Meteor frame grabber card located on a PC Desktop System. The frames from the camera system are captured and stored on the computer system using the Matrox MIL-*Light* software. Table 3.1 shows the hardware and software used in the testing of the algorithms in this thesis.

Table 3.1: Hardware and software setup used in the testing steps

System	**Properties**
PC Desktop System	Intel Pentium 4 2.4GHz 512 MB RAM Windows XP Proffessional
Frame Grabber Card	Matrox Meteor II/Standard
Analog Camera	Sony FCB-IX47AP
IDE	Microsoft Visual C++ 2003
Software Libraries	Intel OpenCV Library beta3.1
	Matrox MIL-*Light* 7.5

The calibration object used in the computation of internal and external calibration is a planar checkerboard pattern. The checkerboard pattern is obtained by a laser quality printer on A4 sized paper. The use of the checkerboard pattern allows sub-pixel corner detection of checkerboard corners hence increasing accuracy. The calibration pattern is shown in Figure 3.1. Each square box over the calibration pattern has dimensions 30mm $\times$ 30mm. Only the inner corners are used in the calibration process giving $8 \times 6 = 48$ feature points per image of the calibration pattern.

The internal calibrations for the cameras are found individually. Then internal calibration together with the stereo images are used to obtain the external parameters. An alternative way is to obtain both internal and external parameters at the same time. However this requires more sophisticated parametrization schemes in the non-

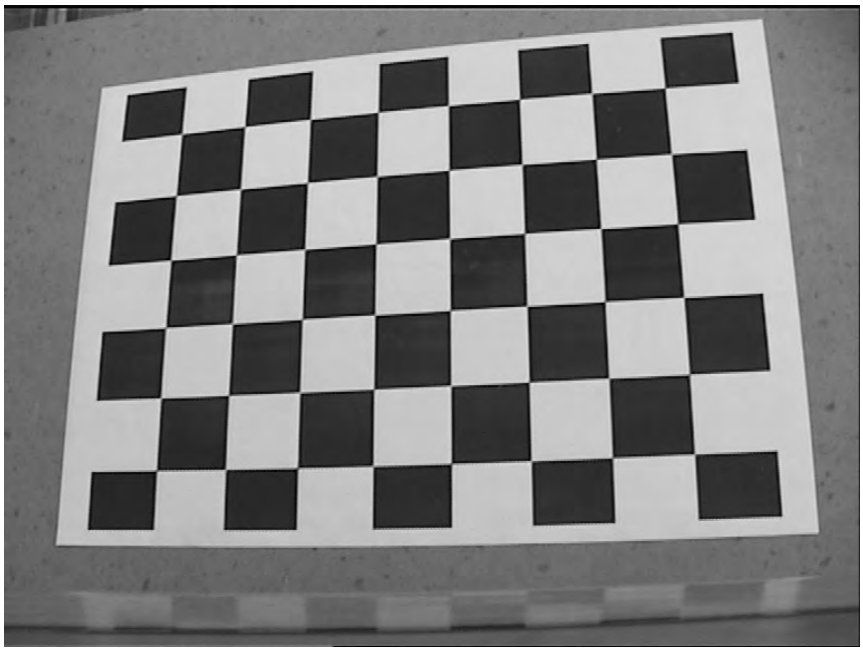

Figure 3.1: An image of the calibration pattern used in the calibration process

linear algorithms hence it is not preferred in this study.

The implementation is tested on several image sequences. To be able to refer to these sequences, their labels together with information on the image contents is presented in Table 3.2. Image sequences *Leftseq1* and *Leftseq2* correspond to the same camera settings however the digitizer is used to sample an image of smaller size. A similar argument applies to *Rightseq1* and *Rightseq2*. Overall the scale column shows the scaling of the image in both directions. The scaling is done by using the related frame grabber options. The thumbnail views of the sequences are given as Appendix D.

3.3 Homography Computation

The first step in the calibration process is to obtain a 3×3 matrix representing a homography between the imaged points and the 3D points of the calibration pattern. To make this possible the world coordinate system is chosen so that its x and y axes correspond to the x and y axes of the pixel coordinate system with the top left corner point as the origin [33]. Hence a corner point which is third from the left, fourth

Table 3.2: Image sequences used in the manual calibration process

Image Sequence	**Resolution**	**# of Images**	**Scale**	**Properties**
Leftseq1	768×576	12	1	Wide-Angle
Rightseq1	768×576	12	1	Wide-Angle
Leftseq2	384×288	10	0.5	Wide-Angle
Rightseq2	384×288	10	0.5	Wide-Angle
Leftseq3	384×288	9	0.5	Telephoto
Rightseq3	384×288	9	0.5	Telephoto
Leftseq4	384×288	10	0.5	Telephoto
Rightseq4	384×288	10	0.5	Telephoto

from the top has 3D homogeneous coordinates $[2 \times 30, 3 \times 30, 0, 1]^{\mathsf{T}} = [60, 90, 0, 1]^{\mathsf{T}}$ in the world coordinate frame where all units are in mm. In Section 2.7 it is shown that the relation between the point on the plane and the imaged point is given by

$$\mathbf{x} = \mathtt{H}\hat{\mathbf{X}} \tag{3.1}$$

where $\hat{\mathbf{X}} = [X, Y, 1]^{\mathsf{T}}$.

Hence a homography is obtained between image features and 3D points. The aim is to compute this homography which will be used in the next section to constrain the internal parameters. In order to do this first checkerboard corners are detected to sub-pixel resolution. The detected corners are ordered interactively in the left to right, top to bottom order. This way correspondences between image pixels and 3D points are obtained.

An initial homography is calculated using the normalized DLT method described in Section 2.7. This initial homography is then refined using Levenberg-Marquardt minimization algorithm. The cost function is the sum of the squared distances between the imaged pixels and the transferred 3D points using `H`. In practice this non-linear minimization step is found to provide improvement only in isolated cases. In most of the cases the linear estimation works just as fine. The comparison is based on *the residual pixel error* which in this case is defined to be

$$\epsilon_{\text{res}} = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \mathtt{H}\hat{\mathbf{X}}_i\right\|^2} \tag{3.2}$$

where $\mathbf{x}_i$ and $\hat{\mathbf{X}}_i$ is the i^{th} corresponding pair of points. See [18] for the definition of residual error in different cases.

3.4 Internal Calibration

The calculated homographies are used to place constraints on the internal parameters. To obtain these constraints consider the equation

$$\mathtt{H} = \lambda \mathtt{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$$

where λ indicates the unknown homogeneous scaling factor of the estimated $\mathtt{H}$. Using orthonormality of $\mathbf{r}_1$ and $\mathbf{r}_2$ the following constraints are derived

$$\begin{aligned} \mathbf{r}_1^\mathsf{T}\mathbf{r}_2 &= \left(\mathtt{K}^{-1}\mathbf{h}_1\right)^\mathsf{T} \mathtt{K}^{-1}\mathbf{h}_2 \\ &= \mathbf{h}_1^\mathsf{T}\mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}\mathbf{h}_2 = 0 \end{aligned} \tag{3.3}$$

$$\begin{aligned} \mathbf{r}_1^\mathsf{T}\mathbf{r}_1 &= \mathbf{r}_2^\mathsf{T}\mathbf{r}_2 \\ \mathbf{h}_1^\mathsf{T}\mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}\mathbf{h}_1 &= \mathbf{h}_2^\mathsf{T}\mathtt{K}^{-\mathsf{T}}\mathtt{K}^{-1}\mathbf{h}_2. \end{aligned} \tag{3.4}$$

These constraints are then used to solve for the internal parameters of the camera (See [33] for details). At least three views are required if no information is available and two views are sufficient if skew is assumed to be zero. The solution obtained from this linear algorithm is then used to calculate the external parameters for each view.

After the linear step we have a complete projection model for each view. Hence we can project the 3D points onto the imaging planes and the resulting projections should match with the detected corner points. In practice this is not satisfied. Hence a non-linear minimization technique is used to minimize the sum of squared distances of the projected points and the detected corner points. This requires a parametrization of the internal and external parameters. The parametrization of $\mathtt{K}$ and $\mathbf{t}$ is easy and is equal to the unknown entries. The parametrization of $\mathtt{R}$ is done using the Rodrigues formula as described in Appendix A [33].

In contrast to the homography computation the non-linear minimization step for the internal calibration is an essential step. The residual error is significantly decreased during the minimization and all parameters of the calibration is optimized.

3.5 Radial Distortion

Up to now it is assumed that the camera model conforms to the pinhole camera model. However this is not true for all imaging conditions. In particular for shorter focal lengths (in the wide-angle settings) the image is subject to a non-linear distortion. Since the distortion amount is related to the distance from the principal point the distortion is called *the radial distortion*. In this section radial distortion model is shown and a way to undistort points when the distortion parameters are known is described.

Radial distortion is applied to points after they are projected onto the imaging plane but before changing to the pixel coordinates. Since it is a non-linear effect we can not use the matrix notation. Let the notation $\tilde{\mathbf{x}} = \mathsf{RadialDistort}(\mathbf{x})$ denote that the projected point $\mathbf{x}$ is distorted radially to get point $\tilde{\mathbf{x}}$. The complete camera model is then

$$\begin{aligned} \mathbf{x} &= [\mathtt{R} \mid \mathbf{t}]\,\mathbf{X} \\ \tilde{\mathbf{x}} &= \mathsf{RadialDistort}(\mathbf{x}) \\ \hat{\mathbf{x}} &= \mathtt{K}\tilde{\mathbf{x}}. \end{aligned} \tag{3.5}$$

The radial distortion function applies only to the radial component and is related to the even powers of the radial component. Hence if a point with polar coordinates (r, θ) is distorted, the resulting point has the same angle θ but the radius is distorted as

$$\tilde{r} = r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right) \tag{3.6}$$

where κ_1 and κ_2 are radial distortion coefficients [33]. In this model two coefficients are used. Higher order models are possible but their effects tend to diminish very rapidly.

In the calibration process the effect of the radial distortion is added after an initial solution is found for the internal and external calibration. As indicated in Section 3.4 the non-linear minimization step minimizes the distance between the detected corners and the projected points. Now the projection is done using Equation (3.5). The initial values for κ_1 and κ_2 are taken as zero.

Although calculating the distorted points is easy, the inverse transformation does not have a closed form expression. Hence an iterative approach is necessary. We have simply used a gradient descent approach to accomplish the undistortion. In the

undistortion problem a distorted point with polar coordinates $(\tilde{r}, \theta)$ and the distortion coefficients κ_1 and κ_2 are given. The aim is to find the point (r, θ) that minimizes

$$\varphi = \left[\tilde{r} - r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)\right]^2. \tag{3.7}$$

The algorithm used in the computations is given in Appendix B.

3.6 External Calibration

In the camera setup used the left camera coordinate system is chosen to be the world coordinate system. The external calibration of the stereo camera system is then defined to be the rotation and translation of the right camera with respect to the left camera. The already computed information can be used to calculate these two parameters if the computations are based on stereo images of the calibration pattern. Assume that the rotation and translation from the world coordinates to the left and right imaging planes are $(\mathtt{R}_0, \mathbf{t}_0)$ and $(\mathtt{R}_1, \mathbf{t}_1)$. Then the transformation from the left camera to the imaging plane is the inverse transformation $\left(\mathtt{R}_0^\mathsf{T}, -\mathtt{R}_0^\mathsf{T}\mathbf{t}_0\right)$. The transformation from the left camera to the right camera can then be computed as

$$\begin{bmatrix} \mathtt{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_0^\mathsf{T} & -\mathtt{R}_0^\mathsf{T}\mathbf{t}_0 \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R}_1\mathtt{R}_0^\mathsf{T} & -\mathtt{R}_1\mathtt{R}_0^\mathsf{T}\mathbf{t}_0 + \mathbf{t}_1 \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{3.8}$$

where the external calibration is $\left(\mathtt{R} = \mathtt{R}_1\mathtt{R}_0^\mathsf{T}, \mathbf{t} = -\mathtt{R}_1\mathtt{R}_0^\mathsf{T}\mathbf{t}_0 + \mathbf{t}_1\right)$.

In earlier discussions the world coordinates system is taken to coincide with the planar surface of the calibration pattern. Hence the orientation of each camera with respect to this common coordinate system has already been computed. Then for each pair of images we can compute the external calibration of the camera system. However since previous steps used a single camera system, the constancy of the external calibration of the stereo camera system is not enforced during the computation. This results in multiple solutions for the external calibration.

One way to solve this problem is to compute the residual error for each case and to choose the one corresponding to the smallest residual error. Another way is to combine the solutions possibly by taking a weighted average using residual error information. However these methods are suboptimal. An optimal way can be found by enforcing the constancy of the external calibration in the non-linear minimization by a suitable parametrization. In the next section the computation of the external calibration is based on choosing the calibration with the minimum residual error.

The results of this approach is shown to perform well enough for the task of robot navigation.

3.7 Results and Conclusion

In this section the results of the implemented manual calibration algorithm will be presented. The algorithm has been tested on the image sequences listed in Table 3.2.

Table 3.3: Internal calibration results

Sequence	$\mathbf{f_x}$	$\mathbf{f_x/f_y}$	**skew**	α	$\mathbf{u_0}$	$\mathbf{v_0}$	κ_1	κ_2
Leftseq1	909.003	0.9987	-0.078	90.000	378.34	310.51	-0.279	0.367
Rightseq1	904.629	0.9989	0.051	90.000	365.39	312.15	-0.267	0.280
Leftseq2	454.189	0.9981	0.015	90.000	181.60	151.73	-0.290	0.362
Rightseq2	452.501	0.9986	0.174	90.000	180.56	147.70	-0.267	0.336
Leftseq3	1522.48	0.9995	0.076	90.000	186.89	147.73	0.088	-10.8
Rightseq3	1631.99	0.9983	1.074	89.999	181.17	149.03	0.192	-13.3
Leftseq4	2033.44	0.9979	1.656	89.999	156.18	146.85	-0.216	-0.536
Rightseq4	2272.06	0.9957	6.449	89.997	166.74	129.98	-0.460	132.9

The calculated internal calibration at the end of the non-linear minimization step is listed in Table 3.3. The aspect ratio is very close to 1 indicating square pixels as expected. The skew value are found to be very close to zero. Since *Leftseq1* and *Rightseq1* are obtained with the same camera as *Leftseq2* and *Rightseq2* respectively, the calibration should be at least close in both cases. Since the images are scaled by 0.5 in the case of *Leftseq2* and *Rightseq2* the focal lengths and principal point coordinates are nearly half the values obtained for *Leftseq1* and *Rightseq1*. Especially the focal lengths have the same ratio as required. This points to the fact that a consistent set of results has been obtained using different image sets. The last two rows shows the results for the telephoto setting. The focal length is much larger as expected. For the sequence *Rightseq4* the value for κ_2 is found to be 132.938. This is the result of the fact that the focal length is very large and the radial components become very small. Since κ_2 value is the multiplier for the fourth power its effect is very small. Hence the non-linear step without bounds on this value swept away from

the actual value. The value of the κ_2 will be taken as zero for sequences *Leftseq4* and *Rightseq4* in the further experiments.

Table 3.4: Residual pixel error for all data points in the sequence

Sequence	$\mathbf{e}_{\text{res}}$ (Linear)	$\mathbf{e}_{\text{res}}$ (Non-Linear)
Leftseq1	130.326	0.127
Rightseq1	19.469	0.201
Leftseq2	7.461	0.060
Rightseq2	42.753	0.057
Leftseq3	8.133	0.058
Rightseq3	33.59	0.052
Leftseq4	8.133	0.058
Rightseq4	8.898	0.062

One way to test the model is to project the 3D points onto the images using Equation (3.5) and to compare these to the detected corners. The quantitative way to do this to measure the residual pixel error computed with Equation (3.2). Table 3.4 show the residual pixel error computed considering all points on all images of the sequence used in the computations. The error value after the linear estimation and after the non-linear minimization step is shown. The linear step performs very poorly due to the fact that the radial distortion coefficients are taken as zero and other parameters are calculated using radially distorted points. As expected the values after non-linear minimization are very close to zero and the projected points differ by a fraction of a pixel.

To get a visual feedback the projected points are drawn on the images. Figure 3.2 shows an image from the sequence *Rightseq2* with projected 3D points marked as gray diamonds. The projected points are very close to the detected corners. To show the effect of the radial distortion the same points are projected without the distortion model. The resulting image is shown in Figure 3.3. As expected the points away from the center are subject to radial distortion and the effect can not be neglected.

The radial distortion coefficients can be used to undistort the image. The image can be sampled to obtain an undistorted version. Figure 3.4 shows a distorted image. The curvature of the edges of the table and the paper shows the effect of the distortion.

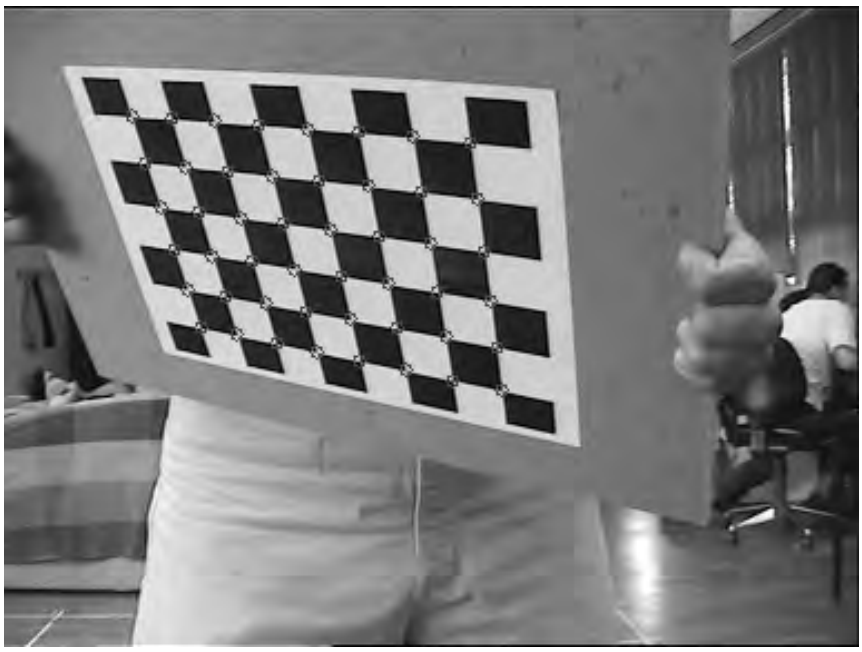

Figure 3.2: The projected 3D points are very close to the corner points

Figure 3.5 shows the undistorted image.

Although the above results show consistency they are not enough to completely evaluate the calibration results. The obtained calibration should be tested with real 3D data. For this purpose a test pattern is prepared and attached to a planar surface. The pattern consists of two line segments each of length 15cm and at an angle of 37°to each other. Stereo images of the test pattern are captured for *Leftseq2*, *Rightseq2*, *Leftseq3*, *Rightseq3* and *Leftseq4*, *Rightseq4* sequences. The corners are detected to sub-pixel resolution and the end points of the line segments are picked by the user. Using the obtained calibration information from these sequences the end points of the line segments are reconstructed in 3D and the reconstruction is used to compute the line segment lengths and to measure the angle between them. Figure 3.6 shows the detected corners and selected end points on the test pattern. The test pattern is shown as Figure 3.7 with the measured lengths and the angle shown as overlaid text. The MetaPost code to draw the test pattern is given in Appendix C.

Tables 3.5, 3.6 and 3.7 shows the obtained results for the test pattern. Lengths are in millimeters. As can be seen from the tables the length estimations and the angle

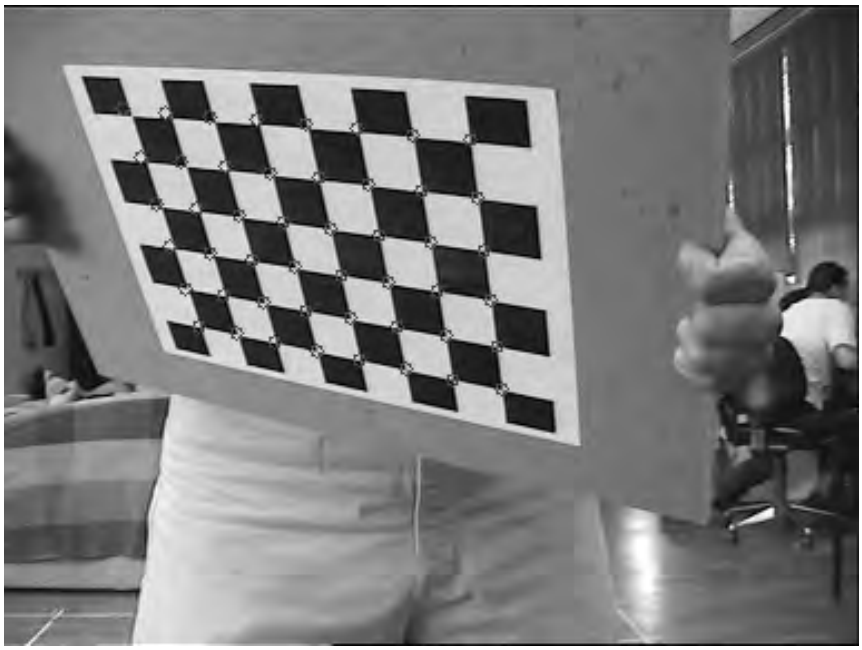

Figure 3.3: The points close to the boundary are distorted significantly when radial distortion is neglected

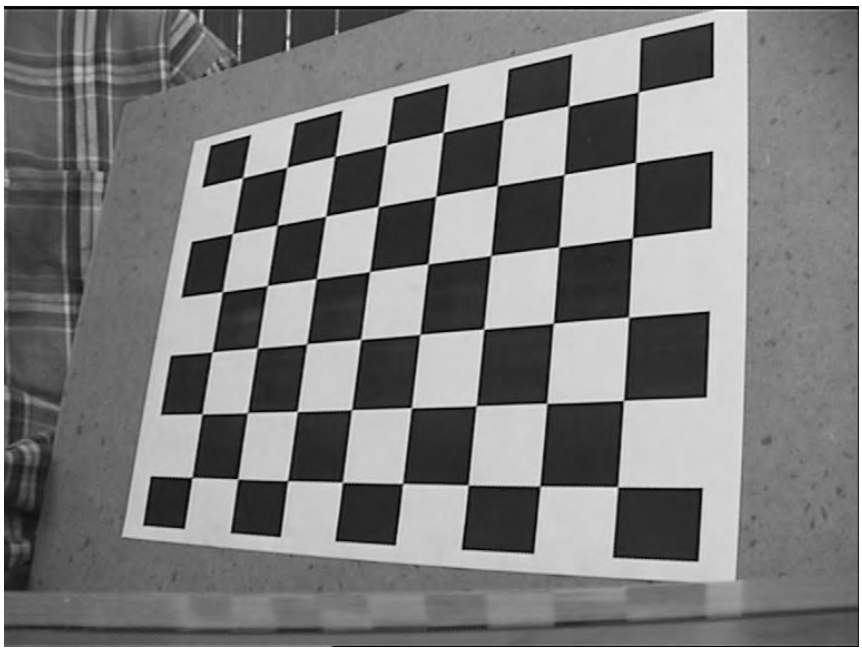

Figure 3.4: The distorted image belonging to the sequence *LeftSeq1*

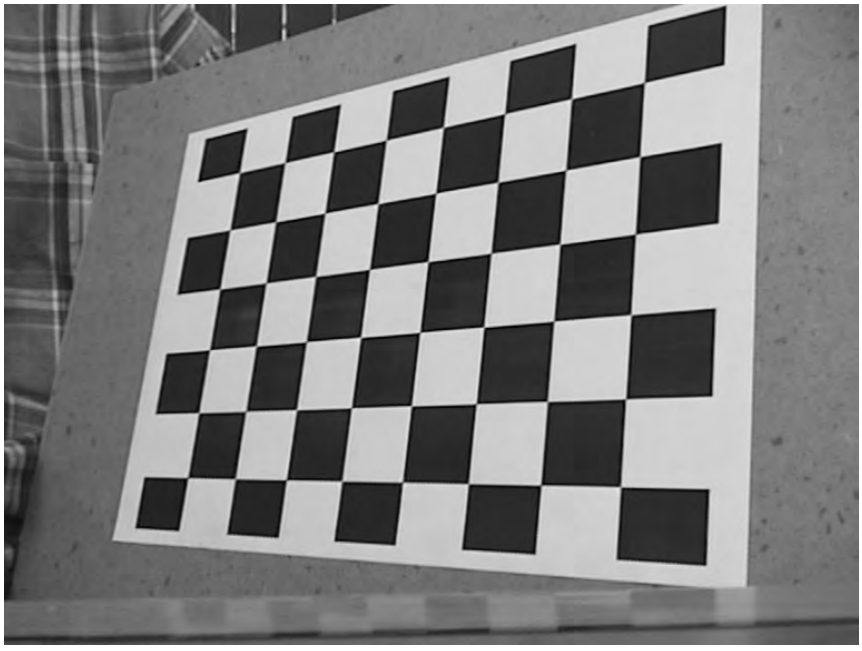

Figure 3.5: The image is undistorted using the calculated radial distortion coefficients

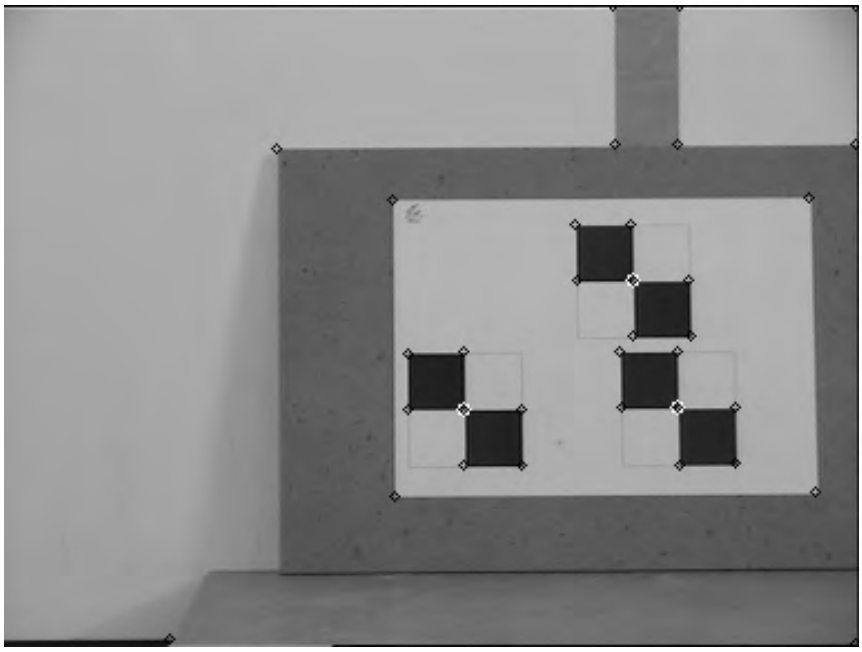

Figure 3.6: The end points of the line segments are selected by the user from the detected corners.

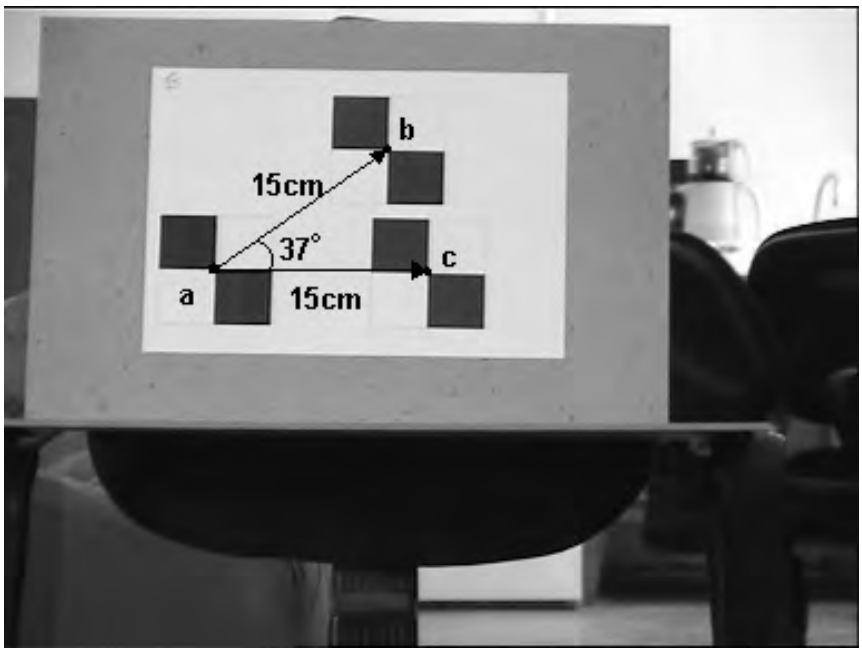

Figure 3.7: An image of the test pattern with measured lengths and the angle indicated

measurements shows that the calibration process is successful. Especially the results for the telephoto sequences *Leftseq3*, *Rightseq3* and *Leftseq4*, *Rightseq4* show that the camera system is calibrated to good precision.

Table 3.5: Results of the measurements on the test pattern using calibration from *Leftseq2* and *Rightseq2*

Sequence	**Image Number**	‖ab‖	‖ac‖	∠(bac)
TestSeq1	1	138.646	155.909	41.4
	2	143.966	163.909	42.1
	3	146.470	152.901	43.4
	4	146.049	155.647	40.0
	5	149.519	158.492	42.9
	6	151.298	160.613	40.2
	7	152.992	157.868	38.7
	8	152.640	160.784	38.2
	9	153.667	154.206	39.1
	10	153.682	157.712	38.6
	11	154.554	155.466	36.6
	12	154.566	154.735	36.3
	Average	149.838	157.354	39.8

Table 3.6: Results of the measurements on the test pattern using calibration from *Leftseq3* and *Rightseq3*

Sequence	**Image Number**	‖ab‖	‖ac‖	∠(bac)
TestSeq2	1	149.976	150.104	37.0
	2	150.124	150.433	37.1
	3	150.256	149.872	37.3
	4	150.032	149.935	36.9
	5	150.016	150.105	36.8
	6	150.028	151.266	37.6
	Average	150.072	150.286	37.1

Table 3.7: Results of the measurements on the test pattern using calibration from *Leftseq4* and *Rightseq4*

Sequence	**Image Number**	‖ab‖	‖ac‖	∠(bac)
TestSeq3	1	150.329	151.278	37.4
	2	151.457	150.239	35.5
	3	151.468	150.927	36.6
	4	152.508	153.132	36.6
	5	149.721	151.343	36.4
	6	151.179	151.336	35.8
	Average	151.11	151.376	36.4

Chapter 4

Autocalibration by Absolute Dual Quadric

4.1 Introduction

In this chapter calibration without using 3D information of the scene points is considered. Although manual calibration using a calibration pattern provides good results it has several disadvantages. Recent developments in the area of uncalibrated vision allowed the development of several algorithms to calibrate the cameras on-line, just using rigidity assumptions on scene points. This is only possible when there is no motion in the scene points.

Autocalibration methods are based on multiple view relations. In this thesis only two view relations are considered. As noted in Section 2.4 two view relations are developed using epipolar geometry and fundamental matrix can be used to pack information on the relation of two views. Therefore robust estimation of fundamental matrix from scene point correspondences is a must for reliable autocalibration. Section 4.2 gives details on robust estimation of scene points. Two methods for the computation of the fundamental matrix from point correspondences were given in Section 2.5. Section 4.3 gives details on another algorithm based on the minimization of geometric error. The results of this algorithm is compared to that of the normalized 8-Point algorithm.

4.2 Robust Correspondence Detection

The estimation of the fundamental matrix requires the computation of the feature matches between pairs of images. To reliably match features across images some properties of the feature points should be conserved between images. Although it is possible to use edge segments as features better results are obtained by matching corner points across images. Hence corner correspondences are used in the estimation of the fundamental matrix. The detection of corner points is relatively easy. However only corners which are invariant to affine changes can be reliably detected across multiple images especially if there is significant scaling between images. Hence the corners are detected to sub-pixel resolution using the implementation present in the Intel Open Computer Vision library. The implementation uses the Harris corner detector [13]. Figure 4.1 and Figure 4.2 shows the detected corner points over a pair of images belonging to a sequence used in the autocalibration process. It can be observed from the figures that many corner points are detected reliably in both of the images.

After corner points are detected the correspondence between them should be found. This requires a measure of similarity between corner points. One such similarity measure is the normalized cross-correlation (NCC) between blocks around the detected corners. Although there are other measures such as squared sum of differences the NCC measure is found to be more reliable. Hence a first matching is done using this measure. NCC is defined to be

$$\mathrm{NCC} = \frac{\sum_R (s_1 - m_{s_1})(s_2 - m_{s_2})}{\sqrt{\sum_R (s_1 - m_{s_1})^2 \sum_R (s_2 - m_{s_2})^2}} \tag{4.1}$$

where the summations are over all pixels in the image blocks R centered at the tested corners, s_1 and s_2 are the image intensity values at the first and second images, and m_{s_i} is the mean intensity value over R [12]. For each corner point at the first image the NCC value for each corner in the second image that is within a search region is calculated. The block size depends on the image size but generally 7×7 blocks are used. The search region size depends on the motion of the sequence but to be on the safe side generally large search regions are used. Typical size of the search region is 2/3 of the image size. When the correspondence between a stereo pair of images is to be found, the search region is a narrow band around the y coordinate of the corner in the left image. Although the NCC measure is quite robust, the correspondence found in a large region contains many mismatches. In the next step these correspondence will be refined using robust measures. Hence the

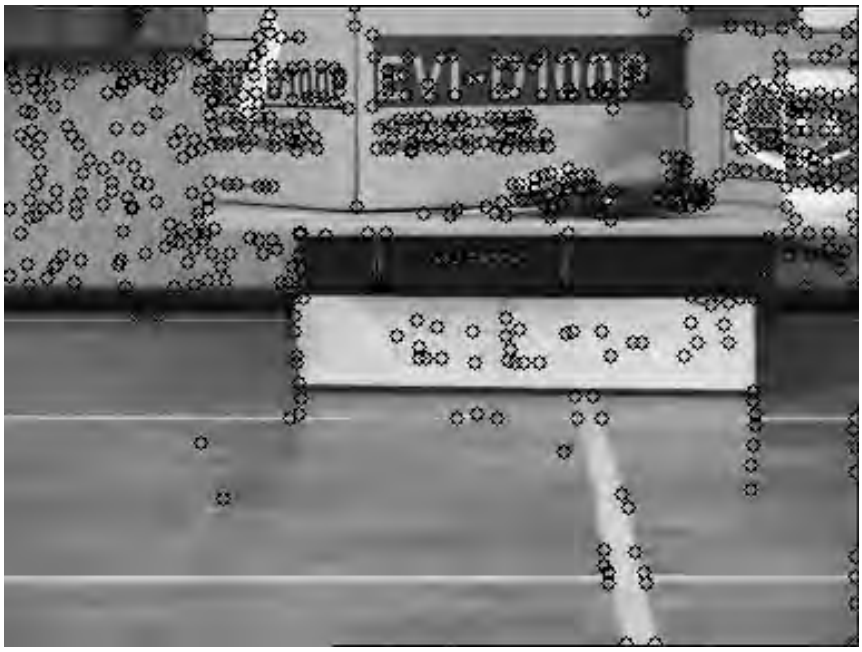

Figure 4.1: Detected corners are shown as black circles on the first image

Figure 4.2: Detected corners are shown as black circles on the second image

correspondence found in this step will be called raw correspondence. Figures 4.3 and 4.4 show the detected raw correspondence between the pair of images with detected corners shown before.

During raw correspondence formation, multiple matches to a point should be avoided. In the implementation this is achieved by deleting the already matched corners from the feature list. Although this is not an optimal choice it has the advantage that less corners are left in each iteration decreasing time complexity. Also the obtained NCC values are thresholded to avoid further mismatches. If a corner does not have a match with an NCC value higher than the threshold then a correspondence is not generated for this corner. A typical value for the threshold is 0.85.

Since the raw correspondence contains a lot of mismatches the obtained correspondence is not suitable for further computations. In fact these mismatches are outliers to the Gaussian error model assumed in much of the algorithms. Especially the linear algorithms find biased solutions in the presence of outliers. Hence a way to classify the raw correspondences as inliers and outliers is needed. The robust algorithms are used to achieve this. The implementation presented in this thesis uses Random Sample Consensus (RANSAC) algorithm introduced in [10] to eliminate outliers. The algorithm is based on drawing minimal sets of random samples and then classifying samples as inliers and outliers to the estimated model. The sample set with most inliers is then taken as the inlier set.

In the case of fundamental matrix computation the minimal set consists of seven point correspondences. With seven correspondences up to three solutions are possible for F. However this does not cause a problem since all of the obtained fundamental matrices can be used in the inlier classification and hence spurious solutions can be eliminated.

One important issue with RANSAC is how to decide if a sample is inlier or not. In the case of F matrix computation the distance to the epipolar line should be used. The epipolar line for a point $\mathbf{x}$ is given as $\mathtt{F}\mathbf{x}$, hence the distance of the corresponding point $\mathbf{x}'$ to the epipolar line is given as

$$\mathbf{d} = \frac{|(\mathtt{F}\mathbf{x})_1 \cdot x' + (\mathtt{F}\mathbf{x})_2 \cdot y' + (\mathtt{F}\mathbf{x})_3|}{\sqrt{(\mathtt{F}\mathbf{x})_1^2 + (\mathtt{F}\mathbf{x})_2^2}} \tag{4.2}$$

where $(\mathtt{F}\mathbf{x})_i$ is the i^{th} component of $\mathtt{F}\mathbf{x}$ and $\mathbf{x}' = [x'\ y'\ 1]^\mathsf{T}$. A similar distance can be calculated between $\mathbf{x}$ and $\mathtt{F}^\mathsf{T}\mathbf{x}'$. The threshold is generally selected as 1.25 but in the implementation it is selected as 1 to ensure the inliers are mismatch free.

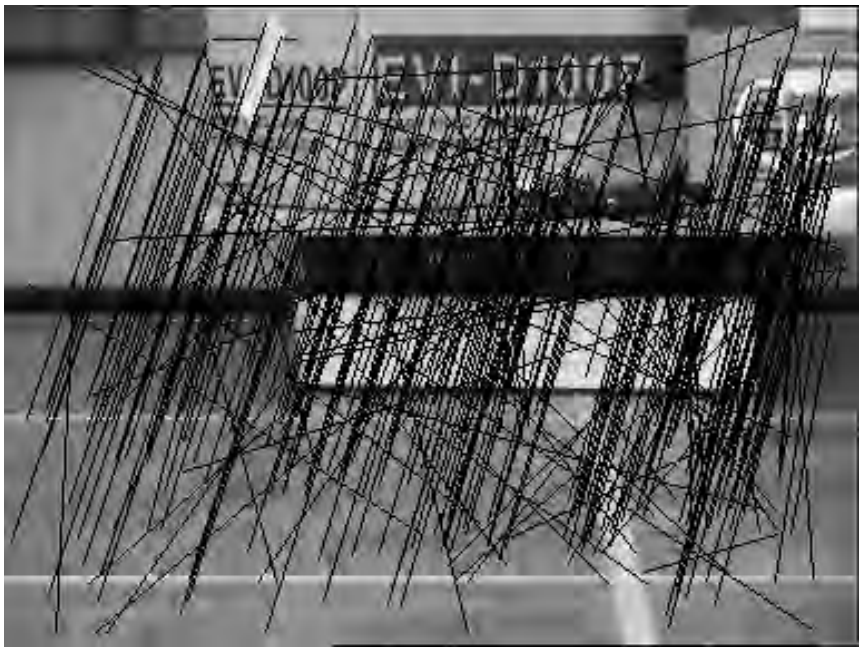

Figure 4.3: Raw correspondance shown as black lines on the first image

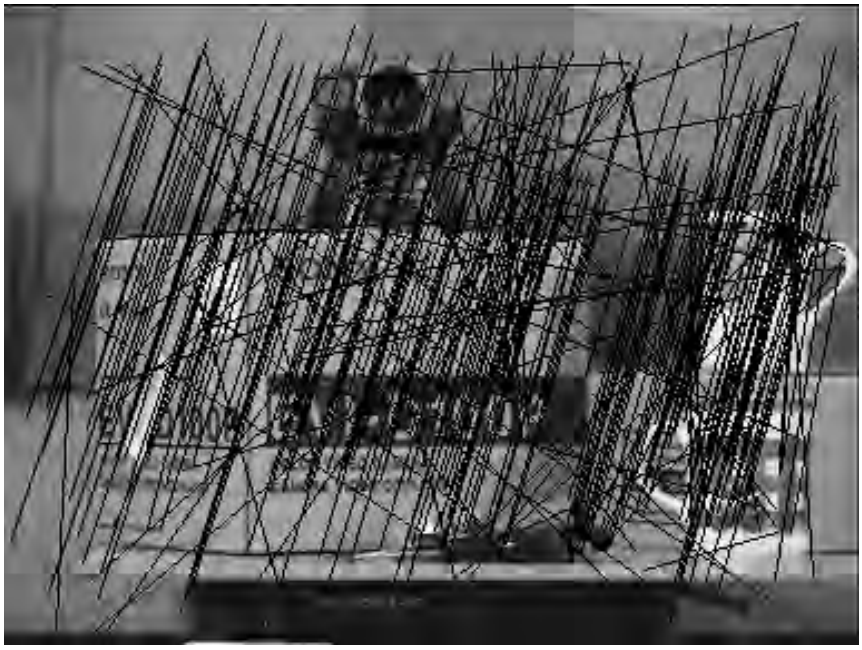

Figure 4.4: Raw correspondance shown as black lines on the second image

The threshold selection is a drawback for the RANSAC algorithm and sometimes its variant MLESAC is used where the threshold is computed automatically from the median of the sample errors. However RANSAC can tolerate much higher ratio of outliers. Hence, it is preferred in the implementation.

Another issue is the number of random sample sets selected before the algorithm stops. If the ratio of inliers is known a priori then it is easy to compute the number of iterations necessary to ensure a high probability that at least one sample set contains all inliers. The required number of iteratons is equal to

$$n = \frac{\log(1-p)}{\log(1-r_{\text{inlier}}^{k})} \tag{4.3}$$

where p is the required probability that all samples are inliers, k is the number of samples in a minimal set and r_{inlier} is the ratio of inliers. p is set to a high value such as 0.99. In the case of F matrix computation with the 7-Point algorithm k is equal to 7. This is also the reason to use the 7-Point algorithm instead of the 8-Point algorithm, i.e. to decrease the number of iterations. In practice the ratio of inliers is not known a priori. In this case an adaptive scheme is used. Initially the number of inliers is set to a low value. In each iteration the number of inliers is calculated and if it is higher than the previous values the number of required iterations is updated with the new inlier ratio using Equation(4.3). Figures 4.5 and 4.6 show the detected inliers. The white circles indicate the corner points and the lines in each image are drawn from the corner point to the corresponding point coordinates.

Also it is better to select the random samples so that they are scattered across the image. This ensures that the calculated F matrix has the same covariance over the image. If this is not ensured then the F matrix is biased to a smaller region [18].

After the RANSAC algorithm a set of inliers are detected. However the F matrix is calculated using only the 7 points in this set. Hence does not reflect the information from all inliers. Here the 8-Point algorithm is used to find the fundamental matrix matching all of the inliers.

In finding the raw correspondence a large search region has been used since there was no information on the possible location of the corresponding point. After the RANSAC algorithm a fundamental matrix is found. Hence the epipolar lines can now be computed which can guide the matching process. This fact is used to find more point matches with a search around the epipolar line. This step ensures that the potential correspondences are not lost due to the raw correspondence search.

Figure 4.5: Detected inliers after running the RANSAC algorithm are shown on the first image

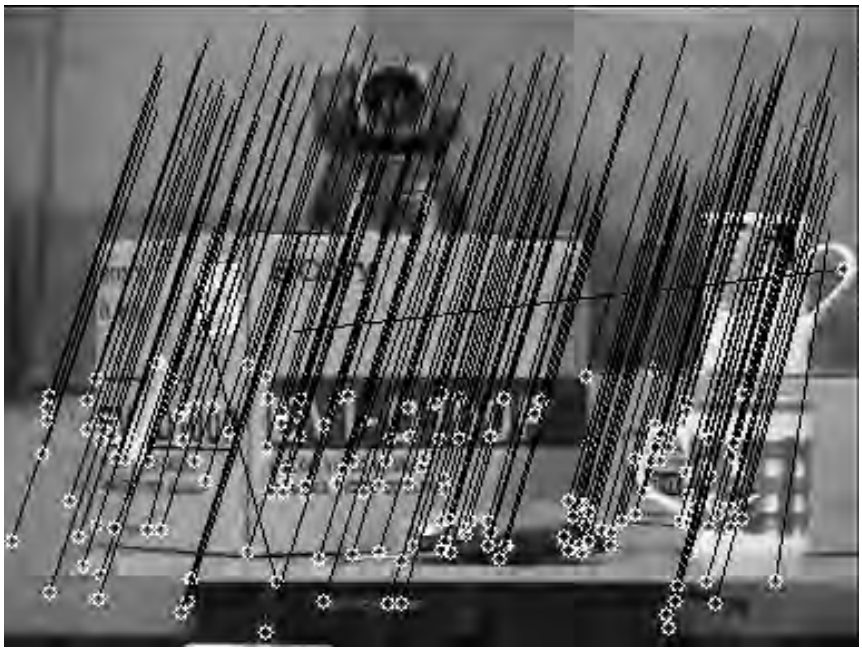

Figure 4.6: Detected inliers after running the RANSAC algorithm are shown on the second image

Although guided matching step finds new corresponding points it may introduce new outliers if the fundamental matrix found after RANSAC is not precise enough. To avoid this the RANSAC step and guided matching are repeated successively until the number of inliers does not change. In practice sometimes the number of inliers oscillate between iteration and a threshold may be used as a stopping criterion. Figures 4.7 and 4.8 shows the results after iteration has converged.

Overall the implemented algorithm can be summarized as follows:

- Detect Harris corner points in both of the images up to sub-pixel resolution.
- Form raw correspondence by a search over a suitably sized window using the normalized cross correlation measure.
- RANSAC algorithm given below is used to compute a set of inliers

  While $p < 0.99$

  - Select seven random correspondence scattered across the image.
  - Calculate a number of F matrices using the 7-Point algorithm and the randomly selected correspondences.
  - For each F matrix find the set of inliers
  - Keep the F matrix with the largest number of inliers
- Use the 8-point algorithm to find a fundamental matrix compatible with all inliers.
- Find more corresponding points using the calculated F matrix and the NCC measure.
- Iterate the RANSAC and guided matching steps until the number of inliers are stable.

For more information on robust algorithm for fundamental matrix computation the reader is referred to [28].

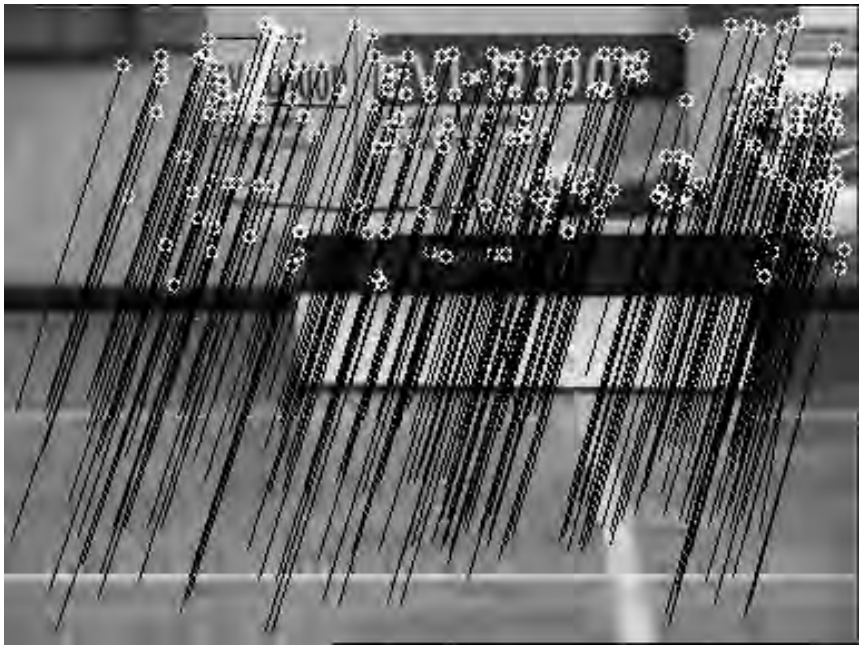

Figure 4.7: Detected inliers after guided matching and RANSAC algorithm is iterated are shown on the first image

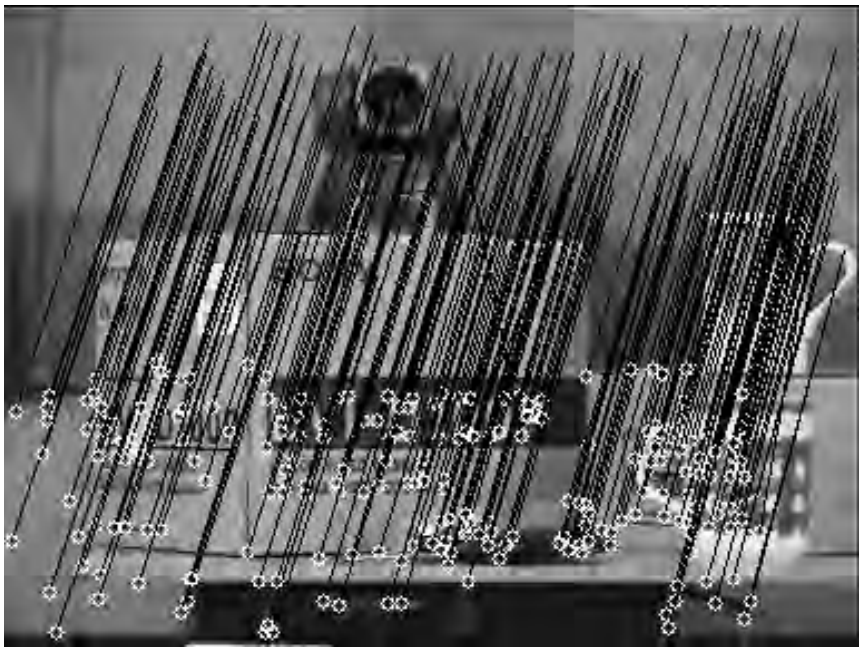

Figure 4.8: Detected inliers after guided matching and RANSAC algorithm is iterated are shown on the second image

4.3 Estimation of Fundamental Matrix Based on Geometric Distance

Up to now the normalized 8-Point and 7-Point algorithms have been used in the estimation of the fundamental matrix. Although the algorithms perform as required when the noise level is low, they have some inherent disadvantages. The error that they minimize is purely algebraic and do not have an immediate geometric meaning. Also the 8-Point algorithm enforces the rank constraint by simply using the singular value decomposition however this method is not optimal since the entries of $\mathtt{F}$ should have different weighting [18]. The rank 2 constraint should be enforced using a suitable parametrization of $\mathtt{F}$.

An alternative method is to minimize a geometric error. In this case the error to be minimized is the distance that the corresponding points should move to ensure that the epipolar constraint (Equation (2.5)) is satisfied exactly. This is called the reprojection error and the error function can be written as

$$\epsilon_{\text{Reproj}} = \sum_{i=1}^{n} \left(\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 + \left\|\mathbf{x}'_i - \hat{\mathbf{x}}'_i\right\|^2 \right) \tag{4.4}$$

subject to

$$\hat{\mathbf{x}}'^{\mathsf{T}} \mathtt{F} \hat{\mathbf{x}} = 0.$$

In practice minimization of Equation (4.4) requires minimization over the coordinates of the corresponding points. This increases the size of the problem significantly and requires special care in the design of the minimization algorithm. Due to these complications it is preferred to minimize an approximation to this error. A first order approximation is the Sampson error [18]. The Sampson error is defined to be

$$\epsilon_{\text{Samps}} = \sum_{i=1}^{n} \frac{(\mathbf{x}_i'^{\mathsf{T}} \mathtt{F} \mathbf{x}_i)^2}{(\mathtt{F}\mathbf{x}_i)_1^2 + (\mathtt{F}\mathbf{x}_i)_2^2 + (\mathtt{F}^{\mathsf{T}}\mathbf{x}'_i)_1^2 + (\mathtt{F}^{\mathsf{T}}\mathbf{x}'_i)_2^2}. \tag{4.5}$$

The minimization of this cost function is relatively easy since the minimization is over the parameters of $\mathtt{F}$.

The Levenberg-Marquardt minimization algorithm is used in the minimization of Equation (4.5). The implementation is based on the method described in [14]. The Levenberg-Marquardt minimization is used commonly in computer vision since it minimizes the squared sum of n functions and the cost function encountered in computer vision is generally of this form. This is due to the fact that generally it is

required to minimize the cost over n entities such as correspondences, pixels and the like. Also the convergence properties of the Levenberg-Marquardt provides the advantages of both Newton's Method and Gradient Descent at a relatively low cost.

Of course the Levenberg-Marquardt minimization uses a linear approximation to the cost function to avoid the computation of the Hessian. Hence, it requires a good initial point to start with. If this is not satisfied the algorithm may converge to a local minimum or may not converge at all.

The computation of the fundamental matrix requires a suitable parametrization of $\mathtt{F}$. A minimization over the nine elements is not attractive because it does not enforce the rank 2 constraint. A better parametrization is obtained as

$$\mathtt{F} = [\mathbf{t}]_\times \mathtt{M}$$

where

$$\mathtt{F} = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}, \quad \mathtt{M} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix}.$$

The skew symmetric nature of $[\mathbf{t}]_\times$ provides the necessary rank deficiency. This requires a parametrization over 12 entries of $\mathbf{t}$ and $\mathtt{M}$. This provides more redundancy than that is necessary but does not create any problem.

The initial solution may be found using the 8-Point algorithm. Then the initial $\mathtt{F}$ is decomposed into $\mathbf{t}$ and $\mathtt{M}$. $\mathbf{t}$ is the left null vector of $\mathtt{F}$ such that $\mathbf{t}^\mathsf{T}\mathtt{F} = \mathbf{0}^\mathsf{T}$ and can be obtained by SVD. The decomposition for $\mathtt{M}$ can be obtained by writing the equality for each component of $\mathtt{F}$. Hence nine linear equations are obtained in the parameters of $\mathtt{M}$ plus a scalar term. The following system of linear equations is formed:

$$\mathbf{Am} = \begin{bmatrix} 0 & 0 & 0 & -t_3 & 0 & 0 & t_2 & 0 & 0 & -f_1 \\ 0 & 0 & 0 & 0 & -t_3 & 0 & 0 & t_2 & 0 & -f_2 \\ 0 & 0 & 0 & 0 & 0 & -t_3 & 0 & 0 & t_2 & -f_3 \\ t_3 & 0 & 0 & 0 & 0 & 0 & -t_1 & 0 & 0 & -f_4 \\ 0 & t_3 & 0 & 0 & 0 & 0 & 0 & -t_1 & 0 & -f_5 \\ 0 & 0 & t_3 & 0 & 0 & 0 & 0 & 0 & -t_1 & -f_6 \\ -t_2 & 0 & 0 & t_1 & 0 & 0 & 0 & 0 & 0 & -f_7 \\ 0 & -t_2 & 0 & 0 & t_1 & 0 & 0 & 0 & 0 & -f_8 \\ 0 & 0 & -t_2 & 0 & 0 & t_1 & 0 & 0 & 0 & -f_9 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ m_8 \\ m_9 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

A solution to this system of equations is found by taking the singular vector corresponding to the least singular value of $\mathtt{A}$. The singular vector is normalized so that the last entry is 1. Although this solution may not be unique it provides a valid starting point for the minimization process.

After an initial solution is found the Levenberg-Marquardt iterations are used to minimize the cost function given by Equation (4.5). An iteration of the algorithm consists of the computation of an increment of the parameters that decreases the cost function. At each step the increment Δ to the parameter vector $\mathbf{p}$ is computed by solving the augmented normal equations

$$\left(\mathtt{J}^{\mathsf{T}}\mathtt{J} + \lambda \mathtt{I}\right)\Delta = -\mathtt{J}^{\mathsf{T}}\epsilon_i \tag{4.6}$$

The parameter vector $\mathbf{p}$ contains the elements of $\mathbf{t}$ and $\mathtt{M}$. It is given as

$$\mathbf{p} = \begin{bmatrix} t_1 & t_2 & t_3 & m_1 & m_2 & m_3 & m_4 & m_5 & m_6 & m_7 & m_8 & m_9 \end{bmatrix}^{\mathsf{T}}.$$

In Equation (4.6) ϵ_i represents the error vector at the i^{th} iteration. $\mathtt{J}$ is the Jacobian matrix and is given as

$$\mathtt{J} = \begin{bmatrix} \frac{\partial e_1}{\partial p_1} & \frac{\partial e_1}{\partial p_2} & \cdots & \frac{\partial e_1}{\partial p_{12}} \\ \frac{\partial e_2}{\partial p_1} & \frac{\partial e_2}{\partial p_2} & \cdots & \frac{\partial e_2}{\partial p_{12}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_n}{\partial p_1} & \frac{\partial e_n}{\partial p_2} & \cdots & \frac{\partial e_n}{\partial p_{12}} \end{bmatrix}$$

where e_i is the error term for the i^{th} correspondence computed using a term in Equation (4.5) and p_i is the i^{th} element of $\mathbf{p}$. The computation of the Jacobian requires the partial derivatives of the error terms with respect to each parameter p_i. Although an analytical formula for the partial derivatives can be obtained a numerical formulation has been used. The numerical derivative is taken to be the forward difference approximation as

$$\frac{\partial e_j}{\partial p_i} = \frac{e_j(p_1, p_2, \ldots, p_i + \delta, \ldots, p_{12}) - e_j(p_1, p_2, \ldots, p_i, \ldots, p_{12})}{\delta}.$$

The value of the increment δ is taken to be $\delta = \max(10^{-6}, 10^{-4}p_i)$ as noted in [14].

The parameter λ in Equation (4.6) plays a vital role in the behavior of the algorithm. When λ is large then the algorithm behaves as the Gradient Descent, when λ is small the algorithm behaves like Newton's Method [18]. At each iteration Equation (4.6) is solved for different values of λ until an acceptable Δ is found that results in a smaller error. The initial value of λ is taken to be $\lambda_0 = 10^{-3}\times$ (the average of the diagonal elements of $\mathtt{J}^{\mathsf{T}}\mathtt{J}$).

Of course the correspondences are normalized before starting the algorithm using the method described in Section 2.5 and then at the end of the algorithm the computed fundamental matrix is denormalized.

Overall the algorithm consists of the following steps:

1. Normalize correspondences.
2. Find an initial solution using the 8-point algorithm.
3. Compute $\mathbf{t}$ and $\mathtt{M}$ such that $\mathtt{F} = [\mathbf{t}]_{\times}\mathtt{M}$.
4. While number of iterations is less than a predefined number
   (a) Compute $\mathtt{J}$, the error vector ϵ_i and the total error E_i.
   (b) Solve for Δ using $\left(\mathtt{J}^{\mathsf{T}}\mathtt{J} + \lambda\mathtt{I}\right)\Delta = -\mathtt{J}^{\mathsf{T}}\epsilon_i$.
   (c) Compute new $\mathtt{F} = [\mathbf{t}]_{\times}\mathtt{M}$ and E_{i+1}.
   (d) If $(E_{i+1} < E_i)$ then set $\lambda = \lambda/10$ and increase the iteration counter else set $\lambda = \lambda \times 10$ and go to step 4(b).
5. Compute the unnormalized F matrix.

The evaluation of the algorithms is then based on the residual error which in this case given by

$$\epsilon_{res} = \frac{1}{n}\sum_{i=1}^{n}\left(\|\mathbf{x}'_i - \mathtt{F}\mathbf{x}_i\|^2 + \|\mathbf{x}_i - \mathtt{F}^{\mathsf{T}}\mathbf{x}'_i\|^2\right). \tag{4.7}$$

Table 4.1 shows the residual error for the 8-Point algorithm and the nonlinear minimization. Results show that the nonlinear algorithm performs slightly better than the 8-Point algorithm and the performance of the 8-point algorithm is still suitable when accuracy is not at utmost importance.

4.4 Linear Method Using Absolute Dual Quadric

The method of absolute dual quadric requires the computation a projective reconstruction. This requires the computation of a set of camera projection matrices $\mathtt{P}_j$ for each view j. A pair camera projection matrices can be computed from the fundamental matrix as

$$\mathtt{P}_0 = [\mathtt{I}_{3\times 3} \mid \mathbf{0}], \quad \mathtt{P}_j = \left[\left[\mathbf{e}'\right]_{\times}\mathtt{F}_{0j} \mid \mathbf{e}'\right] \tag{4.8}$$

Table 4.1: Residual pixel error for a test sequence

$\mathbf{e}_{\text{res}}$ (8-Point)	$\mathbf{e}_{\text{res}}$ (Non-Linear)
0.188109	0.187877
0.197193	0.193001
0.146169	0.146029
0.104161	0.104009
0.136168	0.136168
0.157491	0.157486

where $\mathtt{F}_{0j}$ is the fundamental matrix between views 0 and j and $\mathbf{e}'$ is the corresponding left epipole [18]. However given $\mathtt{F}_{0j}$ this choice of camera matrices is not unique. The above set of projection matrices can be transformed by a projective transformation of 3D and the new set of projection matrices also have the same fundamental matrix. Due to this projective ambiguity, a reconstruction using the calculated camera matrices will have the same ambiguity hence called a projective reconstruction.

Given an image sequence the first camera projection matrix is taken to be $\mathtt{P}_0$ so that a initial projective frame for the sequence is obtained. Then using the fundamental matrices $\mathtt{F}_{0j}$ the camera matrices for other views are obtained. To remove the projective ambiguity the form of the absolute dual quadric needs to be recovered in the computed projective frame. Once the absolute dual quadric is computed then it is easy to compute the calibration for each view. Note that absolute dual quadric is a quadric in three dimensions, so it is represented by a symmetric matrix of dimensions 4×4 and also it is of rank 3.

A linear algorithm to constrain the absolute dual quadric is introduced in [23] so that given enough projective camera matrices one can solve for the calibration of the cameras even when the intrinsic parameters are varying. The constraining equation is that the absolute dual quadric projects to the dual image of the absolute conic in each view:

$$\omega_j^* = \mathtt{P}_j \mathtt{Q}_\infty^* \mathtt{P}_j^\mathsf{T} \qquad \text{for each } j = 0, \ldots, n. \tag{4.9}$$

The dual image of the absolute conic (ω_j^*) is directly related to the internal calibration

as

$$\omega_j^* = \mathtt{K}_j\mathtt{K}_j^{\mathsf{T}}.$$

So the constraints on the internal parameters can be written on constraints on ω^*. And using Equation (4.9) these constraints can be used to constrain $\mathtt{Q}_\infty^*$. Assume that the form of $\mathtt{K}$ is

$$\mathtt{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then the form of ω^* is

$$\omega^* = \begin{bmatrix} f_x^2 + s^2 + u_0^2 & sf_y + u_0v_0 & u_0 \\ sf_y + u_0v_0 & f_y^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{bmatrix}. \tag{4.10}$$

If the coordinates of the principal point is taken to be at the center of the image then the coordinates of each point can be translated so that $u_0 = v_0 = 0$. Further if skew is assumed to be zero and aspect ratio is taken to be unity then the form of ω^* becomes

$$\omega^* = \begin{bmatrix} f^2 & 0 & 0 \\ 0 & f^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.11}$$

Considering Equation (4.9) and Equation (4.11) together the following constraints are avaliable on $\mathtt{Q}_\infty^*$:

$$\begin{aligned} (\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{1,2} &= 0 \\ (\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{1,3} &= 0 \\ (\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{2,3} &= 0 \\ (\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{1,1} &= (\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{2,2} \end{aligned} \tag{4.12}$$

where $(\mathtt{P}_j\mathtt{Q}_\infty^*\mathtt{P}_j^{\mathsf{T}})_{k,l}$ is the component at the k^{th} row and l^{th} column. Since these constraints are linear in the entries of $\mathtt{Q}_\infty^*$ if enough number of these equations are stacked on top of each other then the obtained system of linear equations can be solved for $\mathtt{Q}_\infty^*$. Then using Equation (4.9), ω_j^* can be obtained for each view. Using Cholesky decomposition of ω_j^*, calibration is obtained for each view. For more information on Cholesky decomposition see [11]. The C++ code used in the implementation of Cholesky decomposition is given in Appendix F.

One drawback of the algorithm is that the planar motion is a degenerate motion for the autocalibration methods based on the recovery of the absolute conic or the

absolute dual quadric. A unique solution is not possible if the motion consists of translations over a plane and rotations around a screw axis that is perpendicular to the translation plane. Hence when capturing autocalibration sequences this situation should be avoided. These kind of motion sequences are called critical motion sequences and have been studied extensively [25, 26].

4.5 Results and Conclusion

The algorithm is tested on a sequence of images captured by the stereo system mounted on the robot platform. Since planar motion is a degenerate motion sequence for the algorithm, the robot platform also makes a tilt motion in the first couple of frames to avoid degeneracy. The fundamental matrices between the first view and all other views have been computed. These fundamental matrices are then used in the computation of focal lengths in each view. Since the aspect ratio is taken to be unity and the principal point at the center of the image only the focal lengths in the x-direction are estimated. The resulting focal lengths and the manual calibration results are presented in Table 4.2.

Table 4.2: The results of the autocalibration algorithm on the first sequence

View	$\mathbf{f_x}$ **(Manual Calibration)**	$\mathbf{f_x}$ **(Autocalibration)**
0	454.2	75.04
1	454.2	112.7
2	454.2	182.9
3	454.2	113.0
4	454.2	121.8
5	454.2	136.8
6	454.2	34.50

The results show that the algorithm has failed to find a reliable focal length. The tests are then performed on images captured by a digital camera. Table 4.3 shows the resulting focal lengths.

Although the focal length for view 1 is close to the expected value a consistent focal length could not be obtained.

Table 4.3: The results of the autocalibration algorithm on the second sequence

View	$\mathbf{f_x}$ **(Manual Calibration)**	$\mathbf{f_x}$ **(Autocalibration)**
0	1137.49	5909.0
1	1137.49	1338.2
2	1137.49	839.72

There may be several reasons for the failure of the algorithm. The first one is the problem of critical motion sequences. The sequences are obtained trying to avoid such situations but autocalibration algorithms are known to perform poorly also when the sequence is close to a critical motion sequence. This is not very easy to avoid especially when using the robot platform with a fixed set of cameras. If motion type is known to be strictly planar then that information should be used if possible.

Also the sequence length presents another problem. While setting up a projective set of camera matrices we have used a simple strategy. The fundamental matrices between the first and all other views are computed using point matches between the first view and others. Hence the projective frame is consistent for all views. However in longer sequences when the rotation and translation are large the first and the last views have little overlapping area which prevents consistent matching between views. This severely limits the motion content of the whole sequence.

The strategy for obtaining a projective reconstruction of camera matrices presented in [23] is quite different, the fundamental matrices are computed between neighboring views and then the obtained projective frames are matched to each other using 3D homographies between projectively reconstructed points. However this is costly and the obtained structure may require further optimization over all reconstructed points and cameras. Although this approach is attractive for obtaining a reconstruction from the sequence, it is not suitable for focal length estimation. Chapter 5 details the implementation of another autocalibration method based on pure rotations of the cameras.

Chapter 5

Autocalibration from Rotating Cameras

5.1 Introduction

It is possible to perform autocalibration when there is only camera rotation. The method is introduced in [15] and extended to the case of varying intrinsic parameters in [7]. In this chapter the implementation of the method and comparison of alternative routes in the implementation is discussed.

Since it is necessary that the camera motion has no translational component pan-tilt-zoom (PTZ) cameras are used in capturing image sequences used in this chapter. The camera model used in the study is Sony EVI-D100P.

Sections 5.2 and 5.3 describe the algorithm in the case of constant and varying calibration parameters. The necessary constraints and the form of the image of the absolute conic where these constraints arise is presented. Section 5.4 describes how the required homographies are computed in a robust manner. Section 5.5 shows how the rotations around a specific axis affect the computed homography and discusses possible problems with the procedure. The last section shows the results of the autocalibration process and discusses the validity of the obtained results.

5.2 Linear Algortihm for Constant Calibration

In this section it is assumed that the images are captured by a rotating camera and the camera has the same internal parameters throughout the sequence. The images are denoted by $\mathtt{I}_i$ where $i = 0, \ldots, (n-1)$ and n is the number of images.

Section 2.7 shows that when the camera motion has only a rotational component, the acquired images are related by a 3×3 homography matrix. Assume that the first image is located at the origin and hence has the projection matrix

$$\mathtt{P}_0 = \mathtt{K}[\mathtt{I} \mid \mathbf{0}].$$

Since the internal parameters are the same for all images the projection matrix for $\mathtt{I}_i$ is given as

$$\mathtt{P}_i = \mathtt{K}[\mathtt{R}_i \mid \mathbf{0}]$$

where $\mathtt{R}_i$ is the rotation between $\mathtt{I}_0$ and $\mathtt{I}_i$. Then the matrix $\mathtt{H}_i = \mathtt{K}\mathtt{R}_i\mathtt{K}^{-1}$ is the homography between $\mathtt{I}_0$ and $\mathtt{I}_i$.

The homograhies between the first image and all other images can be easily computed using the methods described in Section 5.4. Then the computed homographies are used to constrain the internal calibration parameters as follows. Writing

$$\mathtt{R}_i = \mathtt{K}^{-1}\mathtt{H}_i\mathtt{K}$$

and using the fact that $\mathtt{R}_i\mathtt{R}_i^{\mathsf{T}} = \mathtt{I}$ due to orthogonality

$$\mathtt{K}\mathtt{K}^{\mathsf{T}} = \mathtt{H}_i\mathtt{K}\mathtt{K}^{\mathsf{T}}\mathtt{H}_i^{\mathsf{T}} \tag{5.1}$$

is obtained. Taking inverse of the both sides of the Equation(5.1) and writing $(\mathtt{K}\mathtt{K}^{\mathsf{T}})^{-1} = \omega$

$$\omega = \mathtt{H}_i^{-\mathsf{T}}\omega\mathtt{H}_i^{-1} \tag{5.2}$$

is the constraint on the internal parameters [18]. Note that ω represents the image of the absolute conic and Equation (5.2) states the the image of the absolute conic is constant when transferred by the related homography.

Equation (5.2) can be used to form a set of linear equations on the entries of ω. Since ω is symmetric there are 6 unknowns. Each homography provides 6 equations in the entries of ω. Hence in general 2 views are enough to solve for ω. After ω is computed one can use Cholesky decomposition to obtain $\omega = \mathtt{U}\mathtt{U}^{\mathsf{T}}$. Then $\mathtt{K}$ is found as $\mathtt{K} = \mathtt{U}^{-\mathsf{T}}$.

For more information on Cholesky decomposition see [11]. The C++ code used in the implementation of Cholesky decomposition is given in Appendix F.

There is one issue that must be taken into account with the described algorithm. Since the matrix $\mathtt{H}$ is homogeneous the estimated homography is of arbitrary scale. However since the homography considered here is conjugate to a rotation it should have unit determinant [15]. Hence the estimated homographies should be scaled as

$$\mathtt{H}' = \frac{\mathtt{H}}{\sqrt[3]{\det(\mathtt{H})}}.$$

If the form of $\mathtt{H}_i^{-1}$ is

$$\mathtt{H}_i^{-1} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$$

where $\mathbf{h}_i$ is the i^{th} column of $\mathtt{H}_i^{-1}$ then the constraints are of the form

$$\omega(i,j) = \mathbf{h}_i^{\mathsf{T}} \omega \mathbf{h}_j \tag{5.3}$$

where $\omega(i,j)$ is the entry in the i^{th} row and j^{th} column of ω.

5.3 Linear Algortihm for Varying Calibration

The algorithm described in Section 5.2 is valid for only when the calibration is constant throughout the sequence. However when there is some information on the calibration parameters the method can be extended to the case of varying calibration parameters [7]. In particular zero-skew and known aspect ratio are two commonly used constraints.

Let $\mathtt{K}_i$ denote the calibration matrix for view i. Then the form of Equation (5.2) for varying calibration is

$$\omega_i = \mathtt{H}_i^{-\mathsf{T}} \omega_0 \mathtt{H}_i^{-1} \tag{5.4}$$

where ω_i is the image of the absolute conic in I_k. If skew is zero and the aspect ratio is unity the following constraints are used:

$$\begin{aligned} \omega_i(1,2) &= 0 \\ \omega_i(1,1) &= \omega_i(2,2). \end{aligned} \tag{5.5}$$

If enough of these constraints are formed then ω_0 can be found. The calibration in other views are found by using Equation (5.4).

The constraints in Equation (5.5) can also be used in the case of constant calibration if the skew is zero and the aspect ratio is unity.

5.4 Homography Estimation

The presented algorithms above assume that a set of homographies between I_0 and all other views is computed. The computation is based on a well known algorithm, namely the Direct Linear Transformation (DLT), which was described in Section 2.7.

The computation of a homography requires a set of point matches between the images. The initial matches are found by using the NCC measure as described in Section 4.2. However this set of matches contains outliers which should be eliminated. Again the RANSAC algorithm is used in this case to compute a homograhy and a set of inliers compatible with this homography. The approach is very similar to the one presented in Section 4.2 with a few differences. A point match is accepted as an inlier if it satisfies both $\|\mathbf{x}' - \mathtt{H}\mathbf{x}\|^2 < d$ and $\left\|\mathbf{x} - \mathtt{H}^{-1}\mathbf{x}'\right\|^2 < d$ where d is the distance threshold typically set to 1.5. In practice the iteration of RANSAC and guided matching steps is found to be unnecessary for homography computation since a good set of inliers is obtained in the first run. The search window used to find the initial matches is set to the whole image to allow for computation of the homography even for large amounts of rotation.

Note that the minimum number of point correspondences necessary to compute a homography is 4. Which means that $k = 4$ in Equation (4.3) in the case of robust point matching using homographies ($k = 7$, in the case of matching points using a fundamental matrix relation). Hence the maximum number of necessary iterations is a lot smaller in the case of matching points by homography computation when compared to point matching using the fundamental matrix.

The overall algorithm is as follows:

- Detect Harris corner points in both of the images up to sub-pixel resolution.
- Form raw correspondence by a search over a suitably sized window using the normalized cross correlation measure.
- Use RANSAC to find inliers.

  While $p < 0.99$

  - Select four random correspondences scattered across the image.
  - Calculate a homography using these correspondences.
  - Find the set of inliers

- Keep the H matrix with the largest number of inliers

- Use the DLT algorithm to find a homography compatible with all inliers.
- Find more corresponding points using the calculated homography and the NCC measure.

Figures 5.1 and 5.2 show the detected corners in two of the images from a sequence used in the autocalibration tests. Figures 5.3 and 5.4 show the result of the guided matching step. Note that there is no false match despite the fact that there is significant rotation. In practice, point matching with RANSAC, based on homography estimation is much more robust then point matching with fundamental matrix computation since the uncertainty is reduced to a point once an initial homography is computed. In the case of fundamental matrix estimation the uncertainty is a line and wrong matches along the line are possible.

5.5 An Analysis of Pan-Tilt Motion

In this section an analysis of pan-tilt motion is done to gain further insight to the constraints that the computed homography puts on the intrinsic parameters. Let $\mathtt{R}_x$ and $\mathtt{R}_y$ denote rotations around the x and y axes respectively. Since a PTZ camera can only tilt (rotate around x-axis) and/or pan (rotate around y-axis), the overall rotation is a combination of these two principal rotations. The forms of $\mathtt{R}_x$ and $\mathtt{R}_y$ are

$$\mathtt{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \qquad \mathtt{R}_y = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}. \tag{5.6}$$

where θ is the tilt angle and ϕ is the pan angle. Assume that the camera has zero-skew and hence

$$\mathtt{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now the forms of the homographies arising from a rotation around each one of the principal axes can be computed to be

$$\mathtt{H}_x = \mathtt{K}\mathtt{R}_x\mathtt{K}^{-1} = \begin{bmatrix} 1 & \frac{u_0 s_\theta}{f_y} & -u_0 - \frac{u_0 v_0 s_\theta}{f_y} + u_0 c_\theta \\ 0 & \frac{f_y c_\theta + v_0 s_\theta}{f_y} & -\frac{f_y v_0 c_\theta + v_0^2 s_\theta}{f_y} - f_y s_\theta + v_0 c_\theta \\ 0 & \frac{s_\theta}{f_y} & -\frac{v_0 s_\theta}{f_y} + c_\theta \end{bmatrix} \tag{5.7}$$

Figure 5.1: Detected corners in a sequence with camera rotation is shown on the first image

Figure 5.2: Detected corners in a sequence with camera rotation is shown on the second image

Figure 5.3: The resultant inliers after guided matching is shown on the first image

Figure 5.4: The resultant inliers after guided matching is shown on the second image

and

$$\mathtt{H}_y = \mathtt{K}\mathtt{R}_y\mathtt{K}^{-1} = \begin{bmatrix} \frac{f_x c_\phi - u_0 s_\phi}{f_x} & 0 & -\frac{f_x u_0 c_\phi - u_0^2 s_\phi}{f_x} + f_x s_\phi + u_0 c_\phi \\ -\frac{v_0 s_\phi}{f_x} & 1 & \frac{u_0 v_0 s_\phi}{f_x} - v_0 + v_0 c_\phi \\ -\frac{s_\phi}{f_x} & 0 & \frac{u_0 s_\phi}{f_x} + c_\phi \end{bmatrix} \tag{5.8}$$

where $s_\theta = \sin(\theta)$, $c_\theta = \cos(\theta)$ and $s_\phi = \sin(\phi)$, $c_\phi = \cos(\phi)$. An analysis of the above homographies reveals a few facts about the autocalibration method. When there is only a rotation about the x-axis the computed homography is independent of f_x. This results in the fact that the computed homography can not be used to constrain f_x. Similarly a rotation around the y-axis results in a homography which is independent of f_y. Therefore these two cases are degenerate and without known aspect ratio they lead to an incomplete calibration.

A more important fact is that in both cases the focal length is coupled with the sine or the cosine of the rotation angle. And if the angle of rotation is small then the computed homographies become nearly independent of the focal lengths. Hence a poor calibration is expected when the rotation is small. The following rule of thumb is then should be taken into account:

- The image that results in the largest rotation angle should be used as $\mathtt{I}_0$. Other choices lead to smaller rotations and they do not constrain the focal lengths in the best possible way.

However automated matching is not possible if the degree of overlap between images is small. If the rotation is so large that automated matching is problematic between $\mathtt{I}_0$ and $\mathtt{I}_k$ then an intermediate image $\mathtt{I}_l$ can be used to compute the homography between $\mathtt{I}_0$ and $\mathtt{I}_k$. The homography should be computed as

$$\mathtt{H}_{0k} = \mathtt{H}_{lk}\mathtt{H}_{0l}.$$

Of course this leads to error propagation and should be used only when direct computation is not possible.

The results show that the analysis presented in this section is also valid in practice.

5.6 Results and Conclusion

Several experiments are performed on three sets of images. The thumbnail views of the sequences are given in Appendix E. At each setting the camera is also calibrated

using the manual calibration method described in Chapter 3. The results will be compared with the manual calibration results. The basis of the discussion will be on the recovered focal lengths. The image size is 768×576 and the principal point coordinates should be around (384, 288). However it should be noted that the reconstruction process is generally not very sensitive to small changes in principal point coordinates.

There are three algorithms that has been used to achieve autocalibration. The first one assumes only constant calibration and will be referred as *Constant*. The second one assumes square pixels (Zero skew and unit aspect ratio) and constant calibration parameters. This will be referred to as *Constant + Square*. The last algorithm assumes square pixels and the calibration can vary hence, it will be referred as *Vary + Square*.

The first sequence will be referred to as *PTRot1*. This sequence is composed of 4 images. There is slight pan and tilt in each of the images. Table 5.1 shows the manual calibration and autocalibration results for the three algorithms when all three of the homographies have been used. A dash in a column means that due to the assumptions on the parameter it has not been calculated. By inspecting the table, the following observations can be made:

- The method *Constant* finds a more reasonable focal length in the x direction then the y direction. The reason for this is the larger pan motion in the images which constrain only f_x as indicated in Section 5.5. The focal lengths could not be detected with good precision. This indicates that the angle of rotation may be insufficient. The x coordinate of the principal point is very reasonable.
- The method *Constant + Square* performs very poorly in estimating any of the parameters.
- The method *Vary + Square* finds a very reasonable principal point but the estimated focal length is not acceptable.

The second sequence will be referred to as *PTRot2*. There are 12 images in the sequence. The dominating motion in the sequence is panning. The camera is continuously panned from left to right while adding some small tilt motions at each frame. The rotation between the last and the first frame is almost 2 times the largest motion in the sequence *PTRot1*. Table 5.2 shows the results of the autocalibration when all homographies have been used. Since this time the rotation is larger, method

Table 5.1: Autocalibration results for sequence *PTRot1*

Method	$\mathbf{f_x}$	$\mathbf{f_y}$	**skew**	α	$\mathbf{u_0}$	$\mathbf{v_0}$
Manual	1035.36	1035.1	0.0248	90.000	380.771	277.972
Constant	1244.13	1402.86	112.3	89.920	386.047	368.715
Constant + Square	570.994	—	—	—	183.599	127.091
Vary + Square	1341.56	—	—	—	320.246	269.635

Constant finds a very good f_x. Again since there is not enough tilt motion f_y and v_0 are estimated poorly. The skew is also very large. When square pixels are assumed for constant calibration the estimated focal length becomes somehow biased but the error remains below 10%. The algorithm *Vary + Square* find a very accurate calibration. The focal length is almost precisely found and the estimated principle point is within expected bounds.

Table 5.2: Autocalibration results for sequence *PTRot2* when all homographies have been used

Method	$\mathbf{f_x}$	$\mathbf{f_y}$	**skew**	α	$\mathbf{u_0}$	$\mathbf{v_0}$
Manual	1175.35	1175.46	-0.167	90.000	408.547	296.843
Constant	1120.62	717.278	592	89.310	400.131	166.284
Constant + Square	1268.73	—	—	—	316.529	350.155
Vary + Square	1203.37	—	—	—	434.686	307.4

When compared to results of *PTRot1* sequence, the accuracy gained in *Ptrot2* sequence can be due to either the larger rotation or the increased number of homographies. To test this only the first two and last three images are used in the following experiment. The number of homographies is four but there is significant rotation. Table 5.3 shows the results. The estimated focal lengths do not change significantly and the calibration from *Vary + Square* method is still very good although the principal point is less accurate. This shows that the rotation angle plays a more important role than the number of homographies.

The third sequence contains four images and will be referred to as *PTRot3*. There

Table 5.3: Autocalibration results for sequence *PTRot2* when four homographies have been used

Method	$\mathbf{f_x}$	$\mathbf{f_y}$	**skew**	α	$\mathbf{u_0}$	$\mathbf{v_0}$
Manual	1175.35	1175.46	-0.167	90.000	408.547	296.843
Constant	1056.82	701.58	461	89.419	351.244	136.472
Constant + Square	1272.94	—	—	—	306.913	374.347
Vary + Square	1133.39	—	—	—	447.35	348.495

is only rotation around the y axis and the amount of rotation is so large that the homography between I_0 and I_3 can not be computed directly. The intermediate image I_2 has been used to compute the combined homography as described in Section 5.5. The total rotation is twice the largest rotation in sequence *PTRot2*. The results are shown in Table 5.4. The *Constant* method finds a quite reasonable focal length in x direction but the focal length in y direction is not acceptable. This is of course expected since there is no tilt motion. When square pixels are enforced using method *Constant + Square* the obtained ω is not positive definite hence Cholesky decomposition fails to find the intrinsic calibration matrix. This is a common problem in many autocalibration methods when the calibration is not constrained enough. Again the results for *Vary + Square* method is very good. The focal length is almost equal to the one found by manual calibration and the principal point location is very close to expected values.

Table 5.4: Autocalibration results for sequence *PTRot3*

Method	$\mathbf{f_x}$	$\mathbf{f_y}$	**skew**	α	$\mathbf{u_0}$	$\mathbf{v_0}$
Manual	1224.05	1227.42	-26.83	90.022	348.9	281.0
Constant	1159.67	63.72	-31.77	90.462	372.2	98.2
Constant + Square	ω is not positive definite					
Vary + Square	1213.59	—	—	—	378.98	257.75

Comparing the above tables the best choice seems to be the *Vary + Square* method even when the calibration is known to be constant. Of course this requires that skew

is zero and aspect ratio is at least known and scaled to unity. In robotics applications these two are very reasonable assumptions. The rotation angle should be chosen large. A large tilt angle is not always possible and simply panning gives very good results.

Chapter 6

Conclusion

6.1 Summary of the Results

In this study a manual calibration method and two autocalibration methods are implemented. Each method is tested on real world image sequences. The results of the algorithms are also presented.

All of the calibration methods use point correspondences and require the computation of a two view relation, either in the form of a homography or a fundamental matrix. Different methods to compute two view relations are implemented and these are then used in robust computation of corresponding points. The matching of the points are done using fully automated processes. The results of this matching process are shown in the related chapters.

In Chapter 3 implementation of a manual calibration system based on the method presented in [33] has been given. The results of the calibration experiments with different camera settings have been presented and shown to be consistent with theoretical values. The obtained calibration results are further tested using 3D measurements on a test pattern. Two known lengths and one angle have been measured. The results show that the system is calibrated with good precision and the obtained measurements are reliable enough.

In Chapter 4 a method based on the work of [23] is implemented to autocalibrate the cameras using constraints on the Absolute Dual Quadric. To obtain near real-time results the setting of the projective frame uses a simpler approach than the work presented in [23] where a 3D reconstruction was aimed. However the results are

not satisfying and the implemented method suffers from a few degeneracies in the fundamental matrix estimation between views that are far away from each other.

In the light of the experience gained from the experiments on the Absolute Dual Quadric method another autocalibration method is implemented in Chapter 5. The method uses images captured by a purely rotating camera as described in [15]. The corresponding point matching is done based on homography computation. The results have been shown to be much more stable than the method presented in Chapter 4. Of course this is only possible for images captured by a purely rotating camera. The method is implemented using different assumptions and constraints. The results show that when the rotation is large enough, using the variable calibration and square pixel assumptions provides the best results. The results of the method is compared with the results of the manual calibration and the focal lengths are found to be nearly the same and the principal point positions are within expected boundaries.

6.2 Discussion and Future Work

Although the implemented manual calibration and autocalibration from rotating system performs at the expected precision, there are numerous possible enhancements to the study presented in this study.

The stereo system used in the study is a fixed platform of two cameras and the autocalibration experiments are done one monocular PTZ cameras. The stereo system should be upgraded to one that uses the PTZ cameras. Of course this requires additional electronics to control the both cameras from the same PC terminal.

The manual calibration algorithm works well but due to poor lightning conditions the corners of the calibration pattern can not be detected reliably at all images. Although this is rare it causes unnecessary delays since new images are needed to be captured. This is almost always due to lost contrast in the black squares due to reflections from the light sources. The images can be preprocessed or an on-line strategy in capturing the images can be used where the image is captured only when all corners are detected.

The external calibration step requires more elaborate methods. Although the presented methods works fine at most occasions sometimes user intervention is necessary to eliminate spurious solutions. More information on this can be found in the discus-

sion of Section 3.6.

A resolution versus depth test can be performed on the manual calibration results to provide more insight on the calibration accuracy. A new test pattern can be used providing more angle and length measurements. Also the points on the test pattern may be detected in an automated way which is a feature the current system lacks.

The autocalibration efforts should be directed more on the pure rotation methods since the results seem more promising. There are less ambiguities and the matching step is much more robust and also faster. The autocalibration results presented here only uses a comparison with the focal lengths of the manual calibration results. When the stereo system is constructed using PTZ cameras, 3D measurements can be obtained and these measurements can be used to evaluate the autocalibration results.

Appendix A

Parametrization of Rotations

The parametrization of rotations are done using a 3-vector of the form

$$\mathbf{r} = \theta \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

where θ is the rotation angle in radians and $[r_1\, r_2\, r_3]^{\mathsf{T}}$ is a unit vector representing the rotation axis. Hence there is a need for conversion between the matrix form $\mathtt{R}$ and the vector form $\mathbf{r}$, and vice versa.

The conversion from $\mathbf{r}$ to $\mathtt{R}$ is easily achieved using the Rodrigues formula ([32]) as

$$\mathtt{R} = \begin{bmatrix} \cos\theta + r_1^2(1-\cos\theta) & r_1 r_2(1-\cos\theta) - r_3\sin\theta & r_2\sin\theta + r_1 r_3(1-\cos\theta) \\ r_3\sin\theta + r_1 r_2(1-\cos\theta) & \cos\theta + r_2^2(1-\cos\theta) & -r_1\sin\theta + r_2 r_3(1-\cos\theta) \\ -r_2\sin\theta + r_1 r_3(1-\cos\theta) & r_1\sin\theta + r_2 r_3(1-\cos\theta) & \cos\theta + r_3^2(1-\cos\theta) \end{bmatrix}.$$

To achieve conversion from $\mathtt{R}$ to $\mathbf{r}$ the rotation matrix is written as

$$\mathtt{R} = \mathrm{e}^{[\mathbf{r}]_\times \theta} = \mathtt{I}_{3\times 3} + [\mathbf{r}]_\times \sin\theta + [\mathbf{r}]_\times^2 (1-\cos\theta)$$

where $[\mathbf{r}]_\times$ is the matrix form of vector cross product as given in Equation (2.3). Now writing

$$\mathrm{trace}(\mathtt{R}) = 3 + 0 + (-r_3^2 - r_2^2 - r_3^2 - r_1^2 - r_2^2 - r_1^2)(1-\cos\theta)$$

and since $\mathbf{r}$ is a unit vector

$$\mathrm{trace}(\mathtt{R}) = 3 + (1-\cos\theta)(-2) = 1 + 2\cos\theta$$

then

$$\theta = \cos^{-1}\left(\frac{\mathrm{trace}(\mathtt{R}) - 1}{2}\right).$$

To obtain the rotation axis the following equation is solved for a unit vector

$$\mathtt{R} - \mathtt{R}^{\mathsf{T}} = 2\ [\mathbf{r}]_{\times} \sin\theta.$$

However in the case where $\sin\theta = 0$ the above equation can not be solved and the equation $\mathtt{R} = \mathtt{I}_{3\times 3} + [\mathbf{r}]_{\times} \sin\theta + [\mathbf{r}]_{\times}^{2} (1 - \cos\theta)$ should be used with the known value of θ to get the rotation axis.

Appendix B

Radial Undistortion

In the undistortion problem a distorted point with polar coordinates $(\tilde{r}, \theta)$ and the distortion coefficients κ_1 and κ_2 are given. The aim is to find the point (r, θ) that minimizes

$$\varphi = \left[\tilde{r} - r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)\right]^2 .$$

Taking the derivative with respect to r

$$\varphi' = -2\left(1 + 3\kappa_1 r^2 + 5\kappa_2 r^4\right)\left[\tilde{r} - r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)\right]$$

is obtained. The following algorithm can be used to find the required point:

1. Set $r = \tilde{r}$
2. Calculate $e = \tilde{r} - r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$
3. While $|e| > tolerance$
   (a) Calculate $\delta r = -\varphi' = 2\,e\left(1 + 3\kappa_1 r^2 + 5\kappa_2 r^4\right)$
   (b) Set $r = r + 0.3\,\delta r$
   (c) Calculate $e = \tilde{r} - r\left(1 + \kappa_1 r^2 + \kappa_2 r^4\right)$
4. Calculate the point (r, θ) taking into account the quadrant of $(\tilde{r}, \theta)$

The coefficient of update used in step (3.b) is found to be 0.3 by heuristic methods. The larger numbers tend to create oscillations and smaller ones converge slowly. The algorithm generally converges in 3 or 4 iterations.

Appendix C

MetaPost Code for The Test Pattern

The following code fragment can be used to generate the test pattern used in the manual calibration tests.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TestPattern.mp
def checkerbox(expr x, n) =
   fill unitsquare scaled n shifted (x);
   fill unitsquare scaled n shifted (x - (n, n));
   draw unitsquare scaled (2 * n) shifted (x - (n, n));
enddef;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
beginfig(1);
pair a[];
a0 = (15cm, 5cm);
a1 = a0 + (0cm, 15cm);
a2 = a0 + 15cm * dir(90 + 37);
checkerbox(a0, 4cm);
checkerbox(a1, 4cm);
checkerbox(a2, 4cm);
endfig;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix D

Manual Calibration Sequences

The following figures in this chapter show the image sequences used in the manual calibration tests. The images are captured by holding the calibration pattern at different distance and angles. The method requires that the calibration pattern plane is not parallel in any pair of images.

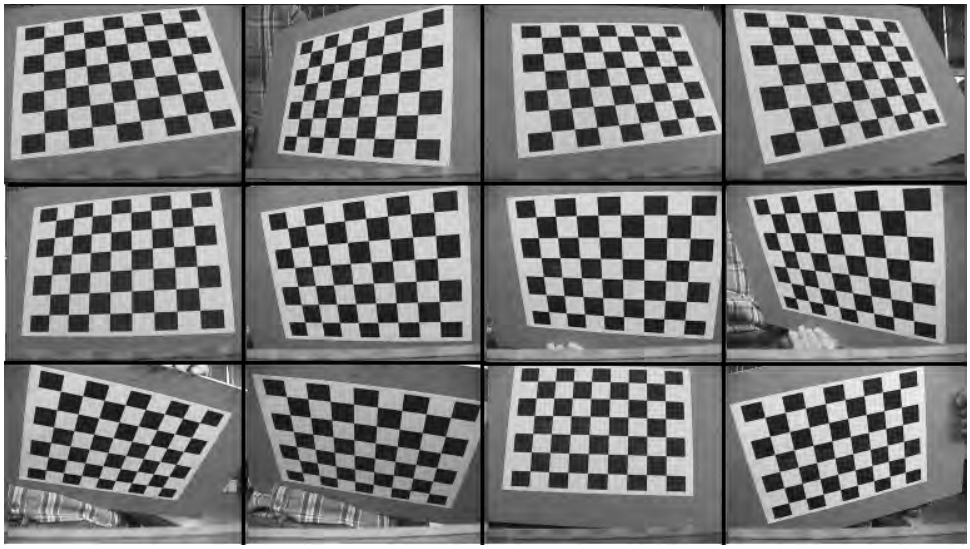

Figure D.1: Thumbnail images of *LeftSeq1*

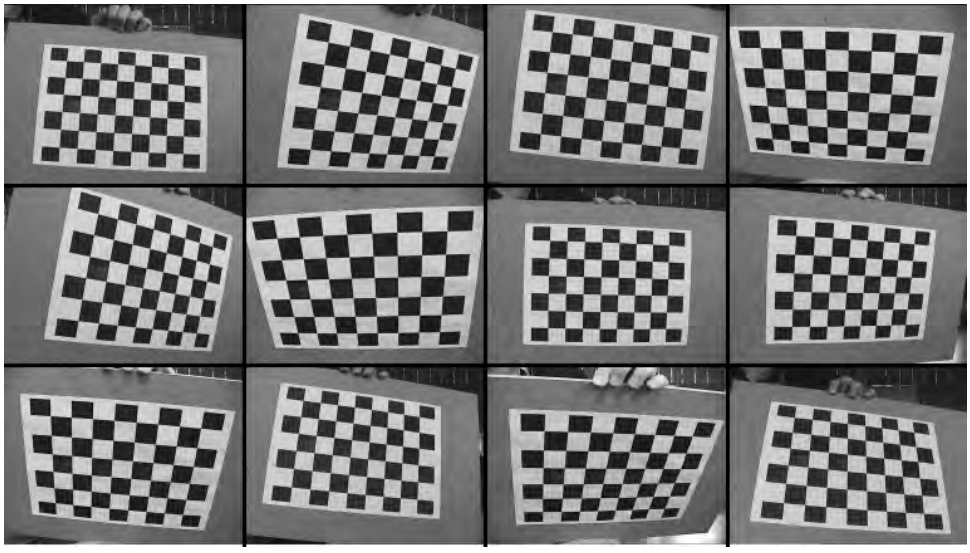

Figure D.2: Thumbnail images of *RightSeq1*

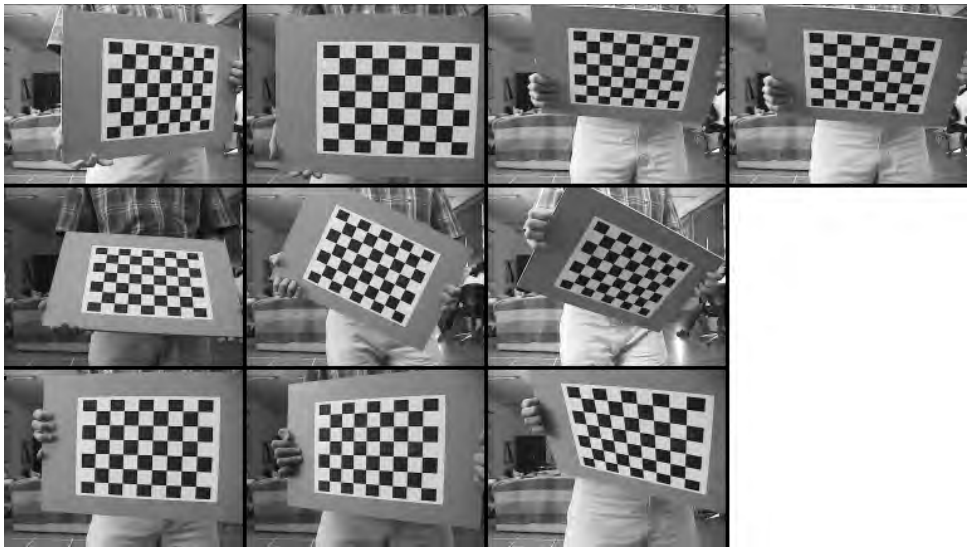

Figure D.3: Thumbnail images of *LeftSeq2*

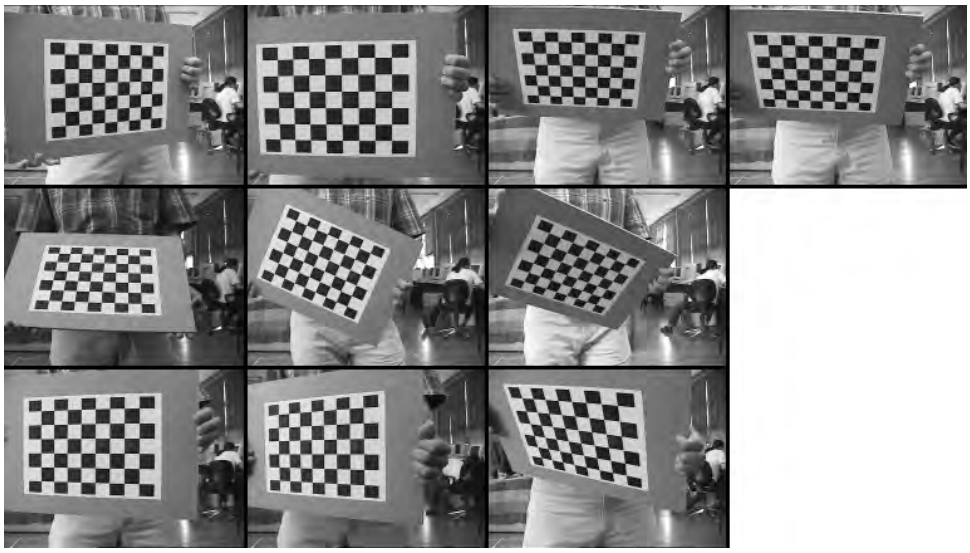

Figure D.4: Thumbnail images of *RightSeq2*

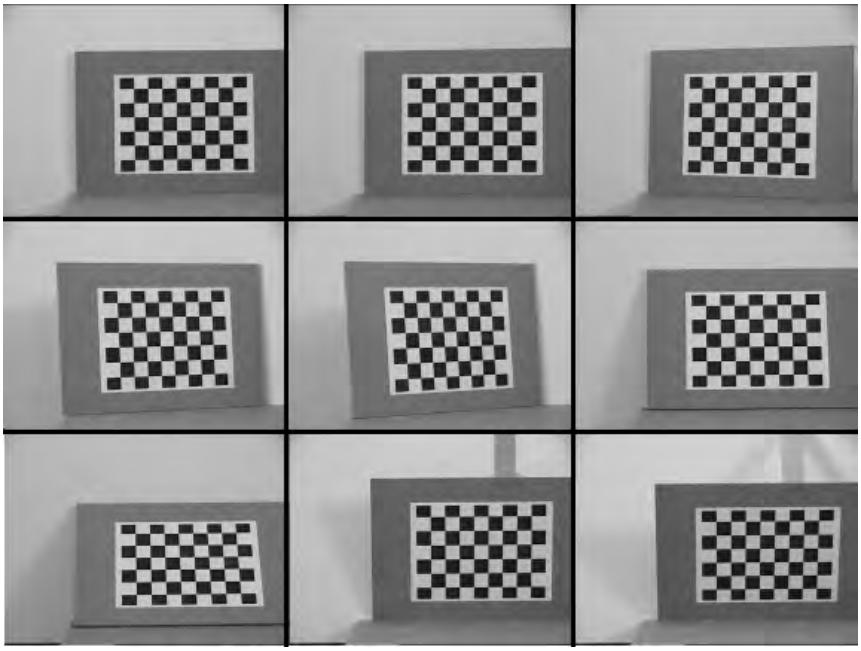

Figure D.5: Thumbnail images of *LeftSeq3*

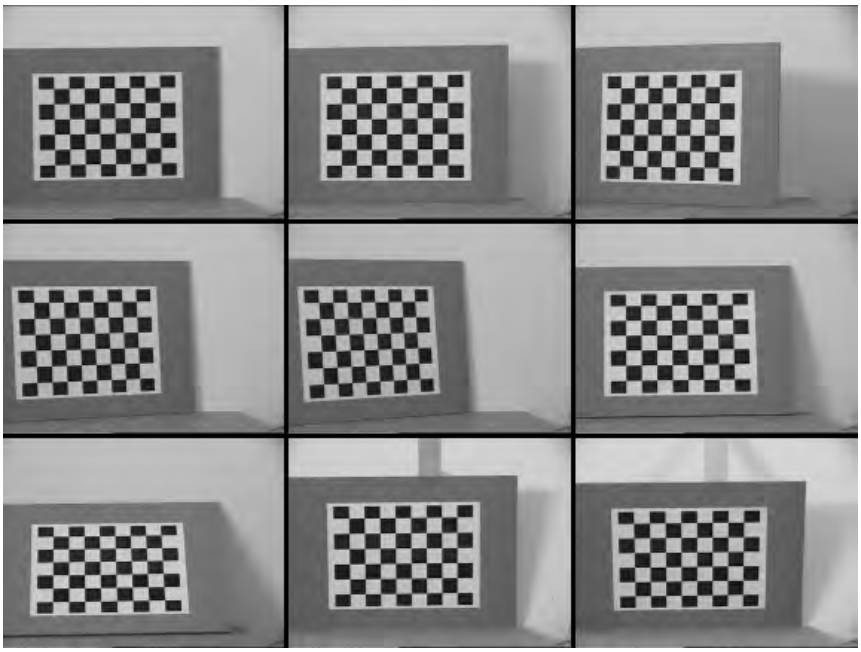

Figure D.6: Thumbnail images of *RightSeq3*

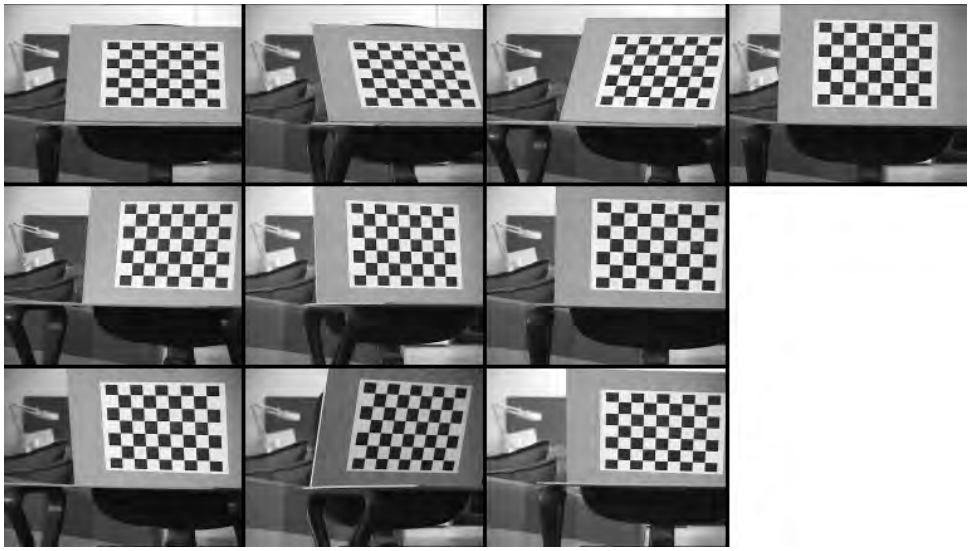

Figure D.7: Thumbnail images of *LeftSeq4*

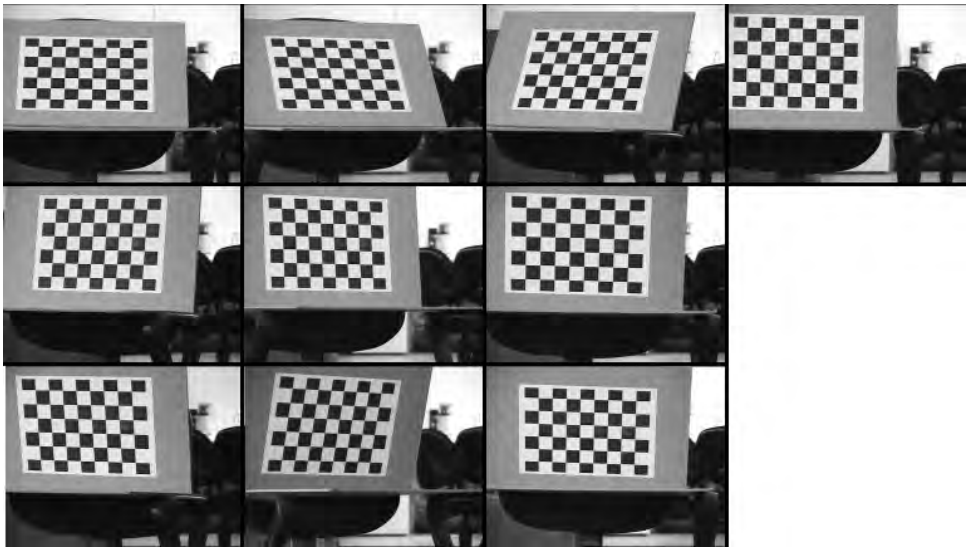

Figure D.8: Thumbnail images of *RightSeq4*

Appendix E

Autocalibration Sequences with Pure Rotation

The following figures show the image sequences that has been used in the autocalibration tests. The motion in the sequences is pure rotation as required by the algorithm. The sequences have been captured by a single PTZ camera.

Figure E.1: Thumbnail images of *PTRot1*

Figure E.2: Thumbnail images of *PTRot2*

Figure E.3: Thumbnail images of *PTRot3*

Appendix F

C++ Code for Cholesky Decomposition

The following code fragment computes the cholesky decomposition of a positive definite symmetric matrix and is based on the pseudo code given in [11]. Note that the `Matrix` class has two methods `Matrix::getData(i, j)` and `Matrix::setData(i, j, val)` which are used to obtain and set the value of the element at row `i` and column `j` respectively.

```
// Compute cholesky decomposition for symmetric positive
// definite matrix M.
// If successful M = UUT where U is lower triangular and returns 0.
// If M is not square then returns -1
// If M is not positive definite then returns -2.
// Uses only the diagonal and lower left part of M.
int Cholesky(LynxLib::Math::Matrix &M, LynxLib::Math::Matrix &U)
{
  if(M.getNumCols() != M.getNumRows())
    return -1;

  U.setZero();
  int n = M.getNumRows();
  LynxLib::Math::Matrix v(n, 1);

  for(int j = 0; j < n; j++)
```

```
  {
    for(int i = j; i < n; i++)
    {
      v.setData(i, 0, M.getData(i, j));
    }
    for(int k = 0; k <= (j - 1); k++) for(int i = j; i < n; i++)
    {
      double temp;
      temp = v.getData(i, 0) - U.getData(j, k) * U.getData(i, k);
      v.setData(i, 0, temp);
    }
    for(int i = j; i < n; i++)
    {
      double temp = v.getData(j, 0);
      if(temp < 0)
        return -2;
      else
        temp = std::sqrt(temp);
      U.setData(i, j, v.getData(i, 0) / temp);
    }
  }

  return 0;
}
```

References

[1] M. Agrawal and L.S. Davis. Camera calibration using spheres: a semi-definite programming approach. In *Proc. Ninth IEEE International Conference on Computer Vision*, volume 2, pages 782–789, 2003.

[2] Zafer Arıcan. Vision based robot localization using artificial and natural landmarks. Master's thesis, Middle East Technical University, 2004.

[3] P. A. Beardsley and A. Zisserman. Affine calibration of mobile devices. In *Europe-China workshop on Geometrical modelling and Invariants for Computer Vision*, pages 214–221, 1995.

[4] M. Brooks, L. de Agapito, D. Q. Huynh, and L. Baumela. Direct methods for self-calibration of a moving stereo head. In *Proc. of the Fourth European Conference on Computer Vision*, volume II, pages 415–426, 1996.

[5] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, November 2000.

[6] G. Csurka, D. Demirdjian, A. Ruf, and R. Horaud. Closed-form solutions for the euclidian calibration of a stereo rig. In *Proc. Fifth European Conference on Computer Vision*, pages 426–442, 1998.

[7] L. de Agapito, R. I. Hartley, and E. Hayman. Linear calibration of a rotating and zooming camera. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 15–21, 1999.

[8] O. Faugeras. Stratification of 3-dimensional vision: Projective, affine, and metric representations. *JOSA-A*, 12(3):465–484, March 1995.

[9] O. Faugeras, Q.-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381–395, 1981.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[12] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Prentice Hall, second edition, 2001.

[13] C. J. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference*, pages 147–151, Manchester, 1988.

[14] R. I. Hartley. Euclidian reconstruction from uncalibrated views. In J. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision*, pages 237–256. Springer-Verlag, 1994.

[15] R. I. Hartley. Self-calibration from multiple views with a rotating camera. In *Proc. European Conference on Computer Vision*, pages 471–478. Springer-Verlag, 1994.

[16] R. I. Hartley. In defense of the 8-point algorithm. *PAMI*, 19(6):580–593, 1997.

[17] R. I. Hartley and P. Sturm. Triangulation. In *Proc. Conference Computer Analysis of Images and Patterns*, Prague, Czech Republic, 1995.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[19] J. Heikkilä and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[20] B. K. P. Horn. Tsai's camera calibration method revisited. 2000.

[21] J. Knight, A. Zisserman, and I. Reid. Linear auto-calibration for ground plane motion. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[22] Q.-T. Luong and O.D. Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–76, 1996.

[23] M. Pollefeys and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proc. Sixth International Conference on Computer Vision*, pages 90–96, 1998.

[24] M. Pollefeys and L. Van Gool. Stratified self-calibration with the modulus constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):707–724, 1999.

[25] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1100–1105, 1997.

[26] P. Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In *Proc. British Machine Vision Conference*, 1999.

[27] Uğur Topay. 3d scene reconstruction from uncalibrated images. Master's thesis, Middle East Technical University, 2002.

[28] P. H. S. Torr. *Motion segmentation and outlier detection.* PhD thesis, University of Oxford, 1995.

[29] B. Triggs. Autocalibration and the absolute quadric. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[30] R. Y. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal Robotics and Automation*, RA-3(4):323–344, 1987.

[31] İlkay Ulusoy. *Active stereo vision: depth perception for navigation, environmental map formation and object recognition.* PhD thesis, Middle East Technical University, 2003.

[32] Eric W. Weisstein et al. Rodrigues' rotation formula. *http://mathworld.wolfram.com/RodriguesRotationFormula.html*, 07 2004.

[33] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.